Streaming Algorithms for Maximizing Monotone Submodular Functions under a Knapsack Constraint*

Chien-Chung Huang[1], Naonori Kakimura[†2], and Yuichi Yoshida[‡3]

1 **CNRS, École Normale Supérieure, Paris, France**
villars@gmail.com
2 **Department of Mathematics, Keio University, Yokohama, Japan**
kakimura@math.keio.ac.jp
2 **National Institute of Informatics, Tokyo, Japan**
yyoshida@nii.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 $(0.363-\varepsilon)$-approximation algorithm, requiring only a single pass through the data; moreover, we propose a $(0.4-\varepsilon)$-approximation algorithm requiring a constant number of passes through the data. The required memory space of both algorithms depends only on the size of the knapsack capacity and ε.

1998 ACM Subject Classification G.2.1 Combinatorics

Keywords and phrases submodular functions, single-pass streaming, multiple-pass streaming, constant approximation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2017.11

1 Introduction

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

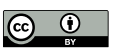
$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T).$$

A 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, 11, 18], sensor location [12], text summarization [16, 17], and maximum entropy sampling [14].

* A full version of the paper is available at http://www.di.ens.fr/~cchuang/work/streaming_knapsack.pdf.

† Partly supported by JST ERATO Grant Number JPMJER1305, and JSPS KAKENHI Grant Number JP17K00028.

‡ Partly supported by JST ERATO Grant Number JPMJER1305 and JSPS KAKENHI Grant Number JP17H04676



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 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

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

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

The problem of maximizing a monotone submodular function under a knapsack constraint is classical and well-studied. First introduced by Wolsey [20], the problem is known to be NP-hard but can be approximated within the factor of (close to) $1 - 1/e$; see e.g., [3, 10, 13, 8, 19].

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. 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 also 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, 7, 21]. Badanidiyuru *et al.* [2] discuss several interesting and useful functions where the oracle can be implemented using a small subset of the entire ground set E.

We note that the problem, under the streaming model, has so far not received its deserved attention in the community. Prior to the present work, we are aware of only two: for the special case of cardinality constraint, Badanidiyuru *et al.* [2] gave a single-pass $(1/2 - \varepsilon)$-approximation algorithm; for the general case of a knapsack constraint, Yu *et al.* [21] gave a single-pass $(1/3 - \varepsilon)$-approximation algorithm, both using $O(K \log(K)/\varepsilon)$ space.

We now state our contribution.

▶ **Theorem 1.** *For the problem* (1)*,*
1. *there is a single-pass streaming algorithm with approximation ratio* $4/11 - \varepsilon \approx 0.363 - \varepsilon$*,*
2. *there is a multiple-pass streaming algorithm with approximation ratio* $2/5 - \varepsilon = 0.4 - \varepsilon$*.*
Both algorithms use $O(K \cdot \mathrm{poly}(\varepsilon^{-1})\mathrm{polylog}(K))$ *space.*

Our Technique

We begin by a straightforward generalization of the algorithm of [2] for the special case of cardinality constraint (Section 2). This algorithm proceeds by adding a new item into the current set only if its marginal-ratio (its marginal return with respect to the current set divided by its size) exceeds a certain threshold. This algorithm performs well when all items in OPT are relatively small in size, where OPT is an optimal solution. However, in general, it only gives $(1/3 - \varepsilon)$-approximation. Note that this technique can be regarded as a variation of the one in [21]. To obtain better approximation ratio, we need new ideas.

The difficulty in improving this algorithm lies in the following case: A new arriving item that is relatively large in size, passes the marginal-ratio threshold, and is part of OPT, but its addition would cause the current set to exceed the capacity K. In this case, we are forced to throw it away, but in doing so, we are unable to bound the ratio of the function value of the current set against that of OPT properly.

We propose a branching procedure to overcome this issue. Roughly speaking, when the function value of the current set is large enough (depending on the parameters), we create a secondary set. We add an item to the secondary set only if it passes the marginal-ratio threshold (with respect to the original set) but its addition to the original set would violate the size constraint. In the end, whichever set achieves the higher value is returned. In a way, the secondary set serves as a "back-up" with enough space in case the original set does not have it, and this allows us to bound the ratio properly. Sections 3 and 4 are devoted to explaining this branching algorithm, which gives $(4/11 - \varepsilon)$-approximation with a single pass.

We note that the main bottleneck of the above singe-pass algorithm lies in the situation where there is a large item in OPT whose size exceeds $K/2$. In Section 5, we show that we can first focus on only the large items (more specifically, those items whose size differ from the largest item in OPT by $(1 + \varepsilon)$ factor) and choose $O(1)$ of them so that at least one of them, along with the rest of OPT (excluding the largest item in it), gives a good approximation to $f(\mathrm{OPT})$. Then in the next pass, we can apply a modified version of the original single-pass algorithm to collect small items. This multiple-pass algorithm gives a $(2/5 - \varepsilon)$-approximation.

We remark that the proofs of some lemmas and theorems are omitted due to the page limitation, which can be found in the full version of this paper.

Related Work

Maximizing a monotone submodular function subject to various constraints is a subject that has been extensively studied in the literature. We are unable to give a complete survey here and only highlight the most representative and 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 [4, 9, 13, 8, 15] and the references therein. In the streaming setting, other than the knapsack constraint that we have discussed before, there are also works considering a matroid constraint. Chakrabarti and Kale [5] gave 1/4-approximation; Chekuri *et al.* [7] gave the same ratio. Very recently, for the special case of partition matroid, Chan *et al.* [6] improved the ratio to 0.3178.

Notation

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 $f(e \mid S) = f(S + e) - f(S)$. We frequently use the following, which is immediate from the diminishing marginal return property:

▶ **Proposition 2.** *Let $f : 2^E \to \mathbb{R}_+$ be a monotone submodular function. For two subsets $S \subseteq T \subseteq E$, it holds that $f(T) \leq f(S) + \sum_{e \in T \setminus S} f(e \mid S)$.*

2 Single-Pass $(1/3 - \varepsilon)$-Approximation Algorithm

In this section, we present a simple $(1/3 - \varepsilon)$-approximation algorithm that generalizes the algorithm for a cardinality constraint in [2]. This algorithm will be incorporated into several other algorithms introduced later.

Algorithm 1

procedure MarginalRatioThresholding(α, v) ▷ $\alpha \in (0,1], v \in \mathbb{R}_+$
  $S := \emptyset$.
  while item e is arriving **do**
    if $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}$ and $c(S+e) \leq K$ **then** $S := S + e$.
  return S.

2.1 Thresholding Algorithm with Approximate Optimal Value

In this subsection, we present an algorithm MarginalRatioThresholding, which achieves (almost) 1/3-approximation given a (good) approximation v to $f(\mathrm{OPT})$ for an optimal solution OPT. This assumption is removed in Section 2.2.

Given a parameter $\alpha \in (0,1]$ and $v \in \mathbb{R}_+$, MarginalRatioThresholding attempts to add a new item $e \in E$ to the current set $S \subseteq E$ if its addition does not violate the knapsack constraint and e passes the *marginal-ratio threshold condition*, i.e.,

$$\frac{f(e \mid S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}. \tag{2}$$

The detailed description of MarginalRatioThresholding is given in Algorithm 1.

Throughout this subsection, we fix $\tilde{S} = \mathsf{MarginalRatioThresholding}(\alpha, v)$ as the output of the algorithm. Then, we have the following lemma.

▶ **Lemma 3.** *The following hold:*

(1) *During the execution of the algorithm, the current set $S \subseteq E$ always satisfies $f(S) \geq \alpha v c(S)/K$. Moreover, if an item $e \in E$ passes the condition (2) with the current set S, then $f(S+e) \geq \alpha v c(S+e)/K$.*

(2) *If an item $e \in E$ fails the condition (2), i.e., $\frac{f(e|S)}{c(e)} < \frac{\alpha v - f(S)}{K - c(S)}$, then we have $f(e \mid \tilde{S}) < \alpha v c(e)/K$.*

An item $e \in \mathrm{OPT}$ is not added to $\tilde{S}$ if either e does not pass the condition (2), or its addition would cause the size of S to exceed the capacity K. We name the latter condition as follows:

▶ **Definition 4.** An item $e \in \mathrm{OPT}$ is called *bad* if e passes the condition (2) but the total size exceeds K when added, i.e., $f(e \mid S) \geq \frac{\alpha v - f(S)}{K - c(S)}$, $c(S+e) > K$ and $c(S) \leq K$, where S is the set we have just before e arrives.

The following lemma says that, if there is no bad item, then we obtain a good approximation.

▶ **Lemma 5.** *If $v \leq f(\mathrm{OPT})$ and there have been no bad item, then $f(\tilde{S}) \geq (1-\alpha)v$ holds.*

Proof. By the submodularity and the monotonicity, we have $v \leq f(\mathrm{OPT}) \leq f(\mathrm{OPT} \cup \tilde{S}) \leq f(\tilde{S}) + \sum_{e \in \mathrm{OPT} \setminus \tilde{S}} f(e \mid \tilde{S})$. Since we have no bad item, $f(e \mid \tilde{S}) \leq \alpha v c(e)/K$ for any $e \in \mathrm{OPT} \setminus \tilde{S}$ by Lemma 3 (2). Hence, we have $v \leq f(\tilde{S}) + \alpha v$, implying $f(\tilde{S}) \geq (1-\alpha)v$. ◀

Consider an algorithm Singleton, which takes the best singleton as shown in Algorithm 2. If some item $e \in \mathrm{OPT}$ is bad, then, together with $\tilde{S}' = \mathsf{Singleton}()$, we can achieve (almost) 1/3-approximation.

▶ **Theorem 6.** *We have $\max\{f(\tilde{S}), f(\tilde{S}')\} \geq \min\{\alpha/2, 1-\alpha\}v$. The right-hand side is maximized to $v/3$ when $\alpha = 2/3$.*

Algorithm 2

procedure Singleton()
  $S := \emptyset$.
  while item e is arriving **do**
    if $f(e) > f(S)$ **then** $S := \{e\}$.
  return S.

Algorithm 3

procedure DynamicMRT(ε, α) ▷ $\varepsilon, \alpha \in (0, 1]$
  $\mathcal{V} := \{(1+\varepsilon)^i \mid i \in \mathbb{Z}_+\}$.
  For each $v \in \mathcal{V}$, set $S_v := \emptyset$.
  while item e is arriving **do**
    $m := \max\{m, f(e)\}$.
    $\mathcal{I} := \{v \in \mathcal{V} \mid m \le v \le Km/\alpha\}$.
    Delete S_v for each $v \notin \mathcal{I}$.
    for each $v \in \mathcal{I}$ **do**
      if $\frac{f(e|S_v)}{c(e)} \ge \frac{\alpha v - f(S_v)}{K - c(S_v)}$ and $c(S_v + e) \le K$ **then** $S_v := S_v + e$.
  return S_v for $v \in \mathcal{I}$ that maximizes $f(S_v)$.

Proof. If there exists no bad item, we have $f(\tilde{S}) \ge (1-\alpha)v$ by Lemma 5. Suppose that we have a bad item $e \in E$. Let $S_e \subseteq E$ be the set just before e arrives in MarginalRatioThresholding. Then, we have $f(S_e + e) \ge \alpha v c(S_e + e)/K$ by Lemma 3 (1). Since $c(S_e + e) > K$, this means $f(S_e + e) \ge \alpha v$. Since $f(S_e + e) \le f(S_e) + f(e)$ by submodularity, one of $f(S_e)$ and $f(e)$ is at least $\alpha v/2$. Thus $f(\tilde{S}) \ge f(S_e) \ge \alpha v/2$ or $f(e) \ge \alpha v/2$. ◀

Therefore, if we have $v \in \mathbb{R}_+$ with $v \le f(\mathrm{OPT}) \le (1+\varepsilon)v$, the algorithm that runs MarginalRatioThresholding$(2/3, v)$ and Singleton() in parallel and chooses the better output has the approximation ratio of $\frac{1}{3(1+\varepsilon)} \ge 1/3 - \varepsilon$. The space complexity of the algorithm is clearly $O(K)$.

2.2 Dynamic Updates

MarginalRatioThresholding requires a good approximation to $f(\mathrm{OPT})$. This requirement can be removed with dynamic updates in a similar way to [2]. We first observe that $\max_{e \in S} f(e) \le f(\mathrm{OPT}) \le K \max_{e \in S} f(e)$. So if we are given $m = \max_{e \in S} f(e)$ in advance, a value $v \in \mathbb{R}_+$ with $v \le f(\mathrm{OPT}) \le (1+\varepsilon)v$ for $\varepsilon \in (0, 1]$ exists in the guess set $\mathcal{I} = \{(1+\varepsilon)^i \mid m \le (1+\varepsilon)^i \le Km, i \in \mathbb{Z}_+\}$. Then, we can run MarginalRatioThresholding for each $v \in \mathcal{I}$ in parallel and choose the best output. As the size of $\mathcal{I}$ is $O(\log(K)/\varepsilon)$, the total space complexity is $O(K \log(K)/\varepsilon)$.

To get rid of the assumption that we are given m in advance, we consider an algorithm, called DynamicMRT, which dynamically updates m to determine the range of guessed optimal values. More specifically, it keeps the (tentative) maximum value $\max f(e)$, where the maximum is taken over the items e arrived so far, and keeps the approximations v in the interval between m and Km/α. The details are provided in Algorithm 3. We have the following guarantee.

▶ **Theorem 7.** *For $\varepsilon \in (0, 1]$, the algorithm that runs* DynamicMRT$(\varepsilon, 2/3)$ *and* Singleton() *in parallel and outputs the better output is a $(1/3 - \varepsilon)$-approximation streaming algorithm with a single pass for the problem* (1)*. The space complexity of the algorithm is $O(K \log(K)/\varepsilon)$.*

Algorithm 4

1: **procedure** BranchingMRT($\varepsilon, \alpha, v, c_1, b$) ▷ $\varepsilon, \alpha \in (0,1]$, $v \in \mathbb{R}_+$, and $c_1, b \in [0, 1/2]$
2: $S := \emptyset$.
3: $\lambda := \frac{1}{2}\alpha(1-b)v$.
4: **while** item e is arriving **do**
5: Delete e with $c(e) > \min\{(1+\varepsilon)c_1, 1/2\}K$.
6: **if** $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}$ and $c(S+e) \leq K$ **then** $S := S + e$.
7: **if** $f(S) \geq \lambda$ **then break** // leave the While loop.
8: Let $\hat{e}$ be the latest added item in S.
9: **if** $c(S) \geq (1-b)K$ **then** $S'_0 := \{\hat{e}\}$ **else** $S'_0 := S$.
10: $S' := S'_0$.
11: **while** item e is arriving **do**
12: Delete e with $c(e) > \min\{(1+\varepsilon)c_1, 1/2\}K$.
13: **if** $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}$ and $c(S+e) \leq K$ **then** $S := S + e$.
14: **if** $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}$ and $c(S+e) > K$ **then**
15: **if** $f(S') < f(S'_0 + e)$ **then** $S' := S'_0 + e$.
16: **return** S or S' whichever has the larger function value.

3 Improved Single-Pass Algorithm for Small-Size Items

Let $\mathrm{OPT} = \{o_1, o_2, \ldots, o_\ell\}$ be an optimal solution with $c(o_1) \geq c(o_2) \geq \cdots \geq c(o_\ell)$. The main goal of this section is achieving $(2/5 - \varepsilon)$-approximation, assuming that $c(o_1) \leq K/2$. The case with $c(o_1) > K/2$ will be discussed in Section 4.

3.1 Branching Framework with Approximate Optimal Value

We here provide a framework of a branching algorithm BranchingMRT as Algorithm 4. This will be used with different parameters in Section 3.2.

Let v and c_1 be (good) approximations to $f(\mathrm{OPT})$ and $c(o_1)/K$, respectively, and let $b \leq 1/2$ be a parameter. The value c_1 is supposed to satisfy $c_1 \leq c(o_1)/K \leq (1+\varepsilon)c_1$, and hence we ignore items $e \in E$ with $c(e) > \min\{(1+\varepsilon)c_1, 1/2\}K$. The basic idea of BranchingMRT is to take only items with large marginal ratios, similarly to MarginalRatioThresholding. The difference is that, once $f(S)$ exceeds a threshold λ, where $\lambda = \frac{1}{2}\alpha\,(1-b)\,v$, we store either the current set S or the latest added item as S'. This guarantees that $f(S') \geq \lambda$ and $c(S') \leq (1-b)K$, which means that S' has a large function value and sufficient room to add more elements. We call the process of constructing S' *branching*. We continue to add items with large marginal ratios to the current set S, and if we cannot add an item to S because it exceeds the capacity, we try to add the item to S'. Note that the set S', after branching, can have at most one extra item; but this extra item can be replaced if a better candidate comes along (See line 14–15).

Remark that the sequence of sets S in BranchingMRT is identical to that in MarginalRatioThresholding. Hence, we do not need to run MarginalRatioThresholding in parallel to this algorithm. We say that an item $e \in \mathrm{OPT}$ is *bad* if it is bad in the sense of MarginalRatioThresholding, i.e., it satisfies the condition in Definition 4. We have the following two lemmas.

▶ **Lemma 8.** *For a bad item e with $c(e) \leq bK$, let S_e be the set just before e arrives in Algorithm 4. Then $f(S_e) \geq \lambda$ holds. Thus branching has happened before e arrives.*

Proof. Sine e is a bad item, we have $c(S_e) > K - c(e) \geq (1-b)K$. Hence $f(S_e) \geq \alpha(1-b)v \geq \lambda$ by Lemma 3 (1). Since the value of f is non-decreasing during the process, it means that branching has happened before e arrives. ◀

▶ **Lemma 9.** *It holds that $f(S'_0) \geq \lambda$ and $c(S'_0) \leq (1-b)K$.*

Proof. We denote by S the set obtained right after leaving the while loop from Line 4. If $c(S) < (1-b)K$, then $f(S'_0) = f(S) \geq \lambda$. Otherwise, since $c(S) \geq (1-b)K$, we have $f(S) \geq \alpha(1-b)v \geq 2\lambda$ by Lemma 3 (1). Hence $f(S'_0) = f(\hat{e}) \geq \lambda$ since $f(S - \hat{e}) < \lambda$ and the submodularity. The second part holds since $c(\hat{e}) \leq K/2 \leq (1-b)K$ by $b \leq 1/2$. ◀

Let $\tilde{S}$ and $\tilde{S}'$ be the final two sets computed by BranchingMRT. Note that we can regard $\tilde{S}$ as the output of MarginalRatioThresholding and $\tilde{S}'$ as the final set obtained by adding at most one item to S'_0.

Observe that the number of bad items depends on the parameter α. As we will show in Section 3.2, by choosing a suitable α, if we have more than two bad items, then the size of $\tilde{S}$ is large enough, implying that $f(\tilde{S})$ is already good for approximation (due to Lemma 3 (1)). Therefore, in the following, we just concentrate on the case when we have at most two bad items.

▶ **Lemma 10.** *Let α be a number in $(0,1]$, and suppose that we have only one bad item o_{b}. If $v \leq f(\mathrm{OPT})$ and $b \in [c(o_{\mathrm{b}})/K, (1+\varepsilon)c(o_{\mathrm{b}})/K]$, then it holds that*

$$\max\{f(\tilde{S}), f(\tilde{S}')\} \geq \frac{1}{2}\left(1 - \alpha\frac{K - c(o_{\mathrm{b}})}{2K}\right)v - \frac{\varepsilon\alpha c(o_{\mathrm{b}})}{4K}v = \left(\frac{1}{2}\left(1 - \alpha\frac{K - c(o_{\mathrm{b}})}{2K}\right) - O(\varepsilon)\right)v.$$

Proof. Suppose not, that is, suppose that both of $f(\tilde{S})$ and $f(\tilde{S}')$ are smaller than βv, where $\beta = \frac{1}{2}(1 - \alpha\frac{K-c(o_{\mathrm{b}})}{2K}) - \frac{\alpha c(o_{\mathrm{b}})}{4K}\varepsilon$. We denote $O_{\mathrm{s}} = \mathrm{OPT} \setminus \{o_{\mathrm{b}}\}$.

Since the bad item o_{b} satisfies $c(o_{\mathrm{b}}) \leq bK$, it arrives after branching by Lemma 8. By Lemma 9, we have $c(S'_0 + o_{\mathrm{b}}) \leq K$. Since $f(\tilde{S}')$ is less than βv, we see that $f(S'_0 + o_{\mathrm{b}}) < \beta v$. Since $f(S'_0) \geq \lambda$,

$$f(\mathrm{OPT}) \leq f(o_{\mathrm{b}} \mid S'_0) + f(S'_0 \cup O_{\mathrm{s}}) < (\beta v - \lambda) + f(S'_0 \cup O_{\mathrm{s}}). \tag{3}$$

Since $S'_0 \subseteq \tilde{S}$, submodularity implies that

$$f(S'_0 \cup O_{\mathrm{s}}) \leq f(\tilde{S} \cup O_{\mathrm{s}}) \leq f(\tilde{S}) + \sum_{e \in O_{\mathrm{s}} \setminus \tilde{S}} f(e \mid \tilde{S}). \tag{4}$$

Since $f(\tilde{S}) < \beta v$ and no item in O_{s} is bad, (3) and (4) imply by Lemma 3 (2) that

$$\begin{aligned} v \leq f(\mathrm{OPT}) &< (\beta v - \lambda) + f(S'_0 \cup O_{\mathrm{s}}) < (\beta v - \lambda) + \beta v + \frac{\alpha c(O_{\mathrm{s}})}{K}v \\ &\leq 2\beta v - \frac{1}{2}\alpha(1-b)v + \alpha\left(1 - \frac{c(o_{\mathrm{b}})}{K}\right)v. \end{aligned}$$

Therefore, we have

$$\beta > \frac{1}{2}\left(1 + \alpha\frac{2c(o_{\mathrm{b}})/K - b - 1}{2}\right).$$

Since $b \leq (1+\varepsilon)c(o_{\mathrm{b}})/K$, we obtain

$$\beta > \frac{1}{2}\left(1 - \frac{(K - c(o_{\mathrm{b}}))\alpha}{2K}\right) - \frac{\alpha c(o_{\mathrm{b}})}{4K}\varepsilon,$$

which is a contradiction. This completes the proof. ◀

For the case when we have exactly two bad items, we obtain the following guarantee.

▶ **Lemma 11.** *Let α be a number in $(0, 1]$, and suppose that we have exactly two bad items o_{b} and o_{m} with $c(o_{\mathrm{b}}) \geq c(o_{\mathrm{m}})$. If $v \leq f(\mathrm{OPT})$ and $b \in [c(o_{\mathrm{b}})/K, (1+\varepsilon)c(o_{\mathrm{b}})/K]$, then it holds that*

$$\max\{f(\tilde{S}), f(\tilde{S}')\} \geq \frac{1}{3}\left(1 + \alpha\frac{c(o_{\mathrm{m}})}{K}\right)v - \frac{\alpha c(o_{\mathrm{b}})}{3K}\varepsilon v = \left(\frac{1}{3}\left(1 + \alpha\frac{c(o_{\mathrm{m}})}{K}\right) - O(\varepsilon)\right)v.$$

3.2 Algorithms with Guessing Large Items

We now use BranchingMRT to obtain a better approximation ratio. In the new algorithm, we guess the sizes of a few large items in an optimal solution OPT, and then use them to determine the parameter α.

We first remark that, when $|\mathrm{OPT}| \leq 2$, we can easily obtain a 1/2-approximate solution with a single pass. In fact, since $f(\mathrm{OPT}) \leq \sum_{i=1}^{\ell} f(o_i)$ where $\ell = |\mathrm{OPT}|$, at least one of o_i's satisfies $f(o_i) \geq f(\mathrm{OPT})/\ell$, and hence Singleton returns a 1/2-approximate solution when $\ell \leq 2$. Thus, in what follows, we may assume that $|\mathrm{OPT}| \geq 3$.

We start with the case that we have guessed the largest two sizes $c(o_1)$ and $c(o_2)$ in OPT.

▶ **Lemma 12.** *Let $\varepsilon \in (0, 1]$, and suppose that $v \leq f(\mathrm{OPT})$ and $c_i \leq c(o_i)/K \leq (1+\varepsilon)c_i$ for $i \in \{1, 2\}$. Then, $\tilde{S}' = \mathsf{BranchingMRT}(\varepsilon, \alpha, v, c_1, b)$ with $\alpha = 1/(2 - c_2)$ or $2/(5 - 4c_2 - c_1)$ and $b = \min\{(1+\varepsilon)c_1, 1/2\}$ satisfies*

$$f(\tilde{S}') \geq \left(\min\left\{\frac{1 - c_2}{2 - c_2}, \frac{2(1 - c_2)}{5 - 4c_2 - c_1}\right\} - O(\varepsilon)\right)v. \tag{5}$$

Proof. Let $\tilde{S} = \mathsf{MarginalRatioThresholding}(\alpha, v)$. Note that $f(\tilde{S}') \geq f(\tilde{S})$. If $\tilde{S}$ has size at least $(1 - (1+\varepsilon)c_2)K$, then Lemma 3 (1) implies that

$$f(\tilde{S}) \geq \alpha(1 - (1+\varepsilon)c_2)v = \alpha(1 - c_2)v - O(\varepsilon)v.$$

Otherwise, $c(\tilde{S}) < (1 - (1+\varepsilon)c_2)K$. In this case, we see that only the item o_1 can have size more than $(1+\varepsilon)c_2K$, and hence only o_1 can be a bad item. If o_1 is not a bad item, then we have no bad item, and hence Lemma 5 implies that

$$f(\tilde{S}) \geq (1 - \alpha)v.$$

If o_1 is bad, then Lemma 10 implies that

$$f(\tilde{S}') \geq \frac{1}{2}\left(1 - \alpha\frac{1 - c_1}{2}\right)v - O(\varepsilon)v.$$

Thus the approximation ratio is the minimum of the RHSes of the above three inequalities. This is maximized when $\alpha = 1/(2 - c_2)$ or $\alpha = 2/(5 - 4c_2 - c_1)$, and the maximum value is equal to the RHS of (5). ◀

Note that the approximation ratio achieved in Lemma 12 becomes $1/3 - O(\varepsilon)$ when, for example, $c_1 = c_2 = 1/2$. Hence, the above lemma does not show any improvement over Theorem 6 in the worst case. Thus, we next consider the case that we have guessed the largest three sizes $c(o_1)$, $c(o_2)$, and $c(o_3)$ in OPT. Using Lemma 11 in addition to Lemmas 3 (1), 5 and 10, we have the following guarantee.

▶ **Lemma 13.** *Let $\varepsilon \in (0,1]$, and suppose that $v \le f(\mathrm{OPT})$ and $c_i \le c(o_i)/K \le (1+\varepsilon)c_i$ for $i \in \{1,2,3\}$. Then the better output $\tilde{S}'$ of* BranchingMRT$(\varepsilon, \alpha, v, c_1, b_1)$ *and* BranchingMRT$(\varepsilon, \alpha, v, c_1, b_2)$ *with $\alpha = 1/(2-c_3)$ or $2/(c_2+3)$, $b_1 = \min\{(1+\varepsilon)c_1, 1/2\}$, and $b_2 = \min\{(1+\varepsilon)c_2, 1/2\}$ satisfies*

$$f(\tilde{S}') \ge \left(\min\left\{\frac{1-c_3}{2-c_3}, \frac{c_2+1}{c_2+3}\right\} - O(\varepsilon)\right) v.$$

Proof. Let $\tilde{S} =$ MarginalRatioThresholding(α, v). If $\tilde{S}$ has size at least $(1-(1+\varepsilon)c_3)K$, then we have by Lemma 3 (1)

$$f(\tilde{S}) \ge \alpha(1-(1+\varepsilon)c_3)v = \alpha(1-c_3)v - O(\varepsilon)v.$$

Otherwise, $c(\tilde{S}) < (1-(1+\varepsilon)c_3)K$. In this case, we see that only o_1 and o_2 can have size more than $(1+\varepsilon)c_3$, and hence only they can be bad items. If we have no bad item, it holds by Lemma 5 that

$$f(\tilde{S}) \ge (1-\alpha)v.$$

Suppose we have one bad item. If it is o_1 then Lemma 10 with b_1 implies

$$f(\tilde{S}') \ge \left(\frac{1}{2}\left(1-\alpha\frac{1-c_1}{2}\right) - O(\varepsilon)\right) v,$$

and, if it is o_2, we obtain by Lemma 10 with b_2

$$f(\tilde{S}') \ge \left(\frac{1}{2}\left(1-\alpha\frac{1-c_2}{2}\right) - O(\varepsilon)\right) v.$$

Moreover, if we have two bad items o_1 and o_2, then Lemma 11 implies

$$f(\tilde{S}') \ge \left(\frac{1}{3}(1+\alpha c_2) - O(\varepsilon)\right) v.$$

Therefore, the approximation ratio is the minimum of the RHSes in the above five inequalities, which is maximized to

$$\min\left\{\frac{1-c_3}{2-c_3}, \frac{c_2+1}{c_2+3}\right\} - O(\varepsilon),$$

when $\alpha = 1/(2-c_3)$ or $\alpha = 2/(c_2+3)$. ◀

We now see that we get an approximation ratio of $2/5 - O(\varepsilon)$ by combining the above two lemmas.

▶ **Theorem 14.** *Let $\varepsilon \in (0,1]$ and suppose that $v \le f(\mathrm{OPT}) \le (1+\varepsilon)v$ and $c_i \le c(o_i)/K \le (1+\varepsilon)c_i$ for $i \in \{1,2,3\}$. If $c(o_1) \le K/2$, then we can obtain a $(2/5 - O(\varepsilon))$-approximate solution with a single pass.*

Proof. We run the two algorithms with the optimal α shown in Lemmas 12 and 13 in parallel. Let $\tilde{S}$ be the output with the better function value. Then, we have $f(\tilde{S}) \ge \beta v$, where

$$\beta = \max\left\{\min\left\{\frac{1-c_2}{2-c_2}, \frac{2(1-c_2)}{5-4c_2-c_1}\right\}, \min\left\{\frac{1-c_3}{2-c_3}, \frac{c_2+1}{c_2+3}\right\}\right\} - O(\varepsilon).$$

We can confirm that the first term is at least $2/5$, and thus $\tilde{S}$ is a $(2/5 - O(\varepsilon))$-approximate solution. ◀

Algorithm 5

procedure DynamicBranchingMRT(ε)
  $\mathcal{V} := \{(1+\varepsilon)^i \mid i \in \mathbb{Z}_+\}$.
  For each $c_1, c_2, c_3 \in \mathcal{V}$ with $c_3 \le c_2 \le c_1 \le 1/2$ and each $b \in \{(1+\varepsilon)c_1, (1+\varepsilon)c_2, 1/2\}$, do the following with α defined based on Lemmas 12 and 13.
  For each $v \in \mathcal{V}$, set $S_v := \emptyset$.
  while item e is arriving **do**
    Delete e with $c(e) > (1+\varepsilon)c_1 K$.
    $m := \max\{m, f(e)\}$.
    $\mathcal{I} := \{v \in \mathcal{V} \mid m \le v \le Km/\alpha\}$.
    Delete S_v (along with $\hat{S}_v$ and S'_v if exists) such that $v \notin \mathcal{I}$.
    for $v \in \mathcal{V}$ **do**
      if $f(S_v) < \lambda$ **then**
        if $\frac{f(e|S_v)}{c(e)} \ge \frac{\alpha v - f(S_v)}{K - c(S_v)}$ and $c(S_v + e) \le K$ **then** $S_v := S_v + e$.
        if $f(S_v) \ge \lambda$ **then**
          if $c(S) \ge (1-b)K$ **then** $S' := \{e\}$ **else** $S' := S$.
          $\hat{S}_v := S'$.
      else
        if $\frac{f(e|S_v)}{c(e)} \ge \frac{\alpha v - f(S_v)}{K - c(S_v)}$ and $c(S_v + e) \le K$ **then** $S_v := S_v + e$.
        if $\frac{f(e|S_v)}{c(e)} \ge \frac{\alpha v - f(S_v)}{K - c(S_v)}$ and $c(S_v + e) > K$ **then**
          if $f(S'_v) < f(\hat{S}_v + e)$ **then** $S'_v := \hat{S}_v + e$.
  $S := S_v$ for $v \in \mathcal{I}$ that maximizes $f(S_v)$.
  $S' := S'_v$ for $v \in \mathcal{I}$ that maximizes $f(S'_v)$.
  return S or S' whichever has the larger function value.

To eliminate the assumption that we are given v, we can design a dynamic-update version of BranchingMRT by keeping the interval that contains the optimal value, similarly to Theorem 7. DynamicBranchingMRT, given in Algorithm 5, is a dynamic-update version of BranchingMRT. The proof for updating the interval $\mathcal{I}$ dynamically is the same as the proof of Theorem 7. The number of streams for guessing v is $O(\log(K)/\varepsilon)$. We also guess c_i for $i \in \{1,2,3\}$ from $\{(1+\varepsilon)^j \mid j \in \mathbb{Z}_+\}$. As $1 \le c(o_i) \le K/2$ for $i \in \{1,2,3\}$, the number of guessing for c_i is $O(\log(K)/\varepsilon)$. Hence, including v, there are $O((\log(K)/\varepsilon)^4)$ streams in parallel. To summarize, we obtain the following:

▶ **Theorem 15.** *Suppose that $c(o_1) \le K/2$. The algorithm that runs* DynamicBranchingMRT *and* Singleton *in parallel and takes the better output is a $(2/5 - \varepsilon)$-approximation streaming algorithm with a single pass for the problem* (1)*. The space complexity of the algorithm is $O(K(\log(K)/\varepsilon)^4)$.*

4 Single-Pass $(4/11 - \varepsilon)$-Approximation Algorithm

In this section, we consider the case that $c(o_1)$ is larger than $K/2$. For the purpose, we consider the problem of finding a set S of items that maximizes $f(S)$ subject to the constraint that the total size is at most pK, for a given number $p \ge 2$. We say that a set S of items is a (p, α)-*approximate solution* if $c(S) \le pK$ and $f(S) \ge \alpha f(\mathrm{OPT})$, where OPT is an optimal solution of the original instance.

▶ **Theorem 16.** *For a number $p \geq 2$, there is a $\left(p, \frac{2p}{2p+3} - \varepsilon\right)$-approximation streaming algorithm with a single pass for the problem* (1)*. In particular, when $p = 2$, it admits $(2, 4/7 - \varepsilon)$-approximation. The space complexity of the algorithm is $O(K(\log(K)/\varepsilon)^3)$.*

The basic framework of the algorithm is the same as in Section 3; we design a thresholding algorithm and a branching algorithm, where the parameters are different and the analysis is simpler.

Using Theorem 16, we can design a $(4/11 - \varepsilon)$-approximation streaming algorithm for an instance having a large item.

▶ **Theorem 17.** *For the problem* (1)*, there exists a $(4/11 - \varepsilon)$-approximation streaming algorithm with a single pass. The space complexity of the algorithm is $O(K(\log(K)/\varepsilon)^4)$.*

Proof. Let o_1 be an item in OPT with the maximum size. If $c(o_1) \leq K/2$, then Theorem 15 gives a $(2/5 - O(\varepsilon))$-approximate solution, and thus we may assume that $c(o_1) > K/2$. Note that there exists only one item whose size is more than $K/2$. Let β be the target approximation ratio which will be determined later. We may assume that $f(o_1) < \beta f(\mathrm{OPT})$, as otherwise Singleton (Algorithm 2) gives β-approximation. Then, we see $f(\mathrm{OPT} - o_1) > (1 - \beta)f(\mathrm{OPT})$ and $c(\mathrm{OPT} - o_1) < K/2$. Consider maximizing $f(S)$ subject to $c(S) \leq K/2$ in the set $\{e \in E \mid c(e) \leq K/2\}$. The optimal value is at least $f(\mathrm{OPT} - o_1) > (1 - \beta)f(\mathrm{OPT})$. We now apply Theorem 16 with $p = 2$ to this problem. Then, the output $\tilde{S}$ has size at most K, and moreover, we have $f(\tilde{S}) \geq \left(\frac{4}{7} - O(\varepsilon)\right)(1 - \beta)f(\mathrm{OPT})$. Thus, we obtain $\min\{\beta, (\frac{4}{7} - O(\varepsilon))(1 - \beta)\}$-approximation. This approximation ratio is maximized to $4/11$ when $\beta = 4/11$. ◀

5 Multiple-Pass Streaming Algorithm

In this section, we provide a multiple-pass streaming algorithm with approximation ratio $2/5 - \varepsilon$.

We first consider a generalization of the original problem. Let $E_R \subseteq E$ be a subset of the ground set E. For ease of presentation, we will call E_R the *red* items. Consider the problem defined below:

$$\text{maximize} \quad f(S) \quad \text{subject to} \quad c(S) \leq K, \quad |S \cap E_R| \leq 1. \tag{6}$$

In the following, we show that, given $\varepsilon \in (0, 1]$, an approximation v to $f(\mathrm{OPT})$ with $v \leq f(\mathrm{OPT}) \leq (1 + \varepsilon)v$, and an approximation θ to $f(o_{\mathrm{r}})$ for the unique item o_{r} in $\mathrm{OPT} \cap E_R$, we can choose $O(1)$ of the red items so that one of them $e \in E_R$ satisfies that $f(\mathrm{OPT} - o_{\mathrm{r}} + e) \geq (\Gamma(\theta) - O(\varepsilon))v$, where $\Gamma(\cdot)$ is a piecewise linear function lower-bounded by $2/3$. For technical reasons, we will choose θ to be one of the geometric series $(1 + \varepsilon)^i/2$ for $i \in \mathbb{Z}$.

▶ **Theorem 18.** *Suppose that we are given $\varepsilon \in (0, 1]$, $v \in \mathbb{R}_+$ with $v \leq f(\mathrm{OPT}) \leq (1 + \varepsilon)v$, and $\theta \in \mathbb{R}_+$ with the following property:*
1. *if $\theta \leq 1/2$, $\theta v/(1 + \varepsilon) \leq f(o_{\mathrm{r}}) \leq \theta v$,*
2. *if $\theta \geq 1/2$, $\theta v \leq f(o_{\mathrm{r}}) \leq (1 + \varepsilon)\theta v \leq v$.*

Then, there is a single-pass streaming algorithm that chooses a set S of red items in E_R with constant size such that (i) *for any item $e \in S$, $\theta v/(1 + \varepsilon) \leq f(o_{\mathrm{r}}) \leq \theta v$ when $\theta \leq 1/2$ and $\theta v \leq f(o_{\mathrm{r}}) \leq (1 + \varepsilon)\theta v \leq v$ when $\theta \geq 1/2$, and* (ii) *some item $e \in S$ satisfies that $f(\mathrm{OPT} - o_{\mathrm{r}} + e) \geq (\Gamma(\theta) - O(\varepsilon))v$, where $\Gamma(\theta)$ is defined as follows: when $\theta \in (0, 1/2)$,*

$$\Gamma(\theta) = \max\left\{\frac{t(t+3)}{(t+1)(t+2)} - \frac{t-1}{t+1}\theta \mid t \in \mathbb{Z}_+, t > \frac{1}{\theta} - 2\right\}, \tag{7}$$

when $\theta \in [1/2, 2/3)$, $\Gamma(\theta) = 2/3$, and when $\theta \in [2/3, 1]$, $\Gamma(\theta) = \theta$.

Algorithm 6

procedure MultiPassKnapsack($\varepsilon, v, \theta, c_1$) ▷ $\varepsilon \in (0,1]$, $v \in \mathbb{R}_+$, and $\theta, c_1 \in [0,1]$.
  Use the algorithm in Theorem 18 to choose a set S of items e with $c_1/(1+\varepsilon) \leq c(e)/K \leq c_1$ so that one of them $e \in S$ satisfies $f(\mathrm{OPT} - o_1 + e) \geq v(\Gamma(\theta) - O(\varepsilon))$.
  for each item $e \in S$ **do**
    Define a submodular function $g_e(\cdot) = f(\cdot \mid e)$.
    Apply the marginal-ratio thresholding algorithm (Lemma 21) with regard to function g_e, where $h = \frac{1-c_1}{1-(c_1/(1+\varepsilon))}$ and $K' = (1-(c_1/(1+\varepsilon))K$.
    Let the resultant set be S_e.
  return the solution $S_e \cup \{e\}$ with $\max_{e \in S} f(S_e + e)$.

We next show that when $c(o_1) \geq K/2$, we can use multiple passes to get a $(2/5-\varepsilon)$-approximation for the problem (1). Let $\mathrm{OPT} = \{o_1, o_2, \ldots, o_\ell\}$ be an optimal solution with $c(o_1) \geq c(o_2) \geq \cdots \geq c(o_\ell)$. Suppose that $c_1 \in \mathbb{R}_+$ satisfies $1/2 \leq c_1/(1+\varepsilon) \leq c(o_1)/K \leq c_1$.

We observe the following claims.

▶ **Claim 19.** *When $c(o_1) \geq K/2$, we may assume that $\frac{3}{10} f(\mathrm{OPT}) < f(o_1) < \frac{2}{5} f(\mathrm{OPT})$.*

▶ **Claim 20.** *We may assume that $c(o_1) \leq (1+\varepsilon)\frac{2}{3}K$.*

We use the first pass to estimate $f(\mathrm{OPT})$ as follows. For an error parameter $\varepsilon \in (0,1]$, perform the single-pass algorithm in Theorem 7 to get a $(1/3-\varepsilon)$-approximate solution $S \subseteq E$, which can be used to upper bound the value of $f(\mathrm{OPT})$, that is, $f(S) \leq f(\mathrm{OPT}) \leq (3+\varepsilon)f(S)$. We then find the geometric series to guess its exact value. Thus, we may assume that we are given the value v with $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$.

Below we show how to obtain a solution of value at least $(2/5 - O(\varepsilon))v$, using two more passes. Before we start, we introduce a slightly modified versions of the algorithms presented in Section 2; it will be used as a subroutine.

▶ **Lemma 21.** *Consider the problem (1) with the knapsack capacity K'. Let $h \in \mathbb{R}_+$, and suppose that Algorithms 1 and 2 are modified as follows:*

- *(At Line 4 in Algorithm 1) A new item e is added into the current set S only if $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{hK' - c(S)}$ and $c(S+e) \leq hK'$.*
- *(At Line 4 in Algorithm 2) A new item e is taken into account only if $c(e) \leq hK'$.*

Then, the best returned set $\tilde{S}$ of the two algorithms with $\alpha = \frac{2h}{h+2}$ satisfies that $c(\tilde{S}) \leq hK'$ and $f(\tilde{S}) \geq \frac{h}{h+2}v$. Moreover, we can obtain a $\left(\frac{h}{h+2} - O(\varepsilon)\right)$-approximate solution with the dynamic update technique.

Let all items $e \in E$ whose sizes $c(e)$ satisfy $c_1/(1+\varepsilon) \leq c(e)/K \leq c_1$ be the red items. By Theorem 18, we can select a set S of the red items so that one of them guarantees $f(\mathrm{OPT} - o_1 + e) \geq (\Gamma(\theta) - O(\varepsilon))v$, where θ satisfies the condition in Theorem 18. Note that any $e \in S$ satisfies $f(e) \geq \theta v/(1+\varepsilon)$. Also, by Claim 19, we see $\frac{3}{10}v < \theta < \frac{2}{5}(1+\varepsilon)v$.

In the next pass, for each $e \in S$, define a new monotone submodular function $g_e(\cdot) = f(\cdot \mid e)$ and apply the modified thresholding algorithm (Lemma 21) with $h = \frac{1-c_1}{1-(c_1/(1+\varepsilon))}$ and $K' = (1-(c_1/(1+\varepsilon))K$. Let S_e be the output of the modified thresholding algorithm. Then our algorithm returns the solution $S_e \cup \{e\}$ with $\max_{e \in S} f(S_e + e)$. The detail is given in Algorithm 6.

The returned solution has size at most K, since $c(S_e) \leq (1-c_1)K$ by Lemma 21. Moreover, it follows that the returned solution $\tilde{S}$ satisfies that $f(\tilde{S}) \geq (2/5 - O(\varepsilon))v$. The next theorem summarizes our results in this section.

▶ **Theorem 22.** *For $\varepsilon \in (0,1]$, suppose that $v \le f(\mathrm{OPT}) \le (1+\varepsilon)v$, $1/2 \le c_1/(1+\varepsilon) \le c(o_1)/K \le c_1$, and θ satisfies the condition in Theorem 18. After running* MultiPassKnapsack$(\varepsilon, v, \theta, c_1)$*, there exists an item $e \in S$ chosen in Line 2, which, along with S_e collected in Line 6, gives $f(S_e + e) \ge (2/5 - O(\varepsilon))v$.*

▶ **Theorem 23.** *Suppose that $c(o_1) > K/2$. There exists an algorithm that uses MultiPassKnapsack as a subroutine so that it returns $(2/5-\varepsilon)$-approximation with 3 passes for the problem (1). The space complexity of the algorithm is $O(K(\log(K)/\varepsilon)^2)$.*

References

1. Noga Alon, Iftah Gamzu, and Moshe Tennenholtz. Optimizing budget allocation among channels and influencers. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 381–388, 2012.
2. Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 671–680, 2014.
3. Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1497–1514, 2013.
4. Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.
5. Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015. doi:10.1007/s10107-015-0900-7.
6. T.-H. Hubert Chan, Zhiyi Huang, Shaofeng H.-C. Jiang, Ning Kang, and Zhihao Gavin Tang. Online submodular maximization with free disposal: Randomization beats for partition matroids online. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1204–1223, 2017.
7. Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 9134, pages 318–330, 2015.
8. Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014. doi:10.1137/110839655.
9. Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. *SIAM Journal on Computing*, 43(2):514–542, 2014.
10. M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions ii. *Mathematical Programming Study*, 8:73–87, 1978.
11. David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.
12. Andreas Krause, Ajit Paul Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.
13. Ariel Kulik, Hadas Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 545–554, 2013.

14 Jon Lee. *Maximum Entropy Sampling*, volume 3 of *Encyclopedia of Environmetrics*, pages 1229–1234. John Wiley & Sons, Ltd., 2006.

15 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010.

16 Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 912–920, 2010.

17 Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 510–520, 2011.

18 Tasuku Soma, Naonori Kakimura, Kazuhiro Inaba, and Ken-ichi Kawarabayashi. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 351–359, 2014.

19 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

20 Laurence Wolsey. Maximising real-valued submodular functions: primal and dual heuristics for location problems. *Mathematics of Operations Research*, 1982.

21 Qilian Yu, Easton Li Xu, and Shuguang Cui. Streaming algorithms for news and scientific literature recommendation: Submodular maximization with a d-knapsack constraint. *IEEE Global Conference on Signal and Information Processing*, 2016.