Multi-Pass Streaming Algorithms for Monotone Submodular Function Maximization

Chien-Chung Huang1 · Naonori Kakimura2

Accepted: 28 September 2021 / Published online: 9 November 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
We consider maximizing a monotone submodular function under a cardinality constraint or a knapsack constraint in the streaming setting. In particular, the elements arrive sequentially and at any point of time, the algorithm has access to only a small fraction of the data stored in primary memory. We propose the following streaming algorithms taking $O(\varepsilon^{-1})$ passes: (1) a $(1 - e^{-1} - \varepsilon)$-approximation algorithm for the cardinality-constrained problem, (2) a $(0.5 - \varepsilon)$-approximation algorithm for the knapsack-constrained problem. Both of our algorithms run deterministically in $O^*(n)$ time, using $O^*(K)$ space, where $n$ is the size of the ground set and $K$ is the size of the knapsack. Here the term $O^*$ hides a polynomial of $\log K$ and $\varepsilon^{-1}$. Our streaming algorithms can also be used as fast approximation algorithms. In particular, for the cardinality-constrained problem, our algorithm takes $O(n \varepsilon^{-1} \log(\varepsilon^{-1} \log K))$ time, improving on the algorithm of Badanidiyuru and Vondrák that takes $O(n \varepsilon^{-1} \log(\varepsilon^{-1} K))$ time.

Keywords Streaming algorithms · Approximation algorithms · Submodular function maximization
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 $S \subseteq T \subseteq E$ and $e \in E \setminus T$,

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

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. In addition to 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 [3, 30, 46], sensor location [31], text summarization [36, 37], and maximum entropy sampling [34].

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

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

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:

$$\text{maximize } f(S) \text{ subject to } |S| \leq K, \quad S \subseteq E. \quad (2)$$

The problem of maximizing a monotone submodular function under a knapsack or a cardinality constraint is classical and well-studied [22, 48]. 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., [5, 16, 23, 32, 47, 50]. Notice that for both problems, it is standard to assume that a value oracle of a function $f$ is given and the time complexity of the algorithms is measured based on the number of oracle calls.

In this work, we study the two problems with a focus on designing space and time efficient approximation algorithms. In particular, we assume the streaming setting: each item in the ground set $E$ arrives sequentially, and we can keep only a small number of the items in memory at any point. This setting renders most of the techniques in the literature ineffective, as they typically require random access to the data.

Our contribution. Our contributions are summarized as follows.

**Theorem 1 (Cardinality Constraint)** Let $n = |E|$. We design streaming $(1 - e^{-1} - \varepsilon)$-approximation algorithms for the problem (2) requiring either

1. $O(K)$ space, $O(\varepsilon^{-1} \log(\varepsilon^{-1} \log K))$ passes, and $O(n \varepsilon^{-1} \log(\varepsilon^{-1} \log K))$ running time, or
2. $O(K \varepsilon^{-1})$ space, $O(\varepsilon^{-1})$ passes, and $O(n \varepsilon^{-2})$ running time.
Table 1 The cardinality-constrained problem. The algorithms [5, 23, 41] are originally not for the streaming setting.

| Algorithm                                 | approx. ratio | # passes | space                        | running time               |
|-------------------------------------------|---------------|----------|------------------------------|----------------------------|
| Badanidiyuru et al. [4]                   | 0.5 − ε       | 1        | $O(K \varepsilon^{-1} \log K)$ | $O(n \varepsilon^{-1} \log K)$ |
| Kazemi et al. [29]                        | 0.5 − ε       | 1        | $O(K \varepsilon^{-1})$     | $O(n \varepsilon^{-1})$   |
| Ours                                      | $1 - \varepsilon^{-1} - \varepsilon$ | $O(\varepsilon^{-1})$ | $O(K \varepsilon^{-1})$ | $O(n \varepsilon^{-2})$   |
| Ours                                      | $1 - \varepsilon^{-1} - \varepsilon$ | $O(\varepsilon^{-1} \log(\varepsilon^{-1} \log K))$ | $O(K)$ | $O(n \varepsilon^{-1} \log(\varepsilon^{-1} \log K))$ |
| Badanidiyuru–Vondrák [5]                 | $1 - \varepsilon^{-1} - \varepsilon$ | $O(\varepsilon^{-1} \log(\varepsilon^{-1} K))$ | $O(K)$ | $O(n \varepsilon^{-1} \log(\varepsilon^{-1} K))$ |
| Mirzasoleiman et al. [41]                 | $1 - \varepsilon^{-1} - \varepsilon$ (in expectation) | $K$ | $O(K)$ | $O(n \log \varepsilon^{-1})$ |
| Greedy [23]                               | $1 - \varepsilon^{-1}$ | $K$ | $O(K)$ | $O(n \varepsilon^{-8} \log^2 K)$ |

To put our results in a better context, we list related work in Tables 1 and 2. For the cardinality-constrained problem, our first algorithm achieves the same ratio $1 - \varepsilon^{-1} - \varepsilon$ as Badanidiyuru and Vondrák [5], using the same space, while strictly improving on the running time and the number of passes. The second algorithm improves further the number of passes to $O(\varepsilon^{-1})$, which is independent of $K$ and $n$, but slightly loses out in the running time and the space requirement.\(^1\)

For the knapsack-constrained problem, our algorithm gives the best ratio so far using only small space (though at the cost of using more passes than [25, 27, 51]). In the non-streaming setting, Sviridenko [47] gave a $(1 - \varepsilon^{-1})$-approximation algorithm, which takes $O(Kn^4)$ time. Very recently, Ene and Nguyễn [17] gave $(1 - \varepsilon^{-1} - \varepsilon)$-approximation algorithm, which takes $O((1/\varepsilon)^O(1/\varepsilon^4)n \log n)$ time.

We remark that we assume in this paper that $K$ is smaller than $n$, which is a natural assumption in practical setting. When $K$ is larger than $n$, the linear dependence of the space complexity on $K$ in Theorem 2, as well as the others in Table 2, can be replaced with $\min\{K, n\}$. Thus the proposed algorithms have space complexity bounded by a polynomial in the input size.

**Theorem 2 (Knapsack Constraint)** Let $n = |E|$. We design streaming $(0.5 - \varepsilon)$-approximation algorithms for the problem (1) requiring $O(K \varepsilon^{-7} \log^2 K)$ space, $O(\varepsilon^{-1})$ passes, and $O(n \varepsilon^{-8} \log^2 K)$ running time.

Technical Overview. We first give an algorithm, called Simple, for the cardinality-constrained problem (2). This algorithm is later used as a subroutine for the knapsack-constrained problem (1). The algorithm Simple is based on thresholding strategy: in each pass, a certain threshold is set; items whose marginal value exceeds the threshold are added into the collection; others are just ignored. We show that, after $O(\varepsilon^{-1})$

\(^1\)Independently of our work, Norouzi-Fard et al. [43] proposed another algorithm with the same performance as the second one in Theorem 1.
Table 2 The knapsack-constrained problem. The algorithms [17, 47] are not for the streaming setting. See also [16, 32]

| approx. ratio | # passes | space | running time |
|---------------|----------|-------|--------------|
| Yu et al. [51] | \(1/3 - \varepsilon\) | 1 | \(O(K\varepsilon^{-1} \log K)\) | \(O(n\varepsilon^{-1} \log K)\) |
| Huang and Kakimura [25] | \(0.4 - \varepsilon\) | 1 | \(O(K\varepsilon^{-4} \log^4 K)\) | \(O(n\varepsilon^{-4} \log^4 K)\) |
| Ours | \(0.39 - \varepsilon\) | \(O(\varepsilon^{-1})\) | \(O(K\varepsilon^{-2} \log K)\) | \(O(n\varepsilon^{-1} \log K + n\varepsilon^{-3})\) |
| Ours | \(0.46 - \varepsilon\) | \(O(\varepsilon^{-1})\) | \(O(K\varepsilon^{-4} \log K)\) | \(O(n\varepsilon^{-5} \log K)\) |
| Ours | \(0.5 - \varepsilon\) | \(O(\varepsilon^{-1})\) | \(O(K\varepsilon^{-7} \log^2 K)\) | \(O(n\varepsilon^{-8} \log^2 K)\) |
| Ene and Nguyên [17] | \(1 - e^{-1} - \varepsilon\) | - | - | \(O\left((1/\varepsilon)^{O((1/\varepsilon)^3)} n \log n\right)\) |
| Sviridenko [47] | \(1 - e^{-1}\) | - | - | \(O(Kn^4)\) |

passes, we reach a \((1 - e^{-1} - \varepsilon)\)-approximation. To set the threshold, we need a prior estimate of the optimal value, similarly to a single-pass algorithm [4, 29], which we show can be found by a pre-processing step requiring either \(O(K\varepsilon^{-1})\) space and a single pass, or \(O(K)\) space and \(O(\varepsilon^{-1} \log(\varepsilon^{-1} \log K))\) passes. See Section 2 for the details.

We remark that the algorithm [5] can be implemented as multi-pass streaming algorithms (see also [40]), which takes \(O(\varepsilon^{-1} \log(\varepsilon^{-1} K))\) passes and \(O(K)\) space. Their algorithm also adopts the thresholding strategy. In [5, 40], the threshold is decreased in a conservative way (by a fixed factor of \(1 - \varepsilon\)) in each pass. In contrast, our algorithm adjusts the threshold dynamically, based on the \(f\)-value of the current collection and the optimal value. See also a remark at the end of Section 2.1.

For the knapsack-constrained problem (1), let us first point out the challenges in the streaming setting. The techniques achieving the best ratios in the literature are in [17, 47]. In [47], partial enumeration and density greedy are used. In the former, small sets (each of size at most 3) of items are guessed and for each guess, density greedy adds items based on the decreasing order of marginal ratio (i.e., the marginal value divided by the item size). To implement density greedy in the streaming setting, large number of passes would be required. In [17], partial enumeration is replaced by a more sophisticated multi-stage guessing strategies (where fractional items are added based on the technique of multilinear extension) and a “lazy” version of density greedy is used so as to keep down the time complexity. This version of density greedy nonetheless requires a priority queue to store the density of all items, thus requiring large space.

We present algorithms, in increasing order of sophistication, in Sections 3 to 5, that give \(0.39 - \varepsilon\), \(0.46 - \varepsilon\), and \(0.5 - \varepsilon\) approximations respectively. The first simpler algorithms are useful for illustrating the main ideas and also are used as subroutines for later, more involved algorithms. The first algorithm adapts the algorithm Simple for the cardinality-constrained case. We show in Section 3 that Simple still performs well if all items in the optimal solution (henceforth denoted by OPT) are small in

---

\[2\text{Recently, it is shown in [44] that it suffices to enumerate all the sets of at most two items.}\]
size (Lemma 6). Therefore, by ignoring the largest optimal item \( o_1 \), we can obtain a \((0.39 - \varepsilon)\)-approximate solution.

The difficulty arises when \( c(o_1) \) is large and the function value \( f(o_1) \) is too large to be ignored. To take care of such a large-size item, we first aim at finding a good item that approximates \( o_1 \) well. It is difficult to correctly identify \( o_1 \) among the items in the streaming setting, but we can nonetheless find a reasonable approximation \( e \) of it by a single pass [27]. Here the item \( e \) satisfies the following properties: (1) \( f(e) \) is large, (2) the function value of \( \text{OPT} - o_1 + e \) is large. After having this item \( e \), we apply \text{Simple} to pack items in \( \text{OPT} - o_1 \). Since the largest item size in \( \text{OPT} - o_1 \) is smaller, the performance of \text{Simple} is better than just applying \text{Simple} to the original instance. This solution, together with \( e \), yields a \((0.46 - \varepsilon)\)-approximation. See Section 4 for the details.

The above strategy would give a \((0.5 - \varepsilon)\)-approximation if \( f(o_1) \) is large enough. When \( f(o_1) \) is small, we need to generalize the above ideas further. In Section 5, we propose a two-phase algorithm. In Phase 1, an initial \textit{good set} \( Y \subseteq E \) is chosen (instead of a single good item); in Phase 2, the algorithm packs items in some subset \( \text{OPT}' \subseteq \text{OPT} \) using the remaining space. Ideally, the good set \( Y \) should satisfy the following properties: (1) \( f(Y) \) is large, (2) the marginal value of \( \text{OPT}' \) with respect to \( Y \) is large, and (3) the remaining space, \( K - c(Y) \), is sufficiently large to pack items in \( \text{OPT}' \). To find such a set \( Y \), we design two strategies, depending on the sizes, \( c(o_1), c(o_2) \) of the two largest items in \( \text{OPT} \). The detailed overview of our technique is described in Section 5.1.

**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 [10, 12, 13, 16, 21, 32, 35] and the references therein.

In the streaming setting, single-pass algorithms have been proposed for the problem with matroid constraints [11, 19], knapsack constraint [25, 27, 51] and \( p \)-system constraint [24], and without monotonicity [2, 14, 19, 42]. Recently, the inapproximability in the single-pass algorithms has been studied [20, 26, 29]. Feldman et al. [20] showed that any algorithm for the cardinality-constrained problem that achieves an approximation better than \( 1/2 \) requires \( \Omega(n/K^3) \) space, which means that the algorithms by [4, 29] are optimal. On the other hand, multi-pass streaming algorithms have not been well studied, except for [5, 11, 27]. Chakrabarti and Kale [11] gave an \( O(\varepsilon^{-3}) \)-pass streaming algorithms for a generalization of the maximum matching problem and the submodular maximization problem with cardinality constraint. We remark that the maximum matching problem is one of the central topic in the streaming setting, and multi-pass streaming algorithms have been developed [1, 28, 38]. For other graph problems, see e.g., [39]. 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 [8, 33, 45] and the adaptivity analysis [6, 7, 15, 18].

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 [40] gave a \((1 - e^{-1} - \varepsilon)\)-approximation algorithm in the multi-pass streaming setting. They use a sampling technique to estimate the value of \(f(OPT)\) and then collect items based on thresholds using \(O(\varepsilon^{-1})\) passes. Batani et al. [9] independently proposed a streaming algorithm with a sketching technique for the same problem.

**Subsequent Work.** For the general knapsack constraint, after our paper, Yaroslavtsev et al. [49] give another algorithm achieving the approximation ratio of \(0.5 - \varepsilon\), which requires \(O(1/\varepsilon)\) passes, \(O(K)\) space, and \(O(n(1/\varepsilon + \log K))\) running time, thus improving on our results in terms of space and time complexity.

**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 \(f : 2^E \rightarrow \mathbb{R}\), 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 \(f(e | S) = f(S + e) - f(S)\).

## 2 Cardinality Constraint

### 2.1 Simple Algorithm with Approximated Optimal Value

In this section, we introduce a procedure \textbf{Simple} (see Algorithm 1). This procedure can be used to give a \((1 - e^{-1} - \varepsilon)\)-approximation with the cardinality constraint; moreover, it will be adapted for the knapsack-constrained problem in Section 3.

**Algorithm 1**

1: \textbf{procedure} Simple (\(\mathcal{I} = (f, K, E); v, W\)) \hspace{1cm} \triangleright v \leq f(OPT) and \(W \geq K\)
2: \hspace{1cm} \(S := \emptyset\).
3: \hspace{1cm} \textbf{repeat}
4: \hspace{2cm} \(S_0 := S\) and \(\alpha := \frac{(1-\varepsilon)v - f(S_0)}{W}\).
5: \hspace{2cm} \textbf{for each} \(e \in E\) \textbf{do}
6: \hspace{3.5cm} \textbf{if} \(f(e | S) \geq \alpha\) and \(|S| < K\) \textbf{then} \(S := S + e\).
7: \hspace{2cm} \textbf{end for}
8: \hspace{1cm} \textbf{until} \(|S| = K\). \hspace{1cm} \triangleright T\) is used just for analysis
9: \hspace{1cm} \textbf{return} \(S\).

The input of \textbf{Simple} consists of

1. An instance \(\mathcal{I} = (f, K, E)\) for the problem (2).
2. An approximated value \(v\) of \(f(OPT)\), where OPT is an optimal solution of \(\mathcal{I}\). Specifically, we suppose \(v \leq f(OPT)\).
3. A positive integer $W$ that satisfies $W \geq K$. Usually, we suppose $W = K$ in this section. When adapted to the knapsack-constrained problem, $W$ will be an upper bound of the size of an optimal solution.

The output of Simple is a set $S$ that satisfies $f(S) \geq \beta v$ for some constant $\beta$ that will be determined later. If $f(\text{OPT}) \leq (1 + \epsilon)v$ in addition, then the output turns out to be a $(\beta - \epsilon)$-approximation. We will describe how to find such $v$ satisfying that $v \leq f(\text{OPT}) \leq (1 + \epsilon)v$ in the next subsection.

The following observations hold for the algorithm Simple.

**Lemma 1** During the execution of Simple in each round (in Lines 3–8), the following hold:

1. The current set $S \subseteq E$ always satisfies $f(T | S_0) \geq \alpha |T|$, where $T = S \setminus S_0$.
2. If an item $e \in E$ fails the condition $f(e | S_e) < \alpha$ at Line 6, where $S_e$ is the set just before $e$ arrives, then the final set $S$ in the round satisfies $f(e | S) < \alpha$.

**Proof** (1) Every item $e \in T$ satisfies $f(e | S_e) \geq \alpha$, where $S_e$ is the set just before $e$ arrives. Hence $f(T | S_0) = \sum_{e \in T} f(e | S_e) \geq \alpha |T|$. (2) follows from the definition of submodularity. \hfill $\Box$

Moreover, we can bound $f(S)$ from below using the size of $S$.

**Lemma 2** At the end of each round (in Lines 3–8), we have

$$f(S) \geq \left(1 - e^{-\frac{|S|}{W}} - 2\epsilon\right)v.$$  

**Proof** We prove the statement by induction on the number of rounds. Let $S$ be a set at the end of some round. Furthermore, let $S_0$ and $T$ be the corresponding two sets in the round; thus $S = S_0 \cup T$. By the induction hypothesis, we have

$$f(S_0) \geq \left(1 - e^{-\frac{|S_0|}{W}} - 2\epsilon\right)v. \quad (3)$$

Note that $S_0 = \emptyset$ in the first round, and hence the first round also satisfies the above inequality.

Due to Lemma 1(1), it holds that $f(S) = f(S_0) + f(T | S_0) \geq f(S_0) + \alpha |T|$, where $\alpha = \frac{(1 - \epsilon)v - f(S_0)}{|T|}$. Hence it holds that

$$f(S) \geq f(S_0) + \frac{(1 - \epsilon)v - f(S_0)}{W} |T| = f(S_0) \left(1 - \frac{|T|}{W}\right) + (1 - \epsilon)\frac{|T|}{W}v$$

$$\geq \left(1 - e^{-\frac{|S_0|}{W}} - 2\epsilon\right)\left(1 - \frac{|T|}{W}\right)v + \frac{|T|}{W}v - |T| \epsilon v$$

$$= \left(1 - \left(1 - \frac{|T|}{W}\right)e^{-\frac{|S_0|}{W}} - \left(2 - \frac{|T|}{W}\right)\epsilon\right)v$$

$$\geq \left(1 - \left(1 - \frac{|T|}{W}\right)e^{-\frac{|S_0|}{W}} - 2\epsilon\right)v.$$
where the second inequality uses (3). Since 
\[1 - \frac{|T|}{W} \leq e^{-\frac{|T|}{W}},\]
we have
\[f(S) \geq \left(1 - e^{-\frac{|S_0|+|T|}{W}} - 2\epsilon\right)v = \left(1 - e^{-\frac{|S|}{W}} - 2\epsilon\right)v,
\]
which proves the lemma.

The next lemma says that the function value increases by at least \(\epsilon f(OPT)\) in each round. This implies that the algorithm terminates in \(O(\epsilon^{-1})\) rounds.

**Lemma 3** Suppose that we run Simple\((I; v, W)\) with \(v \leq f(OPT)\) and \(W \geq K\). At the end of each round, if the final set \(S = S_0 \cup T\) (at Line 7) satisfies \(|S| < K\), then \(f(S) - f(S_0) \geq \epsilon f(OPT)\).

**Proof** Suppose that the final set \(S = S_0 \cup T\) satisfies \(|S| < K\). This means that, in the last round, each item \(e\) in \(OPT \setminus S\) is discarded because the marginal return is not large, which implies that \(f(e | S) < \alpha\) by Lemma 1(2). It holds from submodularity and monotonicity that
\[f(OPT) \leq f(OPT \cup S) \leq f(S) + \sum_{e \in OPT \setminus S} f(e | S).\]

Since \(f(e | S) < \alpha\) for any \(e \in OPT \setminus S\) and \(|OPT \setminus S| \leq K \leq W\), we have
\[f(OPT) \leq f(S) + \alpha W \leq f(S) + (1 - \epsilon)v - f(S_0),\]
since \(\alpha = \frac{(1-\epsilon)v-f(S_0)}{W}\). Since \(v \leq f(OPT)\), this proves the lemma.

From Lemmas 2 and 3 when setting \(W = K\), we have the following.

**Theorem 3** Let \(I = (f, K, E)\) be an instance of the cardinality-constrained problem (2). Suppose that \(v \leq f(OPT) \leq (1 + \epsilon)v\). Then Simple\((I; v, K)\) can compute a \((1 - \epsilon^{-1} - O(\epsilon))\)-approximate solution in \(O(\epsilon^{-1})\) passes and \(O(K)\) space. The total running time is \(O(\epsilon^{-1}n)\).

**Proof** While \(|S| < K\), the \(f\)-value is increased by at least \(\epsilon f(OPT)\) in each round by Lemma 3. Hence, after \(p\) rounds, the current set \(S\) satisfies that \(f(S) \geq pf(OPT)\). Since \(f(S) \leq f(OPT)\), the number of rounds is at most \(\epsilon^{-1} + 1\). As each round takes \(O(n)\) time, the total running time is \(O(\epsilon^{-1}n)\). Since we only store a set \(S\), the space required is clearly \(O(K)\).

The algorithm terminates when \(|S| = K\). From Lemma 2 and the fact that \(f(OPT) \leq (1 + \epsilon)v\), we have
\[f(S) \geq \left(1 - \epsilon^{-1} - 2\epsilon\right)v \geq \left(1 - \epsilon^{-1} - O(\epsilon)\right)f(OPT).
\]

We conclude this section with comparing to the algorithm by Badanidiyuru and Vondrák [5]. In their algorithm, a threshold is initially set to be \(m = \max_{e \in E} f(e)\), and in each pass, the threshold is reduced by a factor of \(1 - \epsilon\). This is repeated until the threshold becomes \(em/K\). Hence it takes \(O(\epsilon^{-1} \log(\epsilon^{-1}K))\) passes. Following the proof in [5], we see that the approximation ratio at the end of each pass is the same.
as in Lemma 2. In Section 3, we will adapt our algorithm Simple for the knapsack constrained problem, and use it as a subroutine in Sections 4 and 5. We note that their algorithm can also be used as a subroutine for our approach, but it will lose the factor of $O(\log(\varepsilon^{-1}K))$ passes.

### 2.2 Algorithm with Guessing the Optimal Value

We first note that $m \leq f(\text{OPT}) \leq mK$, where $m = \max_{e \in E} f(e)$. Hence, if we prepare $V = \{(1 + \varepsilon)^i m \mid (1 + \varepsilon)^i \leq K, i = 0, 1, \ldots\}$, then we can guess $v$ such that $v \leq f(\text{OPT}) \leq (1 + \varepsilon)v$. As the size of $V$ is $O(\varepsilon^{-1} \log K)$, if we run Simple for each element in $V$ in parallel, we need $O(K \varepsilon^{-1} \log K)$ space and $O(\varepsilon^{-1})$ passes in the streaming setting. This, however, will take $O(n \varepsilon^{-2} \log K)$ running time. We remark that, using a $(0.5 - \varepsilon)$-approximate solution $X$ by a single-pass streaming algorithm [29], we can guess $v$ from the range between $f(X)$ and $(2 + \varepsilon)f(X)$, which leads to $O(K \varepsilon^{-1})$ space and $O(n \varepsilon^{-2})$ time, taking $O(\varepsilon^{-1})$ passes. This proves the second part in Theorem 1.

Below we explain how to reduce the running time to $O(\varepsilon^{-1} n \log(\varepsilon^{-1} \log K))$ using the binary search.

**Theorem 4** For the cardinality-constrained problem (2), we can find a $(1 - \varepsilon^{-1} - \varepsilon)$-approximate solution in $O(\varepsilon^{-1} \log(\varepsilon^{-1} \log K))$ passes and $O(K)$ space, running in $O(n \varepsilon^{-1} \log(\varepsilon^{-1} \log K))$ time.

**Proof** We here describe an algorithm using Simple with slight modification. Let $p$ be the minimum integer that satisfies $(1 + \varepsilon)^p \geq K$. It follows that $p = O(\varepsilon^{-1} \log K)$.

We set $s_0 = 1$ and $t_0 = p$ in the beginning. It follows that $m(1 + \varepsilon)^{s_0} \leq f(\text{OPT}) \leq m(1 + \varepsilon)^{t_0}$. Suppose that $m(1 + \varepsilon)^{s_i} \leq f(\text{OPT}) \leq m(1 + \varepsilon)^{t_i}$ for some integer $i \geq 0$. Set $u = [(s_i + t_i)/2]$, and take the middle $v' = m(1 + \varepsilon)^u$. Perform Simple($\mathcal{I}'; v'$, $K$) with $W = K$, but we stop the repetition in $\varepsilon^{-1} + 1$ rounds. We will show below that either the output $S$ of Simple($\mathcal{I}'; v'$, $K$) satisfies that $f(S) \geq (1 - \varepsilon^{-1} - O(\varepsilon))f(\text{OPT})$, or we can restrict the range that $f(\text{OPT})$ exists by half.

First suppose that the output $S$ is of size $K$. Then, if $v' \geq f(\text{OPT})$, we have $f(S) \geq (1 - \varepsilon^{-1} - O(\varepsilon))v' \geq (1 - \varepsilon^{-1} - O(\varepsilon))f(\text{OPT})$ by Lemma 2. Hence we may assume that $v' \leq f(\text{OPT}) \leq m(1 + \varepsilon)^{t_i}$. So we set $s_{i+1} = u$ and $t_{i+1} = t_i$, and we repeat the process.

Next suppose that the output $S$ is of size $< K$. It follows from Lemma 3 that, if $f(\text{OPT}) \geq v'$, it holds that $f(S) > p \varepsilon f(\text{OPT})$ after $p$ rounds. Hence, after $\varepsilon^{-1} + 1$ rounds, we have $f(S) > f(\text{OPT})$, a contradiction. Thus we are sure that $f(\text{OPT}) < v'$. So we see that $m(1 + \varepsilon)^{t_i} \leq f(\text{OPT}) \leq v'$. We set $s_{i+1} = s_i$ and $t_{i+1} = u$, and we repeat the process.

We repeat the above binary search until the interval is 1. As $t_0/s_0 = p$, the number of iterations is $O(\log p) = O(\log(\varepsilon^{-1} \log K))$. Since each iteration takes $O(\varepsilon^{-1})$ passes, it takes $O(\varepsilon^{-1} \log(\varepsilon^{-1} \log K))$ passes in total. The running time is
\(O(n\varepsilon^{-1}\log(\varepsilon^{-1}\log K))\). Notice that there is no need to store the solutions obtained in each iteration, rather, just the function values and the corresponding indices \(u_i\) are enough to find out the best solution. Therefore, just \(O\left(K + \log\left(\varepsilon^{-1}\log K\right)\right) = O(K)\) space suffices.

3 Simple Algorithm for the Knapsack-Constrained Problem

In the rest of the paper, let \(I = (f, c, K, E)\) be an input instance of the problem (1). Let \(\text{OPT} = \{o_1, \ldots, o_\ell\}\) denote an optimal solution with \(c(o_1) \geq c(o_2) \geq \cdots \geq c(o_\ell)\). We denote \(c_i = c(o_i)/K\) for \(i = 1, 2, \ldots, \ell\). We remark that \(c_1 + c_2 \leq 1\), since \(\text{OPT}\) is feasible.

Similarly to Section 2, we suppose that we know in advance the approximate value \(v\) of \(f(\text{OPT})\), i.e., \(v \leq f(\text{OPT}) \leq (1 + \varepsilon)v\). The value \(v\) can be found with a single-pass streaming algorithm with constant ratio (e.g., [51]) in \(O(n\varepsilon^{-1}\log K)\) time and \(O(K\varepsilon^{-1}\log K)\) space. Specifically, letting \(X\) be the output of a single-pass \(\alpha\)-approximation algorithm, we know that the optimal value is between \(f(X)\) and \(f(X)/\alpha\). We can guess \(v\) by a geometric series \(\{(1 + \varepsilon)^i \mid i \in \mathbb{Z}\}\) in this range, and then the number of guesses is \(O(\varepsilon^{-1})\). Therefore, if we design an \(O(\varepsilon^{-1})\)-pass algorithm running in \(O(T_1)\) time and \(O(T_2)\) space provided the approximate value \(v\), then the total running time is \(O(n\varepsilon^{-1}\log K + \varepsilon^{-1}T_1)\) and the space required is \(O(\max\{K\varepsilon^{-1}\log K, \varepsilon^{-1}T_2\})\).

3.1 Simple Algorithm

We first claim that the algorithm \text{Simple} in Section 2 can be adapted for the knapsack-constrained problem (1) as below (Algorithm 2). Let \(W\) satisfy \(W \geq c(\text{OPT})\). In each round, we pick an item \(e\) when the marginal return per unit weight exceeds the threshold \(\alpha\), that is, when \(f(e \mid S) \geq \alpha c(e)\) and \(c(S + e) \leq K\), where \(S\) is the current set. Here, we set \(\alpha = \frac{(1-\varepsilon)v - f(S_0)}{W}\), where \(S_0\) is the initial set in the current round. We stop the repetition when \(f(S) - f(S_0) < \varepsilon v\). Clearly, the algorithm terminates.

| Algorithm 2 |
|-------------|
| 1: procedure Simple(\(I = (f, c, K, E)\); \(v, W\)) \(\triangleright v \leq f(\text{OPT})\) and \(W \geq c(\text{OPT})\) |
| 2: \(S := \emptyset\). |
| 3: repeat |
| 4: \(S_0 := S\) and \(\alpha := \frac{(1-\varepsilon)v - f(S_0)}{W}\). |
| 5: for each \(e \in E\) do |
| 6: \(\text{if } f(e \mid S) \geq \alpha c(e)\) and \(c(S + e) \leq K\) then \(S := S + e\). |
| 7: \(T := S \setminus S_0\). |
| 8: until \(f(S) - f(S_0) < \varepsilon v\) |
| 9: return \(S\). |

In a similar way to Lemmas 1 and 2, we have the following observations. We omit the proof.
Lemma 4 During the execution of \texttt{Simple} in each round (in Lines 3–8), the following hold:

1. The current set $S \subseteq E$ always satisfies $f(T \mid S_0) \geq \alpha c(T)$, where $T = S \setminus S_0$.
2. If an item $e \in E$ fails the condition $f(e \mid S_e) < \alpha c(e)$ at Line 6, where $S_e$ is the set just before $e$ arrives, then the final set $S$ in the round satisfies $f(e \mid S) < \alpha c(e)$.
3. At the end of each round, we have $f(S) \geq \left(1 - e^{\frac{c(S)}{W}} - 2\epsilon\right)v$.

Furthermore, similarly to the proof of Lemma 3, we see that the output has size more than $K - c(o_1)$.

Lemma 5 Suppose that we run \texttt{Simple}$(\mathcal{I}; v, W)$ with $v \leq f(OPT)$ and $W \geq c(OPT)$. At the end of the algorithm, it holds that $c(S) > K - c(o_1)$.

Thus, we obtain the following approximation ratio, that depends on the largest size of items in OPT.

Lemma 6 Let $\mathcal{I} = (f, c, K, E)$ be an instance of the problem (1). Suppose that $v \leq f(OPT) \leq O(1)v$ and $W \geq c(OPT)$. The algorithm \texttt{Simple}$(\mathcal{I}; v, W)$ can find in $O(\varepsilon^{-1})$ passes and $O(K)$ space a set $S$ such that $K - c(o_1) < c(S) \leq K$ and

$$f(S) \geq \left(1 - e^{-\frac{c(S)}{W}} - O(\varepsilon)\right)v \geq \left(1 - e^{-\frac{K-c(o_1)}{W}} - O(\varepsilon)\right)v.$$  \hfill (4)

The total running time is $O(\varepsilon^{-1}n)$.

Proof Let $S$ be the final set of \texttt{Simple}$(\mathcal{I}; v, K)$. By Lemma 5, the final set $S$ satisfies that $c(S) > K - c(o_1)$. Hence (4) follows from Lemma 4(3). The number of passes is $O(\varepsilon^{-1})$, as each round increases the $f$-value by $\varepsilon v$ and $f(OPT) \leq O(1)v$. Hence the running time is $O(\varepsilon^{-1}n)$, and the space required is clearly $O(K)$. \hfill $\square$

Lemma 6 gives us a good ratio when $c(o_1)$ is small (see Corollary 4 in Section 5.1). However, the ratio worsens when $c(o_1)$ becomes larger. In the next subsection, we show that \texttt{Simple} can be used to obtain a $(0.39 - \varepsilon)$-approximation by ignoring large-size items.

3.2 0.39-Approximation: Ignoring Large Items

Let us remark that \texttt{Simple} would work for finding a set $S$ that approximates any subset $X$. More precisely, given an instance $\mathcal{I} = (f, c, K, E)$ of the problem (1), consider finding a feasible set to $\mathcal{I}$ that approximates

(*) a subset $X \subseteq E$ such that $v \leq f(X) \leq O(1)v$ and $W \geq c(X)$.

This means that $v$ and $W$ are approximated values of $f(X)$ and $c(X)$, respectively. Let $X = \{x_1, \ldots, x_\ell\}$ with $f(x_1) \geq \cdots \geq f(x_\ell)$. Note that $X$ is not necessarily feasible to $\mathcal{I}$, i.e., $c(X)$ (and thus $W$) may be larger than $K$, but we assume that
Thus the first part of the lemma holds.

**Corollary 1** Suppose that we are given an instance $\mathcal{I} = (f, c, K, E)$ for the problem (1) and $v, W$ satisfying the above condition (*) for some subset $X \subseteq E$. Then $\text{Simple}(\mathcal{I}; v, W)$ can find a set $S$ in $O(\varepsilon^{-1})$ passes and $O(K)$ space such that $K - c(x_i) < c(S) \leq K$ and

$$f(S) \geq \left(1 - e^{-\frac{c(S)}{W}} - O(\varepsilon)\right) v \geq \left(1 - e^{-\frac{K - c(x_i)}{W}} - O(\varepsilon)\right) v.$$  

The total running time is $O(\varepsilon^{-1}n)$.

In particular, Corollary 1 can be applied to approximate $\text{OPT} - o_1$, with estimates of $c(o_1)$ and $f(o_1)$.

**Corollary 2** Suppose that we are given an instance $\mathcal{I} = (f, c, K, E)$ for the problem (1) such that $v \leq f(\text{OPT}) \leq O(1)v$ and $W \geq c(\text{OPT})$. We further suppose that we are given $c_1$ with $c_1K \leq c(o_1) \leq (1 + \varepsilon)c_1K$ and $\tau$ with $f(o_1) \leq \tau v$. Then we can find a set $S$ in $O(\varepsilon^{-1})$ passes and $O(K)$ space such that $K - c(o_2) < c(S) \leq K$ and

$$f(S) \geq (1 - \tau) \left(1 - e^{-\frac{K - c(o_2)}{W - \xi_1K}} - O(\varepsilon)\right) v.$$  

In particular, when $W = K$, we have $f(S) \geq (1 - \tau) \left(1 - e^{-\frac{1 - c_2}{1 - \xi_1}} - O(\varepsilon)\right) v \geq (1 - \tau) \left(1 - e^{-1} - O(\varepsilon)\right) v$.

**Proof** We may assume that $f(o_1) \leq 0.5v$, as otherwise by taking a singleton $e$ with maximum return $f(e)$, we have $f(e) \geq 0.5v$, implying that $S = \{e\}$ satisfies the inequality as $\tau \geq 0.5$. Hence we may assume that $\tau \leq 0.5$ by replacing $\tau$ with $\min\{\tau, 0.5\}$ if necessary. Moreover, it holds that $c(\text{OPT} - o_1) \leq W - \xi_1K$, $f(\text{OPT} - o_1) \geq f(\text{OPT}) - f(o_1) \geq (1 - \tau)v$, and $f(\text{OPT} - o_1) \leq v \leq 2(1 - \tau)v$. Using the fact, we perform $\text{Simple}(\mathcal{I}; (1 - \tau)v, W - \xi_1K)$ to approximate $\text{OPT} - o_1$. Since the largest-size item in $\text{OPT} - o_1$ is $o_2$, by Corollary 1, we can find a set $S$ such that $K - c(o_2) < c(S) \leq K$ and

$$f(S) \geq (1 - \tau) \left(1 - e^{-\frac{K - c(o_2)}{W - \xi_1K}} - O(\varepsilon)\right) v.$$  

Thus the first part of the lemma holds.

Consider the case when $W = K$. Then the above bound is equal to $f(S) \geq (1 - \tau) \left(1 - e^{-\frac{1 - c_2}{1 - \xi_1}} - O(\varepsilon)\right) v$. We note that

$$\frac{1 - c_2}{1 - \xi_1} \geq 1 - \varepsilon.$$  

Indeed, the inequality clearly holds when $c_2 \leq \xi_1$. Consider the case when $c_2 > \xi_1$. Then, since $c_1 + c_2 \leq 1$ and $c_1 \geq \xi_1$, we have $\xi_1 \leq 0.5$. Hence, since $c_2 \leq c_1 \leq \xi_1 < 0.5$, the inequality still holds.
(1 + ϵ)ζ1, we obtain $\frac{1 - c_2}{1 - c_1} \geq 1 - \epsilon \frac{\epsilon_1}{\epsilon_1 - c_1} \geq 1 - \epsilon$, where the last inequality holds since $\epsilon_1 \leq 0.5$. Thus we have a desired inequality. \qed

The above corollary, together with Lemma 6, allows us to find a $(0.39 - \epsilon)$-approximate solution.

**Corollary 3** Suppose that we are given an instance $\mathcal{I} = (f, c, K, E)$ for the problem (1) with $v \leq f(OPT) \leq (1 + \epsilon)v$. Then we can find a $(0.39 - O(\epsilon))$-approximate solution in $O(\epsilon^{-1})$ passes and $O(\epsilon^{-1}K)$ space. The total running time is $O(\epsilon^{-2}n)$.

**Proof** First suppose that $c(o_1) \leq 0.505K$. Then Lemma 6 with $W = K$ implies that we can find a set $S_1$ such that

$$f(S_1) \geq \left(1 - e^{-\frac{K}{K} - (c(o_1)}} - O(\epsilon)\right)v \geq \left(1 - e^{-(1-0.505)} - O(\epsilon)\right)v \geq (0.39 - O(\epsilon))v.$$ 

Thus we may suppose that $c(o_1) > 0.505K$. We guess $\epsilon_1$ with $\epsilon_1 K \leq c(o_1) \leq (1 + \epsilon)\epsilon_1 K$ by a geometric series of the interval $[0.505, 1.0]$, i.e., we find $\epsilon_1$ such that $0.505 \leq \epsilon_1 \leq c(o_1)/K \leq (1 + \epsilon)\epsilon_1 \leq 1$ using $O(\epsilon^{-1})$ more space. We may also suppose that $f(o_1) < 0.39v$, as otherwise we can just take a singleton with maximum return from $E$. By Corollary 2 with $W = K$ and $\tau = 0.39$, we can find a set $S_2$ such that

$$f(S_2) \geq 0.61 \left(1 - e^{-\frac{1-c_2}{1-c_1}} - O(\epsilon)\right)v.$$ 

Since $c_2 \leq 1 - \epsilon_1 \leq 0.495$, we have $\frac{1-c_2}{1-\epsilon_1} \geq 1.02$. Therefore, it holds that

$$f(S_2) \geq 0.61 \left(1 - e^{-1.02} - O(\epsilon)\right) \geq (0.39 - O(\epsilon))v.$$ 

This completes the proof. \qed

### 4 0.46-Approximation Algorithm

This section presents a $(0.46 - \epsilon)$-approximation algorithm for the knapsack-constrained problem. In our algorithm, we assume that we know in advance approximations of $c_1$ and $c_2$. That is, we are given $\epsilon_j, \bar{\epsilon}_j$ such that $\epsilon_j \leq c_i \leq \bar{\epsilon}_j$ and $\bar{\epsilon}_i \leq (1 + \epsilon)\epsilon_i$ for $i \in \{1, 2\}$. Define $E_i = \{e \in E \mid c(e) \in [\epsilon_i, K, \bar{\epsilon}_j K]\}$ for $i \in \{1, 2\}$. We call items in $E_1$ large items, and items in $E \setminus (E_1 \cup E_2)$ are small. Notice that we often distinguish the cases $c_1 \leq 0.5$ and $c_1 \geq 0.5$. In the former case, we assume that $\bar{\epsilon}_1 \leq 0.5$ while in the latter, $\epsilon_1 \geq 0.5$.

We first show that we may assume that $c_1 + c_2 \leq 1 - \epsilon$. This means that we may assume that $\bar{\epsilon}_1 + \bar{\epsilon}_2 \leq 1$. See Appendix A for the proof.

**Lemma 7** Suppose that we are given $v$ such that $v \leq f(OPT) \leq (1 + \epsilon)v$. If $c_1 + c_2 > 1 - \epsilon$, we can find a $(0.5 - O(\epsilon))$-approximate solution in $O(\epsilon^{-1}K)$ space using $O(\epsilon^{-1})$ passes. The total running time is $O(\epsilon n \epsilon^{-1})$. 

© Springer
As mentioned in Introduction, the main idea of our algorithm is to find a good item \( e \in E_1 \) that approximates \( o_1 \). After having this item \( e \), we define \( g(\cdot) = f(\cdot | e) \), and consider the problem:

\[
\text{maximize } g(S) \quad \text{subject to } c(S) \leq K - c(e), \quad S \subseteq E.
\] (5)

We then apply the algorithm Simple for (5) as in Section 3.2. Recall that, in Section 3.2, we run Simple to approximate \( \text{OPT} \) and \( \text{OPT} - o_1 \), respectively. In the case of (5), as we already have an approximation \( e \) of \( o_1 \), we approximate \( \text{OPT} - o_1 \) and \( \text{OPT} - o_1 - o_2 \). Since the largest item sizes in \( \text{OPT} - o_1 \) and \( \text{OPT} - o_1 - o_2 \) are smaller, the performances of Simple are better than just applying Simple to the original instance. These solutions to (5), together with the item \( e \), will give us well-approximate solutions for the original instance.

More specifically, we find an item \( e \in E_1 \) such that \( f(e) \approx f(o_1) \), \( f(\text{OPT} - o_1 | e) \geq p_1 v \) and \( f(\text{OPT} - o_1 - o_2 | e) \geq p_2 v \) for some \( p_1, p_2 \). Let \( S_1 \) and \( S_2 \) be the outputs of Simple for (5) to approximate \( \text{OPT} - o_1 \) and \( \text{OPT} - o_1 - o_2 \), respectively. Then both of \( S_1 + e \) and \( S_2 + e \) are feasible to the original instance, and moreover, if \( g(S_1) \geq \kappa_1 g(\text{OPT} - o_1) \) and \( g(S_2) \geq \kappa_2 g(\text{OPT} - o_1 - o_2) \), where \( \kappa_1 \) and \( \kappa_2 \) are determined by Corollary 1, then it holds that

\[
f(S_1 + e) \geq f(e) + \kappa_1 p_1 v \quad \text{and} \quad f(S_2 + e) \geq f(e) + \kappa_2 p_2 v.
\] (6)

To make both the RHSs large, we aim to find an item \( e \) from \( E_1 \) such that \( f(e) \approx f(o_1) \) and \( p_1, p_2 \) are large simultaneously. We propose two algorithms for finding such \( e \) in Section 4.1. The following subsections give the details.

### 4.1 Finding a Good Item

One of the important observation from submodularity is the following, which is useful for analysis when \( c_1 \leq 0.5 \). Here, \( \beta \) is a parameter that represents a target approximation ratio, which will be optimized later.

**Lemma 8** Let \( e_0 \in E \). Suppose that \( f(\text{OPT}) \geq v \). If \( f(e_0 + o_1) < \beta v \), then we have

\[
f(\text{OPT} - o_1 | e_0) \geq (1 - \beta) v.
\]

Moreover, if \( f(e_0 + o_2) < \beta v \) in addition, then we obtain

\[
f(\text{OPT} - o_1 - o_2 | e_0) \geq (1 - 2\beta) v + f(e_0).
\]

**Proof** By assumption, it holds that \( \beta v > f(e_0 + o_1) = f(e_0) + f(o_1 | e_0) \), implying

\[
f(\text{OPT} - o_1 | e_0) \geq f(\text{OPT} | e_0) - f(o_1 | e_0)
\]

\[
\geq (f(\text{OPT}) - f(e_0)) - (\beta v - f(e_0)) \geq (1 - \beta) v.
\]

Moreover, if \( f(e_0 + o_2) < \beta v \) in addition, then we have \( \beta v > f(e_0 + o_2) = f(e_0) + f(o_2 | e_0) \), implying

\[
f(\text{OPT} - o_1 - o_2 | e_0) \geq f(\text{OPT} - o_1 | e_0) - f(o_2 | e_0) \geq (1 - \beta) v - (\beta v - f(e_0)).
\]

Thus the statement holds. \( \square \)
When $c_1 \leq \bar{c}_1 \leq 0.5$, for any item $e_0 \in E_1$, we see that $e_0 + o_1$ is a feasible set. Hence, by checking whether $f(e_0 + e') \geq \beta v$ for some $e' \in E$ using a single pass, it holds that, either we have a feasible set $e_0 + e'$ such that $f(e_0 + e') \geq \beta v$, or we bound $f(OPT - o_1 | e_0)$ and $f(OPT - o_1 - o_2 | e_0)$ from below by the above lemma.

Another way to lower-bound $p_1$ and $p_2$ in (6) is to use the algorithm in [27] where we find a reasonable approximation of $o_1$ by a single pass. For the sake of convenience, we define a procedure $\text{PickNiceItem}$. This procedure $\text{PickNiceItem}$ takes an estimate $v$ of $f(OPT)$ along with the estimate of the size of $o_1$ and of its $f$-value. It then returns an item of similar size, which, together with $\text{OPT} - o_1$, guarantees $(2/3 - O(\varepsilon))v$. More precisely, we have the following theorem.

**Theorem 5** [27] Let $X \subseteq E$ such that $f(X) \geq v$. Furthermore, assume that there exists $x_1 \in X$ such that $cK \leq c(x_1) \leq \bar{c}K$ and $\tau v/(1 + \varepsilon) \leq f(x_1) \leq \tau v$. Then $\text{PickNiceItem}(v, (\bar{c}, \bar{c}), \tau)$, a single-pass streaming algorithm using $O(1)$ space, returns a set $Y$ of $O(1)$ items such that some item $e^*$ in $Y$ satisfies

$$f(X - x_1 + e^*) \geq \Gamma(\tau)v - O(\varepsilon)v,$$

where

$$\Gamma(t) = \begin{cases} 
2 - \frac{t}{10} & \text{if } t \geq 0.5 \\
\frac{t}{2} - \frac{t}{3} & \text{if } 0.5 \geq t \geq 0.4 \\
\frac{t}{10} - \frac{t}{2} & \text{if } 0.4 \geq t \geq 0.
\end{cases}$$

Moreover, for any item $e \in Y$, we have $\tau v/(1 + \varepsilon) \leq f(e) \leq \tau v$ and $cK \leq c(e) \leq \bar{c}K$.

Using the procedure $\text{PickNiceItem}$, we can find a good item $e$.

**Lemma 9** Let $Y := \text{PickNiceItem}(v, (\bar{c}_1, \bar{c}_1), \tau)$, where we assume $f(OPT) \geq v$ and $\tau v/(1 + \varepsilon) \leq f(o_1) \leq \tau v$. Then there exists $e \in Y$ such that $\tau v/(1 + \varepsilon) \leq f(e) \leq \tau v$ and

$$f(OPT - o_1 | e) \geq (\Gamma(\tau) - \tau)v - O(\varepsilon)v.$$ 

Moreover, if $f(e + o_2) < \beta v$ in addition, then

$$f(OPT - o_1 - o_2 | e) \geq (\Gamma(\tau) - \beta)v - O(\varepsilon)v.$$ 

**Proof** It follows from Theorem 5 that some $e \in Y$ satisfies that $f(OPT - o_1 + e) \geq \Gamma(\tau)v - O(\varepsilon)v$ and $f(e) \leq \tau v$, and hence

$$f(OPT - o_1 | e) = f(OPT - o_1 + e) - f(e) \geq (\Gamma(\tau) - \tau)v - O(\varepsilon)v.$$ 

Moreover, if $f(e + o_2) < \beta v$ in addition, then we have

$$\beta v > f(e + o_2) = f(e) + f(o_2 | e) \geq \frac{\tau}{1 + \varepsilon}v + f(o_2 | e) \geq \tau v + f(o_2 | e) - O(\varepsilon)v,$$

implying

$$f(OPT - o_1 - o_2 | e) \geq f(OPT - o_1 | e) - f(o_2 | e) \geq (\Gamma(\tau) - \tau)v - (\beta - \tau)v - O(\varepsilon)v.$$
Thus the statement holds. □

4.2 Algorithm: Taking a Good Large Item First

Suppose that we have \( e \in E_1 \) such that \( f(\text{OPT} - \alpha_1 \mid e) \geq p_1v \) and \( f(\text{OPT} - \alpha_1 - \alpha_2 \mid e) \geq p_2v \), knowing that such \( e \) can be found by Lemma 8 or 9. More precisely, when \( c_1 \geq 0.5 \), we find a set \( T \) by PickNicItem\((v, (c_1, \bar{c}_1), \tau)\), where \( \tau v/(1 + \varepsilon) \leq f(o_1) \leq \tau v \); when \( c_1 \leq 0.5 \), set \( T = \{ e \} \) for arbitrary \( e \in E_1 \). Then \( |T| = O(1) \) and some \( e \in T \) satisfies \( f(\text{OPT} - \alpha_1 \mid e) \geq p_1v \) and \( f(\text{OPT} - \alpha_1 - \alpha_2 \mid e) \geq p_2v \), where \( p_1 \) and \( p_2 \) are determined by Lemma 8 or 9.

Then, for each item \( e \in T \), consider the problem (5), and let \( T' \) be the corresponding instance. We apply Simple to the instance \( T' \) approximating \( \text{OPT} - \alpha_1 \) and \( \text{OPT} - \alpha_1 - \alpha_2 \), respectively. Here we set \( v_\ell = p_\ell v (\ell = 1, 2) \), \( W_1 = W - \bar{c}_1K \), and \( W_2 = W - \bar{c}_1K - \bar{c}_2K \), where \( c(\text{OPT}) \leq W \). It then follows that \( c(\text{OPT} - \alpha_1) \leq W_1 \) and \( c(\text{OPT} - \alpha_1 - \alpha_2) \leq W_2 \). Define \( S'_\ell = e + \text{Simple}(T'; v_\ell, W_\ell) \) for \( \ell = 1, 2 \). Also define \( S'_0 = e + e^* \), where \( e^* = \arg \max_{e' \in E : c(e') \leq K - c(e)} f(e + e') \). Finally, for \( \ell = 0, 1, 2 \), define \( \tilde{S}_\ell \) to be the set that achieves \( \max\{ f(S'_\ell) \mid e \in T \} \). We return the best one among \( \tilde{S}_\ell \)'s.

We call the algorithm described above LargeFirst\((T; v, W, \tau)\). We remark that we can solve the problem (5) for each item \( e \in T \) in parallel using the same \( O(\varepsilon^{-1}) \) passes. Since \( |T| = O(1) \), it takes \( O(K) \) space.

The following bounds follow from Corollary 1.

**Lemma 10** Suppose that \( v \leq f(\text{OPT}) \leq (1 + \varepsilon)v \) and \( c(\text{OPT}) \leq W \). We further suppose that \( c_\ell \leq c_\ell \leq \bar{c}_\ell \leq (1 + \varepsilon)c_\ell (\ell = 1, 2) \) and \( \tau v/(1 + \varepsilon) \leq f(\alpha_1) \leq \tau v \). Let \( c = c_1 + c_2 \leq \varepsilon/\delta \) for some constant \( \delta \), we have

\[
f(\tilde{S}_1) \geq \left( \tau + p_1 \left( 1 - e^{-\frac{K - \tau v}{W - c_\ell K}} \right) \right) - O(\varepsilon)v, \quad (7)
\]

\[
f(\tilde{S}_2) \geq \left( \tau + p_2 \left( 1 - e^{-\frac{K - \tau v}{W - c_\ell K}} \right) \right) - O(\varepsilon)v. \quad (8)
\]

In particular, if \( W = K \) and \( c_1 + c_2 \leq \varepsilon/\delta \) for some constant \( \delta \), we have

\[
f(\tilde{S}_1) \geq \left( \tau + p_1 \left( 1 - e^{-\frac{(1 - \delta)\mu}{\mu - 1}} \right) \right) - O(\varepsilon)v, \quad (9)
\]

\[
f(\tilde{S}_2) \geq \left( \tau + p_2 \left( 1 - e^{-\frac{(1 - \delta)\mu}{\mu - 1}} \right) \right) - O(\varepsilon)v, \quad (10)
\]

where \( \mu = \frac{1 - \bar{c}_1 - \bar{c}_2}{1 - \bar{c}_1} \). Springer
Proof We note that \( c(OPT - o_1) \leq W_1 = W - c_1 K \), and items in \( OPT - o_1 \) are of size at most \( c(o_2) \). By Corollary 1, Simple(\( I' ; p_1 v, W_1 \)) can find a set \( S \) such that
\[
g(S) \geq p_1 \left( 1 - e^{-\frac{K - c_1 K - c_2 K}{W - c_1 K}} - O(\varepsilon) \right) v
\]
as the capacity \( K - c(e) \geq K - \bar{c}_1 K \). Therefore, since \( f(e) \geq (\tau - O(\varepsilon))v \), the inequality (7) follows from (6). The inequality (8) holds in a similar way, noting that \( c(OPT - o_1 - o_2) \leq W - \bar{c}_1 K - c_2 K \), and items in \( OPT - o_1 - o_2 \) are of size at most \( c(o_3) \).

Suppose that \( W = K \). Then the above inequalities (7) and (8) can be transformed to
\[
f(\tilde{S}_1) \geq \left( \tau + p_1 \left( 1 - e^{-\frac{1 - \bar{c}_1 c_3}{W - \bar{c}_1 c_2}} \right) - O(\varepsilon) \right) v,
\]
and
\[
f(\tilde{S}_2) \geq \left( \tau + p_2 \left( 1 - e^{-\frac{1 - \bar{c}_1 c_3}{W - \bar{c}_1 c_2}} \right) - O(\varepsilon) \right) v.
\]
Since \( \bar{c}_\ell \leq (1 + \varepsilon)c_\ell \) for \( \ell = 1, 2 \), we have
\[
\lambda_1 := \frac{1 - \bar{c}_1}{1 - c_1} \geq 1 - \varepsilon \frac{c_3}{1 - c_1} \geq 1 - \delta + \varepsilon, \quad \text{and}
\]
\[
\lambda_2 := \frac{1 - \bar{c}_1 - \bar{c}_2}{1 - c_1 - c_2} \geq 1 - \varepsilon \frac{c_1 + c_2}{1 - c_1 - c_2} \geq 1 - \delta + \varepsilon,
\]
where the second inequalities of each follow because \( c_1 \leq c_1 + c_2 \leq 1 - \varepsilon/\delta \). Using \( \lambda_1 \), the exponent in (11) is equal to
\[
\frac{1 - \bar{c}_1 - c_3}{1 - c_1 - c_2} \geq (1 - \delta) \mu.
\]
Thus (9) holds. Moreover, since \( c_3 \leq 1 - c_1 - c_2 \), using \( \lambda_2 \), the exponent in (12) is equal to
\[
\frac{1 - \bar{c}_1 - c_3}{1 - c_1 - c_2} \geq \frac{1 - \bar{c}_1}{1 - c_1 - c_2} - 1 = \lambda_2 \frac{1 - \bar{c}_1}{1 - c_1 - c_2} - 1 \geq (1 - \delta) \frac{1 - \bar{c}_1}{1 - c_1 - c_2} - 1.
\]
Thus (10) holds.

\[
\square
\]

4.3 Analysis: 0.46-Approximation

We next analyze the approximation ratio of the algorithm. We consider two cases when \( c_1 \leq 0.5 \) and \( c_1 \geq 0.5 \) separately; we will show that LargeFirst, together with Simple, admits a \((0.46 - \varepsilon)\)-approximation when \( c_1 \leq 0.5 \) and a \((0.49 - \varepsilon)\)-approximation when \( c_1 \geq 0.5 \), respectively.

Lemma 11 Suppose that \( c_1 \leq 0.5 \) and \( c_1 + c_2 \leq 1 - \varepsilon/\delta \), where \( \delta = 0.01 \). We further suppose that \( c_\ell \leq c_\ell \leq \bar{c}_\ell \leq (1 + \varepsilon)c_\ell \) (\( \ell = 1, 2 \)), \( \bar{c}_1 \leq 0.5 \), and \( v \leq f(OPT) \leq (1 + \varepsilon)v \). Then Algorithm LargeFirst with \( \beta = 0.46 \), together with Simple, can find a \((0.46 - O(\varepsilon))\)-approximate solution in \( O(\varepsilon^{-1}) \) passes and \( O(\varepsilon^{-1} K) \) space. The total running time is \( O(\varepsilon^{-2} n) \).

\[
\square
\]
Proof First suppose that \(f(o_1) \leq 0.272v\). Then, by Corollary 2 with \(W = K\), we can find a set \(S\) such that

\[
    f(S) \geq 0.728 \left(1 - e^{-1} - O(\varepsilon)\right) v \geq (0.46 - O(\varepsilon))v.
\]

Thus we may suppose that \(f(o_1) \geq 0.272v\). We may also suppose that \(f(o_1) \leq 0.46v\), as otherwise taking a singleton with maximum return from \(E_1\) gives a 0.46-approximation. We guess \(\tau\) and \(\overline{\tau}\) with \(0.272v \leq \tau v \leq f(o_1) \leq \overline{\tau} v \leq 0.46v\) and \(\overline{\tau} \leq (1 + \varepsilon)\tau\) from the interval \([0.272, 0.46]\) by a geometric series, that uses \(O(\varepsilon^{-1})\) space.

Consider the final sets \(\tilde{S}_0, \tilde{S}_1, \text{ and } \tilde{S}_2\) of LargeFirst\((I; v, K, \tau)\). If \(f(\tilde{S}_0) \geq \beta v = 0.46v\), then we are done. By Lemmas 8 and 10, \(f(\tilde{S}_1)\), and \(f(\tilde{S}_2)\) are lower-bounded by the RHSs of the following two inequalities:

\[
    f(\tilde{S}_1) \geq \left(\overline{\tau} + (1 - \beta) \left(1 - e^{-(1-\delta)\mu}\right) - O(\varepsilon)\right) v, \quad (13)
\]

\[
    f(\tilde{S}_2) \geq \left(\overline{\tau} + (1 - 2\beta + \overline{\tau}) \left(1 - e^{-\left(\frac{1-\delta}{\mu}\right)^{-1}}\right) - O(\varepsilon)\right) v, \quad (14)
\]

where \(\mu = \frac{1 - \varepsilon}{1 - \varepsilon_1}, \overline{\tau} \geq 0.272\) and \(\beta = 0.46\). If \(\mu \geq 0.5\), then (13) implies that

\[
    f(\tilde{S}_1) \geq 0.272 + (1 - 0.46) \left(1 - e^{-\frac{1-\delta}{\mu}}\right) - O(\varepsilon) v \geq (0.46 - O(\varepsilon))v,
\]

when \(\delta = 0.01\). On the other hand, if \(\mu \leq 0.5\), then (14) implies that

\[
    f(\tilde{S}_2) \geq 0.272 + (1 - 2 \cdot 0.46 + 0.272) \left(1 - e^{-\left(\frac{1-\delta}{0.5\mu}\right)^{-1}}\right) - O(\varepsilon) v \\
    \geq (0.46 - O(\varepsilon))v,
\]

when \(\delta = 0.01\). Thus the statement holds. \(\square\)

Similarly, we have the following guarantee when \(c_1 \geq 0.5\).

Lemma 12 Suppose that \(c_1 \geq 0.5\) and \(c_1 + c_2 \leq 1 - \varepsilon/\delta\), where \(\delta = 0.01\). We further suppose that \(c_\ell \leq c_\ell \leq c_\ell \leq 1 + \varepsilon, c_\ell (\ell = 1, 2), c_1 \geq 0.5, \text{ and } v \leq f(\text{OPT}) \leq (1 + \varepsilon)v\). Then Algorithm LargeFirst with \(\beta = 0.49\), together with Simple, can find a \((0.49 - O(\varepsilon))\)-approximate solution in \(O(\varepsilon^{-1})\) passes and \(O(\varepsilon^{-1}K)\) space. The total running time is \(O(\varepsilon^{-2}n)\).

Proof First suppose that \(f(o_1) \leq 0.22v\). Then, by Corollary 2 with \(W = K\), we can find a set \(S\) such that

\[
    f(S) \geq 0.78 \left(1 - e^{-1} - O(\varepsilon)\right) v \geq (0.49 - O(\varepsilon))v.
\]

Thus we may suppose that \(f(o_1) \geq 0.22v\). We may also suppose that \(f(o_1) \leq 0.49v\), as otherwise taking a singleton with maximum return from \(E_1\) gives a 0.49-approximation. We guess \(\tau\) and \(\overline{\tau}\) with \(0.22v \leq \tau v \leq f(o_1) \leq \overline{\tau} v \leq 0.49v\) and \(\overline{\tau} \leq (1 + \varepsilon)\tau\) from the interval \([0.22, 0.49]\) by a geometric series using \(O(\varepsilon^{-1})\) space.
Let $\mu = \frac{1 - \tau}{1 - \frac{\tau}{2}}$. By Corollary 2 with $W = K$, we can find a set $\tilde{S}$ such that

$$f(\tilde{S}) \geq (1 - \tau) \left(1 - e^{-\frac{1 - \frac{\tau}{2}}{1 - \frac{\tau}{2}} - O(\varepsilon)}\right) v \geq (1 - \tau) \left(1 - e^{-(1-\delta)\mu - 1 - O(\varepsilon)}\right) v,$$

where we note that

$$\frac{1 - \bar{c}}{1 - \bar{c}_1} = \frac{1 - \bar{c}_1}{1 - \bar{c}_1} \left(\frac{1 - \bar{c}_1 - \bar{c}_2}{1 - \bar{c}_1}\right) + \frac{\bar{c}_1}{1 - \bar{c}_1} \geq (1 - \delta) \mu + 1,$$

since $\frac{\bar{c}_1}{1 - \bar{c}_1} \geq 1$ when $\bar{c}_1 \geq \bar{c}_1 \geq 0.5$.

Consider the final sets $\tilde{S}_0$, $\tilde{S}_1$, and $\tilde{S}_2$ of LargeFirst$(I; v, K, \tau)$. If $f(\tilde{S}_0) \geq \beta v = 0.49v$, then we are done. Recall that there exists $e' \in T$ such that $f(\text{OPT} - o_1 \mid e')$ and $f(\text{OPT} - o_1 - o_2 \mid e')$ are bounded as in Lemma 9. Hence, by Lemma 10, the output of LargeFirst$(I; v, K, \tau)$ is lower-bounded by the RHSs of the following two inequalities:

$$f(\tilde{S}_1) \geq \left(\tau + (\Gamma(\tau) - \tau) \left(1 - e^{-(1-\delta)\mu}\right) - O(\varepsilon)\right)v,$$

$$f(\tilde{S}_2) \geq \left(\tau + (\Gamma(\tau) - \beta) \left(1 - e^{-\left(1 - \frac{1 - \beta}{\mu} - 1\right)}\right) - O(\varepsilon)\right)v.$$

The above inequalities (15)–(17) imply that one of $\tilde{S}, \tilde{S}_\ell$ ($\ell = 0, 1, 2$) admits a $(0.49 - O(\varepsilon))$-approximation.

More specifically, we can obtain the ratio as follows. First suppose that $\mu \geq 0.505$. Then, if $\bar{\tau} \leq 0.3562$, then (15) implies that

$$f(\tilde{S}) \geq (1 - 0.3562) \left(1 - e^{-(1-\delta)0.505 + 1}\right) O(\varepsilon) v \geq (0.50 - O(\varepsilon))v.$$

If $0.4 \geq \tau \geq 0.3562$, then (16) implies that

$$f(\tilde{S}_1) \geq \left(0.3562 + \left(\frac{9}{10} - \frac{3 \cdot 0.3562}{2}\right) \left(1 - e^{-(1-\delta)0.505}\right) - O(\varepsilon)\right)v \geq (0.50 - O(\varepsilon))v$$

since $\Gamma(\bar{\tau}) = 9/10 - \bar{\tau}/2$ when $\bar{\tau} \leq 0.4$. If $\tau \geq 0.4$, then (16) implies that

$$f(\tilde{S}_1) \geq \left(0.4 + \left(\frac{5}{6} - \frac{4 \cdot 0.4}{3}\right) \left(1 - e^{-(1-\delta)0.505}\right) - O(\varepsilon)\right)v \geq (0.51 - O(\varepsilon))v.$$
since \( \Gamma(\overline{\tau}) = 5/6 - \overline{\tau}/3 \) when \( \overline{\tau} \geq 0.4 \). Thus we obtain a \((0.5 - O(\varepsilon))\)-approximation when \( \mu \geq 0.505 \) using (15) and (16).

Next suppose that \( \mu < 0.505 \). Recall that we may suppose that \( \overline{\tau} \geq 0.22 \) as discussed in the beginning of the proof. First assume that \( \mu \geq 3.5 \). Then (1) implies that

\[
\begin{align*}
f(\tilde{S}) &\geq (1 - \overline{\tau}) \left( 1 - e^{-((1-\delta)3.5(\overline{\tau}-0.22)+1)} - O(\varepsilon) \right) v.
\end{align*}
\]

Since we assume \( \mu < 0.505 \), we have \( 0.505 > \mu \geq 3.5 \) and hence \( \overline{\tau} \leq 0.365 \) holds. Therefore, the above lower bound turns out to be at least \((0.50 - O(\varepsilon))v\). Next assume that \( \mu < 3.5 \). Then it holds that \( \overline{\tau} \geq 2\mu/7 + 0.22 \). Since \( \Gamma(\overline{\tau}) \geq 5/6 - \overline{\tau}/3 \), (17) implies that

\[
\begin{align*}
f(\tilde{S}_2) &\geq \left( \frac{2}{7} \mu + 0.22 + \left( \frac{5}{6} - \frac{1}{3} \left( \frac{2}{7} \mu + 0.22 \right) \right) - 0.49 \right) \left( 1 - e^{-\frac{(1-\delta)\varepsilon}{\mu}-1} \right) - O(\varepsilon) v,
\end{align*}
\]

which is at least \((0.49 - O(\varepsilon))v\) when \( \mu < 0.505 \). Thus the statement holds. \( \square \)

We remark that the proof of Lemma 11 works when \( K \geq W \), and that of Lemma 12 works when \( K \geq W \) and \( c_1 \geq \varepsilon_1 \geq 0.5W \).

In summary, we have the following theorem.

**Theorem 6** Suppose that we are given an instance \( I = (f, c, K, E) \) for the problem (1). Then we can find a \((0.46 - \varepsilon)\)-approximate solution in \( O(\varepsilon^{-1}) \) passes and \( O(K\varepsilon^{-4} \log K) \) space. The total running time is \( O(ne^{-5} \log K) \).

**Proof** As mentioned in the beginning of Section 3, if we can design an algorithm running in \( O(T_1) \) time and \( O(T_2) \) space provided the approximate value \( v \), then the total running time of the algorithm guessing \( v \) is \( O(ne^{-1} \log K + e^{-1}T_1) \) and the space required is \( O(\max\{KE^{-1} \log K, e^{-1}T_2\}) \). Thus suppose that we are given \( v \) such that \( v \leq f(OPT) \leq (1+\varepsilon)v \).

By Lemma 7, we may assume that \( c_1 + c_2 \leq 1 - \varepsilon/\delta \) where \( \delta = 0.01 \). We may further assume that \( c_1 \geq 0.383 \), as otherwise Lemma 6 implies that \( \text{Simple}(I; v, K) \) yields a \((0.46 - \varepsilon)\)-approximation. For \( i \in \{1, 2\} \), we guess \( c_i, \tilde{c}_i \) such that \( c_i \leq c_i \leq \tilde{c}_i \) and \( \tilde{c}_i \leq (1+\varepsilon)c_i \) by a geometric series. This takes \( O(e^{-2} \log K) \) space, since the range of \( c_1 \) is \([0.383, 0.46]\) and that of \( c_2 \) is \([1/K, 1]\).

When \( c_1 \leq 0.5 \), it follows from Lemma 11 that we can find a \((0.46 - \varepsilon)\)-approximate solution in \( O(e^{-1}) \) passes and \( O(e^{-1}K) \) space. When \( c_1 \geq 0.5 \), it follows from Lemma 12 that we can find a \((0.49 - \varepsilon)\)-approximate solution in \( O(e^{-1}) \) passes and \( O(e^{-1}K) \) space. Hence, for each \( c_i, \tilde{c}_i \), it takes \( O(e^{-1}) \) passes and \( O(e^{-1}K) \) space, running in \( O(ne^{-2}) \) time in total. Thus, for a fixed \( v \), the space required is \( O(Ke^{-3} \log K) \), and the running time is \( O(ne^{-4} \log K) \). Therefore, the algorithm in total uses \( O(e^{-1}) \) passes and \( O(Ke^{-4} \log K) \) space, running in \( O(ne^{-5} \log K) \) time. Thus the statement holds. \( \square \)
5 Improved 0.5-Approximation Algorithm

In this section, we further improve the approximation ratio to 0.5. Recall that we may assume that we are given $v$ with $v \leq f(\text{OPT}) \leq (1 + \varepsilon)v$ by taking more $O(\varepsilon^{-1})$ space.

5.1 Overview

We first remark that algorithms so far give us a $(0.5 - \varepsilon)$-approximation for some special cases. In fact, Lemma 6 and Corollary 2 lead to a $(0.5 - \varepsilon)$-approximation when $c(o_1) \leq 0.3K$ or $f(o_1) \leq 0.15v$.

**Corollary 4** If $c(o_1) \leq 0.3K$ or $f(o_1) \leq 0.15v$, then we can find a set $S$ in $O(\varepsilon^{-1})$ passes and $O(K)$ space such that $c(S) \leq K$ and $f(S) \geq (0.5 - O(\varepsilon))v$.

**Proof** First suppose that $c(o_1) \leq 0.3K$. By Lemma 6, the output $S$ of $\text{Simple}(I; v, K)$ satisfies that

$$f(S) \geq \left(1 - e^{-\frac{K - c(o_1)}{K}} - O(\varepsilon)\right)v \geq \left(1 - e^{-0.7} - O(\varepsilon)\right)v \geq (0.5 - O(\varepsilon))v.$$

Next suppose that $f(o_1) \leq 0.15v$. We see that $f(\text{OPT} - o_1) \geq f(\text{OPT}) - f(o_1) \geq 0.85v$. By Corollary 2, we can find a set $S$ such that $K - c(o_2) < c(S) \leq K$ and

$$f(S) \geq 0.85 \left(1 - e^{-1} - O(\varepsilon)\right)v \geq (0.5 - O(\varepsilon))v.$$  

Moreover, the following corollary asserts that we may suppose that $f(o_1)$ and $f(o_2)$ are small.

**Corollary 5** In the following cases, $\text{LargeFirst}$, together with $\text{Simple}$, can find a $(0.5 - \varepsilon)$-approximate solution in $O(\varepsilon^{-1})$ passes and $O(K \varepsilon^{-4} \log K)$ space:

1. when $c_1 \geq 0.5$ and $f(o_1) \geq 0.362v$.
2. when $c_1 \leq 0.5$ and $f(o_1) \geq 0.307v$.
3. when $f(o_2) \geq 0.307v$.

**Proof** We may suppose that $c_1 + c_2 \leq 1 - \varepsilon/\delta$, where $\delta = 0.01$, by Lemma 7.

(1) Suppose that $c_1 \geq 0.5$ and $f(o_1) \geq 0.362v$. We may also suppose that $f(o_1) < 0.5v$, as otherwise we can just take a singleton with maximum return from $E$. We guess $\tau$ and $\bar{\tau}$ such that $0.362v \leq \tau v \leq f(\text{OPT}) \leq \bar{\tau} v \leq 0.5v$ and $\bar{\tau} \leq (1 + \varepsilon)\tau$ from the interval $[0.362, 0.5]$ by a geometric series using $O(\varepsilon^{-1})$ space. Consider applying $\text{LargeFirst}(I; v, K, \tau)$ with $\beta = 0.5$. By Lemmas 9 and 10, the output of $\text{LargeFirst}(I; v, K, \tau)$ is lower-bounded by the RHSs of the inequalities (16) and (17).

First suppose that $\mu = \frac{1 - \tau_1 - \tau_2}{1 - \tau_1} \geq 0.495$. Then (16) implies that

$$f(\tilde{S}_1) \geq \left(\tau + (\Gamma(\psilon) - \tau) \left(1 - e^{-(1-\delta)0.495}\right) - O(\varepsilon)\right)v.$$
If $\tau \geq 0.4$, then $\Gamma(\tau) = 5/6 - \tau/3$, which implies that

$$f(\tilde{S}_1) \geq 0.4 + \left( \frac{5}{6} - \frac{4}{3} \cdot 0.4 \right) \left( 1 - e^{-(1-\delta)0.495} - O(\varepsilon) \right) v \geq (0.51 - O(\varepsilon))v,$$

and if $\tau < 0.4$, then $\Gamma(\tau) = 9/10 - \tau/2$, and hence

$$f(\tilde{S}_1) \geq 0.362 + \left( \frac{9}{10} - \frac{3}{2} \cdot 0.362 \right) \left( 1 - e^{-(1-\delta)0.495} - O(\varepsilon) \right) v \geq (0.50 - O(\varepsilon))v.$$

Next suppose that $\mu < 0.495$. Then, since we set $\beta = 0.5$, (17) implies

$$f(\tilde{S}_2) \geq \left( \tau + (\Gamma(\tau) - 0.5) \left( 1 - e^{-\left( \frac{1-\delta}{0.495} \right)} \right) - O(\varepsilon) \right) v.$$

Similarly to the previous case, if $\tau \geq 0.4$, we have

$$f(\tilde{S}_2) \geq 0.4 + \left( \frac{5}{6} - \frac{1}{3} \cdot 0.4 - 0.5 \right) \left( 1 - e^{-\left( \frac{1-\delta}{0.495} \right)} \right) v \geq (0.52 - O(\varepsilon))v.$$

and if $\tau < 0.4$, then

$$f(\tilde{S}_2) \geq 0.362 + \left( \frac{9}{10} - \frac{1}{2} \cdot 0.362 - 0.5 \right) \left( 1 - e^{-\left( \frac{1-\delta}{0.495} \right)} \right) v \geq (0.50 - O(\varepsilon))v,$$

Thus the statement holds.

(2) The argument is similar to (1). Suppose that $c_1 \leq 0.5$ and $f(o_1) \geq 0.307v$. We guess $\tau$ and $\tau$ such that $0.307v \leq \tau v \leq f(OPT) \leq \tau v \leq 0.50v$ and $\tau \leq (1 + \varepsilon)\tau$ from the interval $[0.307, 0.5]$ by a geometric series using $O(\varepsilon^{-1})$ space. Consider applying LargeFirst($I; v, K, \tau$) with $\beta = 0.5$. By Lemmas 8 and 10, the output of LargeFirst($I; v, K, \tau$) is lower-bounded by the RSHs of (13) and (14).

First suppose that $\mu = \frac{1-\tau_1 - \tau_2}{1-\tau_1} \geq 0.495$. Then (13) implies that

$$f(\tilde{S}_1) \geq \left( 0.307 + (1 - 0.5) \left( 1 - e^{-\left( \frac{1-\delta}{0.495} \right)} \right) - O(\varepsilon) \right) v \geq (0.50 - O(\varepsilon))v.$$

Next suppose that $\mu < 0.495$. Then (14) implies that

$$f(\tilde{S}_2) \geq \left( 0.307 + (1 - 2 \cdot 0.5 + 0.307) \left( 1 - e^{-\left( \frac{1-\delta}{0.495} \right)} \right) - O(\varepsilon) \right) v \geq (0.50 - O(\varepsilon))v.$$

Thus the statement holds.

(3) This case can be shown by applying LargeFirst to $E_2$. More precisely, we replace $c_1$ with $c_2$ in LargeFirst with $\tau v \geq f(o_2) \geq \tau v/(1 + \varepsilon)$. We also set $W_1 = W - c_2 K$ instead of $W - c_1 K$. Then, since $(\tau_1 + \tau_2)K \leq K$, we can use the same
analysis as in the proof of Lemma 11; the output of LargeFirst(I; v, W, τ) with β = 0.5 is lower-bounded by the RHSs of the following two inequalities:

\[
\begin{align*}
    f(\tilde{S}_1) & \geq \left(\tau + (1 - \beta) \left(1 - e^{-\left(1 - \delta\right)\mu'}\right) - O(\varepsilon)\right) v, \\
    f(\tilde{S}_2) & \geq \left(\tau + (1 - 2\beta + \tau) \left(1 - e^{-\left(1 - \frac{1}{\mu'}\right)}\right) - O(\varepsilon)\right) v,
\end{align*}
\]

where \(\mu' = \frac{1 - \tau_1 - \tau_2}{1 - \tau_2}\). Since the lower bounds are the same as (13) and (14) in the proof (2) of this corollary, the statement holds. \(\square\)

Recall that, in Section 4, we found an item \(e\) such that the RHSs in (6) are large, that is, \(f(\text{OPT} - o_1 | e)\) and \(f(\text{OPT} - o_1 - o_2 | e)\) are large. In this section, we aim to find a good set \(Y \subseteq E\) such that \(f(\text{OPT}' | Y)\) is large for some \(\text{OPT}' \subseteq \text{OPT}\), using \(O(\varepsilon^{-1})\) passes, while guaranteeing that the remaining space \(K - c(Y)\) is sufficiently large. We then solve the problem of maximizing the function \(f(\cdot | Y)\) to approximate \(\text{OPT}'\) with algorithms in previous sections. Specifically, we devise two strategies depending on the size of \(c_1 + c_2\) (see Sections 5.2 and 5.3 for more specific values of \(c_1\) and \(c_2\)).

**First Strategy: Packing Small Items First.**

First consider the case when \(c_1 + c_2\) is large. Recall that \(f(o_1)\) and \(f(o_2)\) are supposed to be small by Corollary 5. Hence, there is a “dense” set \(\text{OPT} - o_1 - o_2\) of small items, i.e., \(f(\text{OPT}\backslash\{o_1, o_2\})\) is large. Therefore, we consider collecting such small items. However, if we apply Simple to the original instance (1) to approximate \(\text{OPT} - o_1 - o_2\), then we can only find a set whose function value is at most \(f(\text{OPT} - o_1 - o_2)\).

The main idea of this case is to stop collecting small items early. That is, we introduce

\[
\text{maximize } f(S) \text{ subject to } c(S) \leq K_1, \quad S \subseteq E, \tag{18}
\]

where \(K_1 \leq K - c(o_1)\), and apply Simple to this instance to approximate \(\text{OPT} - o_1 - o_2\). Let \(Y\) be the output. The key observation is that, in Phase 2, since we still have space to take \(o_1\), we may assume that \(f(\text{OPT} - o_1 | Y) \geq 0.5v\) in a way similar to Lemma 8.

Given such a set \(Y\), define \(g(\cdot) = f(\cdot | Y)\) and the problem:

\[
\text{maximize } g(S) \text{ subject to } c(S) \leq K - c(Y), \quad S \subseteq E. \tag{19}
\]

We apply approximation algorithms in Sections 3–4 to approximate \(\text{OPT} - o_1\), using the fact that \(g(\text{OPT} - o_1) \geq 0.5v\) and \(c(\text{OPT} - o_1) \leq (1 - \varepsilon_1)K\). Let \(\tilde{S}\) be the output of this phase. Then \(Y \cup \tilde{S}\) is a feasible set to the original instance, and it holds that \(f(Y \cup \tilde{S}) = f(Y) + g(\tilde{S})\).

We remark that the lower bound for \(f(Y)\) depends on the size \(c(Y)\) by Corollary 1, and that for \(g(\tilde{S})\) depends on the knapsack capacity \(K - c(Y)\). Hence the lower bound for \(f(Y \cup \tilde{S})\) can be represented as a function with respect to \(c(Y)\). By balancing the two lower bounds with suitable \(K_1\), we can obtain a (0.5 – \(O(\varepsilon)\))-approximation. See Sections 5.2.1 and 5.2.2 for more details.
Second Strategy: Packing Small Items Later.

Suppose that \( c_1 + c_2 \) is small. Then \( c(\text{OPT} \setminus \{o_1, o_2\}) \) is large and we do not have the dense set of small items as before. For this case, we introduce a modified version of Simple for the original problem (1) to find a good set \( Y \). The difference is that, in each round, we check whether any item in \( E \), by itself, is enough to give us a solution with value at least \( 0.5v \). Such a modification would allow us to lower bound \( f(\text{OPT}' \mid Y) \) for some \( \text{OPT}' \subseteq \text{OPT} \) for Phase 2. We may assume that \( c(Y) < 0.7K \), as otherwise we are done by Lemma 4, which means that we still have enough space to pack other items. That is, define \( g(\cdot) = f(\cdot \mid Y) \) and the problem:

\[
\text{maximize } g(S) \quad \text{subject to } c(S) \leq K - c(Y), \quad S \subseteq E.
\]  

Let \( \text{OPT}' = \{ e \in \text{OPT} \mid c(e) \leq K - c(Y) \} \). We aim to find a feasible set to this problem that approximates \( \text{OPT}' \) in Phase 2. Thanks to the modification of Simple, we can assume that \( g(\text{OPT}') \) is large. However, an extra difficulty arises if \( K - c(Y) \geq c(\text{OPT}') \). In this case, we cannot apply our algorithms developed in previous sections. For this, we need to combine Simple and LargeFirst to obtain better ratios, where the results are summarized as below (The proof will be given in Section 5.4).

**Lemma 13** Suppose that we are given an instance \( \mathcal{I}' = (f, c, K', E) \) for the problem (1). Let \( X \) be a subset such that \( c(e) \leq K' \) for any \( e \in X \) and \( c(X) \leq W' = \eta K' \), where \( \eta > 1 \). We further suppose that \( v' \leq f(X) \leq O(1)v' \). Then we can find a set \( S \) in \( O(n \varepsilon^{-4} \log K') \) time and \( O(K' \varepsilon^{-3} \log K') \) space, using \( O(\varepsilon^{-1}) \) passes, such that the following hold:

(a) If \( \eta \in [1, 1.4] \), then \( f(S) \geq (0.315 - O(\varepsilon))v' \).
(b) If \( \eta \in [1.4, 1.5] \), then \( f(S) \geq (0.283 - O(\varepsilon))v' \).
(c) If \( \eta \in [1.5, 2] \), then \( f(S) \geq (0.218 - O(\varepsilon))v' \).
(d) If \( \eta \in [2, 2.5] \), then \( f(S) \geq (0.178 - O(\varepsilon))v' \).

Using Lemma 13 with case analysis, we can find a feasible set to (20) that approximates \( \text{OPT}' \). This solution, together with \( Y \), gives a \((0.5 - O(\varepsilon))\)-approximate solution.

5.2 Packing Small Items First

5.2.1 When \( c_1 \geq 0.5 \)

In this section, we assume that \( c_1 \geq \bar{c}_1 \geq 0.5 \). Since the range of \( c_1 \) is \([0.5, 1]\), we can guess \( \bar{c}_1 \) and \( \bar{c}_1 \) using \( O(\varepsilon^{-1}) \) space. We also guess \( \bar{c}_2 \) and \( \bar{c}_2 \) using \( O(\varepsilon^{-1} \log K) \) space.

Recall that in the proof of Lemma 12, we have shown that we obtain a \((0.5 - \varepsilon)\)-approximation when \( \mu = \frac{1 - \bar{c}_1 - \bar{c}_2}{1 - \bar{c}_1} \geq 0.505 \). Therefore, in this section, we assume that \( \mu < 0.505 \), i.e.,

\[
1 - \bar{c}_1 \geq \frac{200}{101} (1 - \bar{c}_1 - \bar{c}_2) \geq 1.98(1 - \bar{c}_1 - \bar{c}_2).
\]  

This implies that \( \bar{c}_1 + \bar{c}_2 \geq 0.747 \).
Lemma 14 Suppose that \( c_1 \geq 0.5 \). Then, if \((21)\) holds, we can find a \((0.5 - \varepsilon)\)-approximate solution in \( O(\varepsilon^{-1}) \) passes and \( O(K\varepsilon^{-3} \log^2 K) \) space. The total running time is \( O(n\varepsilon^{-6} \log^2 K) \).

The rest of this subsection is devoted to the proof of the above lemma. It suffices to design an \( O(\varepsilon^{-1}) \)-pass algorithm provided the approximated value \( v \) and \( \bar{c}_i, c_j \) \((i = 1, 2)\) such that \( \bar{c}_i \leq (1 + \varepsilon)c_j \), running in \( O(K\varepsilon^{-2} \log K) \) space and \( O(n\varepsilon^{-3} \log K) \) time. We may also assume that \( c_1 + c_2 \leq 1 - \varepsilon / \delta \) where \( \delta = 0.01 \) by Lemma 7.

Finding a Good Set \( Y \) By Corollary 5, we may assume that \( f(\text{OPT} - o_1 - o_2) \) is relatively large. More specifically, \( f(\text{OPT} - o_1 - o_2) \geq f(\text{OPT}) - f(o_1) - f(o_2) \geq v - 0.362v - 0.307v \geq 0.33v \). On the other hand, \((21)\) implies that \( \bar{c}_1 + \bar{c}_2 \geq 0.747 \), which means that \( c(\text{OPT} - o_1 - o_2) \) is small, that is, \( c(\text{OPT} - o_1 - o_2) \leq 0.253K \).

We consider collecting such a “dense” set of small items by introducing

\[
\text{maximize } f(S) \text{ subject to } c(S) \leq 1.98c_s, \quad S \subseteq E,
\]

where we define \( c_s = 1 - \bar{c}_1 - \bar{c}_2 \). We apply Simple to \((22)\) to find a set \( Y \) that approximates \( \text{OPT} - o_1 - o_2 \). By \((21)\), we have \( K - c(Y) \geq \bar{c}_1K \), and hence we still have space to take \( o_1 \) after taking \( Y \). We denote \( c_s = 1 - c_1 - c_2 \).

Lemma 15 We can find a subset \( Y \) in \( O(\varepsilon^{-1}) \) passes and \( O(K) \) space such that

\[
f(Y) \geq 0.33 \left(1 - e^{-\frac{c(Y)}{c_s}}\right) v - O(\varepsilon)v \quad \text{and}
1.98c_sK \geq c(Y) \geq (0.98c_s - 1.98\epsilon(c_1 + c_2))K.
\]

Moreover, if \( f(Y + o_1) < 0.5v \), then \( f(\text{OPT} - o_1 \mid Y) \geq 0.5v \).

Proof The first inequality follows from Corollary 1 applied to approximate \( \text{OPT} - o_1 - o_2 \) for the instance \((22)\), noting that \( f(\text{OPT} - o_1 - o_2) \geq 0.33v \) and \( c(\text{OPT} - o_1 - o_2) \leq c_sK \). Since items in \( \text{OPT} - o_1 - o_2 \) are of size at most \( c(o_3) \), it is obvious from Lemma 5 that \( (1.98c_s - c_3)K \leq c(Y) \leq 1.98c_sK \). Since \( c_s \geq c_s - \varepsilon(c_1 + c_2) \) and \( c_3 \leq c_s \), the lower bound is bounded by

\[
(1.98c_s - c_3)K \geq 1.98(c_s - \varepsilon(c_1 + c_2))K - c_sK = (0.98c_s - 1.98\epsilon(c_1 + c_2))K.
\]

Finally, if \( f(Y + o_1) < 0.5v \), then we have

\[
f(\text{OPT} - o_1 \mid Y) \geq f(\text{OPT} \mid Y) - f(o_1 \mid Y)
\geq (f(\text{OPT}) - f(Y)) - (0.5v - f(Y)) \geq 0.5v.
\]

Packing the Remaining Space Define \( g(\cdot) = f(\cdot \mid Y) \). Consider the problem \((19)\), and let \( \mathcal{I}' \) be the corresponding instance. We shall find a feasible set to approximate \( \text{OPT} - o_1 \). By Lemma 15, we may assume that \( g(\text{OPT} - o_1) \geq v' = v/2 \), as otherwise we can find an item \( e \) such that \( c(Y + e) \leq K \) and \( f(Y + e) \geq 0.5v \) using a single pass. Let \( W' = (1 - c_1)K \) and \( K' = K - c(Y) \). Then \( c(\text{OPT} - o_1) \leq W' \) holds.
The algorithm Simple($I'$; 0.5$v$, $W'$) can find a set $\tilde{S}$ such that
\[
g(\tilde{S}) \geq \frac{1}{2} \left( 1 - e^{-\frac{1-y-c_2}{1-c_1}} - O(\varepsilon) \right) v,
\]
where $y = c(Y)/K$.

Moreover, noting that $c(Y) \leq 1.98c_s K \leq (1 - \bar{c}_1)K \leq 0.5K \leq \xi_1 K$ since $\xi_1 \geq 0.5$ and (21), we have $W' \leq K'$. Hence we can apply a $(0.46 - \varepsilon)$-approximation algorithm in Lemmas 11 and 12 with $g(OPT - o_1) \geq v' = v/2$ and $c(OPT - o_1) \leq W'$. That is, we can find a set $\tilde{S}'$ such that
\[
g(\tilde{S}') \geq \frac{1}{2} (0.46 - O(\varepsilon)) v = (0.23 - O(\varepsilon)) v.
\]

Then $Y \cup \tilde{S}$ and $Y \cup \tilde{S}'$ are both feasible sets to the original instance. By Lemma 15, we have
\[
f(Y \cup \tilde{S}) = f(Y) + g(\tilde{S}) \geq 0.33 \left( 1 - e^{-\frac{y}{c_s}} \right) v + \frac{1}{2} \left( 1 - e^{-\frac{1-y-c_2}{1-c_1}} \right) v - O(\varepsilon) v, \quad (23)
\]
\[
f(Y \cup \tilde{S}') = f(Y) + g(\tilde{S}') \geq 0.33 \left( 1 - e^{-\frac{y}{c_s}} \right) v + 0.23 v - O(\varepsilon) v. \quad (24)
\]

Since each bound is a concave function with respect to $y$, the worst case is achieved when $y = 0.98c_s - 1.98\varepsilon(c_1 + c_2)$ or $1.98c_s$.

Suppose that $y = 0.98c_s - 1.98\varepsilon(c_1 + c_2)$. Then it holds that
\[
\frac{y}{c_s} = 0.98 - 1.98\varepsilon \frac{c_1 + c_2}{1 - c_1 - c_2} \geq 0.98 - 1.98\delta,
\]
assuming that $c_1 + c_2 \leq 1 - \varepsilon/\delta$. Moreover, since $y \leq c_s = 1 - c_1 - c_2$,
\[
\frac{1 - y - c_2}{1 - c_1} \geq \frac{1 - (1 - c_1 - c_2) - c_2}{1 - c_1} \geq \frac{c_1}{1 - c_1} - \frac{c_2}{1 - c_1} \geq 1 - \varepsilon,
\]
where the last inequality follows since $c_1 \geq 0.5$ and $c_2 \leq 1 - c_1$. Hence, by (23), we obtain
\[
f(Y \cup \tilde{S}) \geq 0.33 \left( 1 - e^{-\left(0.98 - 1.98\delta\right)} \right) v + \frac{1}{2} \left( 1 - e^{-1} \right) v - O(\varepsilon) v \geq (0.51 - O(\varepsilon)) v
\]
when $\delta = 0.01$.

Suppose that $y = 1.98c_s$. Then we have
\[
\frac{y}{c_s} = 1.98 \frac{c_s}{c_s} \geq 1.98 (1 - \delta).
\]

Hence (24) implies that
\[
f(Y \cup \tilde{S}') \geq 0.33 \left( 1 - e^{-1.98(1-\delta)} \right) v + 0.23 v - O(\varepsilon) v \geq (0.51 - O(\varepsilon)) v
\]
when $\delta = 0.01$. 
Therefore, it follows that the maximum of $f(Y \cup \tilde{S})$ and $f(Y \cup \tilde{S}')$ is at least $(0.51 - O(\varepsilon))v$ for any $c(Y)$. Thus we can find a $(0.5 - O(\varepsilon))$-approximate solution assuming (21).

In the above, we apply the algorithms in Section 4 to $I'$ to approximate $\text{OPT} - o_1$. To do it, we need to have approximated sizes of $c(o_2)$ and $c(o_3)$, which are the two largest items in $\text{OPT} - o_1$. Since $\overline{c}_2$, $\overline{c}_3$ are given in the beginning, it suffices to guess approximated values $\overline{c}_3$ and $\overline{c}_3$ of $c(o_3)$ using $O(\varepsilon^{-1} \log K)$ space. Therefore, the space required is $O(K \varepsilon^{-2} \log K)$ and the running time is $O(n \varepsilon^{-3} \log K)$. The proof of Lemma 14 is complete.

In summary, when $c_1 \geq 0.5$, we have the following, combining the above discussion with Lemmas 7 and 12.

**Theorem 7** For any instance $I = (f, c, K, E)$ for the problem (1) when $c_1 \geq 0.5$, we can find a $(0.5 - \varepsilon)$-approximate solution in $O(\varepsilon^{-1})$ passes and $O(K \varepsilon^{-3} \log^2 K)$ space. The total running time is $O(n \varepsilon^{-6} \log^2 K)$.

### 5.2.2 When $c_1 \leq 0.5$

In this section, we assume that $c_1 \leq \overline{c}_1 \leq 0.5$. Note that we may assume that $c_1 \geq 0.3$ by Corollary 4. Furthermore, we suppose that

$$2.4(1 - \overline{c}_1 - \overline{c}_2) \leq 1 - \overline{c}_1. \quad (25)$$

(Section 5.3 handles the case when this inequality does not hold.) This is equivalent to $7\overline{c}_1 + 12\overline{c}_2 \geq 7$, implying that $\overline{c}_1 + \overline{c}_2 \geq 14/19 \geq 0.735$, where the minimum is when $\overline{c}_1 = \overline{c}_2$. Thus $\overline{c}_1 \geq 7/19 \geq 0.36$ and $\overline{c}_2 \geq 0.735 - \overline{c}_1 \geq 0.235$.

The argument is similar to the previous subsection. That is, we first try to find a dense set of small items, and then apply algorithms in Sections 3–4.

**Lemma 16** Suppose that $0.3 \leq c_1 \leq 0.5$. Then, if (25) holds, we can find a $(0.5 - \varepsilon)$-approximate solution in $O(\varepsilon^{-1})$ passes and $O(K \varepsilon^{-6} \log K)$ space. The total running time is $O(n \varepsilon^{-7} \log K)$.

The rest of this subsection is devoted to the proof of the above lemma. Since the range of $c_1$ is $[0.3, 0.5]$, we can guess $\overline{c}_1$ and $\overline{c}_1$, where $\overline{c}_1 \geq 0.3$ and $\overline{c}_1 \leq 0.5$, using $O(\varepsilon^{-1})$ space. We also guess $\overline{c}_2$ and $\overline{c}_2$ using $O(\varepsilon^{-1})$ space, since the range of $c_2$ is $[0.235, 0.5]$ by (25). Therefore, it suffices to design an $O(\varepsilon^{-1})$-pass algorithm provided the approximated value $v$ and $\overline{c}_i$, $\overline{c}_i$ ($i = 1, 2$) such that $\overline{c}_i \leq (1 + \varepsilon)\overline{c}_i$, running in $O(K \varepsilon^{-3} \log K)$ space and $O(n \varepsilon^{-4} \log K)$ time. We may also assume that $c_1 + c_2 \leq 1 - \varepsilon/\delta$ where $\delta = 0.01$ by Lemma 7.

**Finding a Good Set $Y$** By Corollary 5, we may assume that $f(\text{OPT} - o_1 - o_2)$ is relatively large, while $c(\text{OPT} - o_1 - o_2)$ is small. More specifically, $f(\text{OPT} - o_1 -
\( o_2 \) ≥ \( f(\text{OPT}) - f(o_1) - f(o_2) \) ≥ \( v - 0.307v - 0.307v \geq 0.386v \), but \( c(\text{OPT} - o_1 - o_2) \leq (1 - \epsilon_1 - \epsilon_2) \leq (5/19 + \epsilon)K \). We consider collecting such a “dense” set of small items by introducing

\[
\text{maximize } f(S) \text{ subject to } c(S) \leq 2.4c_xK, \quad S \subseteq E, \tag{26}
\]

where we recall \( c_x = 1 - \overline{c_1} - \overline{c_2} \). By (25), we still have space to take \( o_1 \) after applying Simple to (26). We denote \( \overline{c_x} = 1 - \epsilon_1 - \epsilon_2 \).

Similarly to Lemma 15, we have the following lemma.

**Lemma 17** We can find a subset \( Y \) in \( O(\varepsilon^{-1}n) \) time and \( O(K) \) space such that

\[
f(Y) \geq 0.386 \left( 1 - e^{-\frac{c(y)}{2.4K}} \right)v + O(\varepsilon)v \quad \text{and} \quad 2.4c_xK \geq c(Y) \geq (2.4c_x - c_3)K.
\]

Moreover, if \( f(Y + o_1) < 0.5v \), then \( f(\text{OPT} - o_1 \mid Y) \geq 0.5v \).

**Packing the Remaining Space** Let \( Y \) be a set found by Lemma 17. Define \( g(\cdot) = f(\cdot \mid Y) \). Consider the problem (19). By Lemma 17, we may assume that \( g(\text{OPT} - o_1) \geq v/2 \) by checking whether adding an item \( e \) to \( Y \) gives us a 0.5-approximation using a single pass. We set \( W' = (1 - \epsilon_1)K \geq c(\text{OPT} - o_1) \) and \( K' = K - c(Y) \). There are two cases depending on the sizes of \( W' \) and \( K' \). Note that \( K' \geq W' \) if and only if \( y \leq c_1 \), where we denote \( y = c(Y)/K \).

(a) \( y \leq c_1 \). In this case, \( K' \geq W' \) holds. Hence we can apply our algorithm in Section 4 with \( g(\text{OPT} - o_1) \geq v' = 0.5v \) and \( c(\text{OPT} - o_1) \leq W' \). Our algorithm in fact admits a \((0.49 - \varepsilon)\)-approximation by Lemma 12 since the biggest size in \( \text{OPT} - o_1 \) is \( c_2K \) and, by (25),

\[
c_2K \geq (1 - \varepsilon)c_xK \geq \frac{1.4}{2.4}(1 - \overline{c_1})K - \overline{c_2}\varepsilon K \geq 0.5W',
\]

when \( \varepsilon \) is small, e.g., \( \varepsilon < 1/12 \). Let \( S \) be the obtained set, that is, it satisfies that \( c(S) \leq K - c(Y) \) and \( g(Y) \geq (0.49 - O(\varepsilon))v' \). Then \( Y \cup S \) is a feasible set to the original instance.

By Lemma 17, the set \( Y \cup S \) satisfies

\[
f(Y \cup S) = f(Y) + g(S) \geq 0.386 \left( 1 - e^{-\frac{v'}{c_2}} \right)v + 0.5 \cdot 0.49v - O(\varepsilon)v. \tag{27}
\]

Since \( y \geq 2.4c_x - c_3 \geq 2.4c_x - \overline{c_x} \) by Lemma 17, the exponent in (27) is

\[
y = \frac{v}{c_x} \geq 2.4 \frac{c_x}{c_x} - 1 \geq 2.4(1 - \delta) - 1 \geq 1.4 - 2.4\delta,
\]

when \( \epsilon_1 + \epsilon_2 \leq 1 - \varepsilon/\delta \). Hence the RHS of (27) is lower-bounded by

\[
0.386 \left( 1 - e^{-1.4 + 2.4\delta} \right)v + 0.5 \cdot 0.49v - O(\varepsilon)v \geq (0.53 - O(\varepsilon))v.
\]

To apply the algorithms in Section 4 to approximate \( \text{OPT} - o_1 \), we need to have approximated sizes of \( c(o_2) \) and \( c(o_3) \). Since we need to guess \( \overline{c_3}, \overline{c_3} \) using \( O(\varepsilon^{-1} \log K) \) additional space, the space required is \( O(K \varepsilon^{-2} \log K) \) and the running time is \( O(ne^{-3} \log K) \).
(b) \( y > c_1 \) In this case, \( K' < W' \) holds. We consider the problem (19) to approximate \( \text{OPT} - a_1 - a_2 \). Recall that \( v' = v/2 \).

Suppose that \( \tau v' \geq g(o_2) \geq \tau v'/(1 + \varepsilon) \). Since \( g(\text{OPT} - a_1) \geq v' \), it holds that \( g(\text{OPT} - a_1 - a_2) \geq g(\text{OPT} - a_1) - g(o_2) \geq (1 - \tau)v' \). Since \( v' = v/2 \), it follows from Corollary 1 that we can find a set \( S_1 \) such that \( c(S_1) \leq K - c(Y) \) and

\[
g(S_1) \geq \frac{1}{2}(1 - \tau) \left( 1 - e^{-\frac{1 - v - c}{\tau}} \right) v - O(\varepsilon)v. \tag{28}
\]

Moreover, if we take a singleton \( e \) with maximum return \( g(e) \) such that \( c(e) \leq K - c(Y) \), then letting \( S_2 = \{ e \} \), we have \( c(S_2) \leq K - c(Y) \) and

\[
g(S_2) \geq g(o_2) \geq \frac{1}{2}(1 - \tau)v - O(\varepsilon)v. \tag{29}
\]

Note that \( c(\text{OPT} - a_1 - a_2) = (1 - c_1 - c_2)K \leq (5/19 + \varepsilon)K \) and \( K' = K - c(Y) \geq \overline{c}_1 K \geq 7/19K \), and hence \( c(\text{OPT} - a_1 - a_2) \leq K' \) when \( \varepsilon \) is small. Therefore, Lemmas 11 and 12 are applicable to approximate \( \text{OPT} - a_1 - a_2 \), and we can find a set \( S_3 \) such that \( c(S_3) \leq K - c(Y) \) and

\[
g(S_3) \geq \frac{1}{2}(1 - \tau)0.46v - O(\varepsilon)v. \tag{30}
\]

Then the lower bound of the best solution is

\[
\max\{ f(Y \cup S_\ell) \mid \ell = 1, 2, 3 \} \geq 0.386 \left( 1 - e^{-\frac{c_1}{\tau}} \right) v + \max \left\{ g(S_\ell) \mid \ell = 1, 2, 3 \right\} - O(\varepsilon)v.
\]

Since every bound is a concave function with respect to \( y \), the worst case is achieved when \( y = c_1 \) or \( 2.4c_1 \). Recall that \( \overline{c}_1 \geq 7/19 \) and \( \overline{c}_1 + \overline{c}_2 \geq 14/19 \).

Suppose that \( y = c_1 \). If \( \tau \geq 0.42 \), then (29) implies that

\[
f(Y \cup S_2) \geq 0.386 \left( 1 - e^{-\frac{c_1}{\tau}} \right) v + \frac{1}{2}0.42v - O(\varepsilon)v \geq (0.50 - O(\varepsilon))v,
\]

since \( \frac{c_1}{\tau} \geq \frac{7/19 - \varepsilon}{5/19 + \varepsilon} \geq 1.4 - O(\varepsilon) \). If \( \tau \leq 0.42 \), then (28) implies

\[
f(Y \cup S_1) \geq 0.386 \left( 1 - e^{-\frac{c_1}{\tau}} \right) v + \frac{1}{2}(1 - 0.42) \left( 1 - e^{-\frac{1 - \overline{c}_1 - \overline{c}_3}{\tau}} \right) \varepsilon v. \tag{31}
\]

Since \( \overline{c}_3 \leq \overline{c}_s \leq 5/19 + \varepsilon \), we have

\[
\frac{c_1}{\overline{c}_s} \geq \frac{19}{5} \overline{c}_1 - O(\varepsilon) \text{ and } \frac{1 - \overline{c}_1 - \overline{c}_3}{\overline{c}_s} \geq \frac{1}{K - c(Y)} - 1 \geq \frac{19}{5} (1 - \overline{c}_1) - 1 - O(\varepsilon).
\]

Hence (31) implies that

\[
f(Y \cup S_1) \geq 0.386 \left( 1 - e^{-\frac{19}{5} \overline{c}_1} \right) v + 0.29 \left( 1 - e^{-\frac{19}{5} (1 - \overline{c}_1)} v - O(\varepsilon)v
\]

\[
\geq (0.50 - O(\varepsilon))v
\]

as \( 0.5 \geq \overline{c}_1 \geq 7/19 \).
Suppose that $y = 2.4c_s$. Then we have $\frac{y}{\ell} \geq 2.4(1 - \delta)$, since $c_1 + c_2 \leq 1 - \varepsilon/\delta$.
If $\tau \geq 0.314$, then (29) implies that
$$f(Y \cup \tilde{S}_2) \geq 0.386 \left( 1 - e^{-2.4(1 - \delta)} \right) v + \frac{1}{2} 0.314 - v - O(\varepsilon) v \geq (0.50 - O(\varepsilon)) v.$$ 
If $\tau \leq 0.314$, then (30) implies that
$$f(Y \cup \tilde{S}_3) \geq 0.386 \left( 1 - e^{-2.4(1 - \delta)} \right) v + \frac{1}{2} 0.314 - 0.46 - O(\varepsilon) v \geq (0.50 - O(\varepsilon)) v.$$ 

Therefore, it holds that
$$\max\{ f(Y \cup \tilde{S}_\ell) \mid \ell = 1, 2, 3 \} \geq (0.50 - O(\varepsilon)) v.$$ 
Thus we can find a $(0.5 - O(\varepsilon))$-approximate solution.

Note that we apply the algorithms in Section 4 to approximate $\text{OPT} - o_1 - o_2$ in the above, and hence we need to estimate approximations of $c(o_3)$ and $c(o_4)$, which are the two largest items in $\text{OPT} - o_1 - o_2$. This requires $O(\varepsilon^{-2} \log K)$ space in a similar way to the proof of Theorem 6. Therefore, the space required is $O(K\varepsilon^{-3} \log K)$ and the running time is $O(n\varepsilon^{-4} \log K)$. This completes the proof of Lemma 16.

5.3 Packing Small Items Later

In this section, we consider the remaining case. By Corollary 4 and Theorem 7, it suffices to consider the case when $0.3 \leq c_1 \leq 0.5$. Moreover, we may assume that (25) does not hold, that is, $2.4(1 - \bar{c}_1 - \bar{c}_2) > 1 - \bar{c}_1$, as otherwise Lemma 16 implies a $(0.5 - \varepsilon)$-approximation. This condition is equivalent to $7\bar{c}_1 + 12\bar{c}_2 < 7$. Since $0.3 \leq \bar{c}_1 \leq 0.5$, this implies that $c_2 \leq 7/19 \leq 0.37$.

Lemma 18 Suppose that $0.3 \leq c_1 \leq 0.5$. Then, if (25) does not hold, then we can find a $(0.5 - \varepsilon)$-approximate solution in $O(\varepsilon^{-1})$ passes and $O(K\varepsilon^{-7} \log^2 K)$ space. The total running time is $O(n\varepsilon^{-8} \log^2 K)$.

We first show that we may assume that $\bar{c}_2$ is bounded from below.

Corollary 6 Suppose that $0.3 \leq c_1 \leq 0.5$ and $c_1 + c_2 \leq 1 - \varepsilon/\delta$ where $\delta = 0.01$. If $\frac{1 - \bar{c}_2}{1 - \bar{c}_1} \geq 1.3$, then we can find a set $S$ such that $f(S) \geq (0.5 - O(\varepsilon)) v$ in $O(K)$ space and $O(\varepsilon^{-1})$ passes.

Proof By Corollary 5, we may suppose that $f(o_1) < 0.307v$. If $\frac{1 - \bar{c}_2}{1 - \bar{c}_1} \geq 1.3$, then it holds that
$$\frac{1 - \bar{c}_2}{1 - \bar{c}_1} \geq \frac{1 - \bar{c}_1}{1 - \bar{c}_1} \geq 1.3(1 - \delta).$$
Hence Corollary 1 with $\tau = 0.307$ implies that we can find a set $S$ such that
$$f(S) \geq (1 - 0.307) \left( 1 - e^{-1.3(1 - \delta)} - O(\varepsilon) \right) v \geq (0.5 - O(\varepsilon)) v.$$ 

 Springer
Since the range of $c_1$ is $[0.3, 0.5]$, we can guess $c_1, \bar{c}_1$ with $\bar{c}_1 \leq (1 + \varepsilon)c_1$ using $O(\varepsilon^{-1})$ space. Moreover, the above corollary implies that we may assume that $\bar{c}_2 \geq 1 - 1.3(1 - \bar{c}_1) \geq 0.09$ as $\bar{c}_1 \geq 0.3$. Hence the range of $c_2$ is $[0.09, 0.5]$, which implies that we can guess $c_2, \bar{c}_2$ with $\bar{c}_2 \leq (1 + \varepsilon)c_2$ using $O(\varepsilon^{-1})$ space. We also guess $c_3$ and $\bar{c}_3$ using $O(\varepsilon^{-1} \log K)$ space.

To prove Lemma 18, we will show that, given such $\bar{c}_i, c_i$ ($i = 1, 2, 3$) and $v$, there is an algorithm using $O(K\varepsilon^{-3} \log K)$ space and $O(n\varepsilon^{-4} \log K)$ time.

Finding a Good Set $Y$ The first phase, called ModifiedSimple, is roughly similar to Simple (see Algorithm 3). As before, we assume $v \leq f(OPT) \leq (1 + \varepsilon)v$ and $c(OPT) \leq K$ (notice that we here set $W = K$). The difference is in that, in each round, we check whether any item in $E$, by itself, is enough to give us a solution with value at least $0.5v$ (Lines 4–5). We terminate the repetition when $c(S) > (1 - \bar{c}_1)K$.

As will be explained (see Lemma 21), we can lower-bound $f(OPT - Z | Y)$ for some subset $Z \subseteq OPT$, because $c_2$ is small.

Algorithm 3

1: procedure ModifiedSimple($\mathcal{I}; v$)
2: $S := \emptyset$.
3: repeat
4: \hspace{1em} if $\exists e \in E$ such that $f(S + e) \geq 0.5v$ and $c(S + e) \leq K$ then
5: \hspace{2em} return $S + e$.
6: \hspace{1em} $S_0 := S$ and $\alpha := \frac{(1 - \varepsilon)v - f(S_0)}{K}$.
7: \hspace{1em} for each $e \in E$ do
8: \hspace{2em} if $f(e | S) \geq \alpha c(e)$ and $c(S + e) \leq K$ then $S := S + e$.
9: \hspace{1em} $T := S \setminus S_0$.
10: \hspace{1em} until $c(S) > (1 - \bar{c}_1)K$.
11: return $S$.

It is clear that Lemma 4(1)(2) still hold in ModifiedSimple. Moreover, ModifiedSimple terminates in $O(\varepsilon^{-1}n)$ time.

In the following discussion, let $Y$ be the final output set of ModifiedSimple, $Y'$ the set in the beginning of the last round, and $T'$ be the elements added in the last round, i.e., $Y = Y' \cup T'$. We now give two different bounds on $f(Y)$. The proof is identical to Lemmas 2 and 4(3), where the first one is a stronger bound obtained in the proof of Lemma 2.

Lemma 19 1. $f(Y) \geq \left(1 - \frac{c(T')}{K}\right) e^{-\frac{c(Y')}{\varepsilon K}} - O(\varepsilon)\right)v$.
2. $f(Y) \geq \left(1 - e^{-\frac{c(Y)}{\varepsilon K}} - O(\varepsilon)\right)v$.

In what follows, we assume that $f(Y) < 0.5v$ to avoid triviality. Then we may assume that $c(Y) \leq 0.7K$, as otherwise, Lemma 19(2) immediately implies that $f(Y) \geq (0.5 - O(\varepsilon))v$ (cf. Corollary 4).
Lemma 20 Suppose that $f(Y) < 0.5v$. Then for any $j$, we have $f(o_j \mid Y) \leq \left( e^{-c(Y)/K} - 0.5 \right) v$.

Proof By submodularity, $f(o_j \mid Y) \leq f(o_j \mid Y')$. As $c(Y') < (1 - \bar{c}_1)K$ and $f(Y) < 0.5v$, in the last round, Lines 4–5 imply that every item $e$, including $o_j$, has $f(e \mid Y') \leq 0.5v - f(Y') \leq \left( e^{-c(Y')/K} - 0.5 \right) v$, where the last inequality follows by Lemma 19(2).

Lemma 21 If $f(Y) < 0.5v$ and $c_2 \leq 0.37$, then the following are satisfied:

Case 1: If $c(Y) \geq (1 - \bar{c}_1)K$ then $f(OPT - o_1 \mid Y) \geq 0.693v - f(Y)$.

Case 2: If $c(Y) \geq (1 - \bar{c}_2)K$ then $f(OPT - o_1 - o_2 \mid Y) \geq 0.54v - f(Y)$.

Case 3: If $c(Y) \geq (1 - \bar{c}_3)K$ then $f(OPT - o_1 - o_2 - o_3 \mid Y) \geq 0.567v - f(Y)$.

Proof Case 1 follows immediately, as $f(o_1) \leq 0.307v$ by Corollary 5(2).

For Case 2, since $\bar{c}_2 \leq 0.37$, we can assume that $c(Y) \geq (1 - \bar{c}_2)K \geq 0.63K$.

Claim If $c(Y) \geq 0.63K$ and $c(T') \geq 0.315K$, then $f(Y) \geq (0.5 - O(\varepsilon))v$.

Proof We write $c(Y)/K = a$ and $c(T')/K = b$. Then Lemma 19(1) implies that

$$f(Y) \geq \left( 1 - (1 - b)e^{-(a - b)} - O(\varepsilon) \right) v.$$

We lower-bound the function $h(a, b) = 1 - (1 - b)e^{b - a}$ as follows. As $\frac{\partial h}{\partial a}, \frac{\partial h}{\partial b} \geq 0$, we plug in the lower bound of $a$ and $b$ into $h$. By assumption, $b \geq 0.315$; $a = c(Y)/K \geq 0.63$. Then $h(a, b) \geq 1 - 0.685e^{-0.315} \geq 0.5$. The proof follows.

Therefore, we may assume that $c(T') < 0.315K$. This implies that $c(Y') = c(Y) - c(T') > 0.315K$. Hence we have $f(o_j \mid Y) \leq \left( e^{-0.315} - 0.5 \right) v \leq 0.2297v$ for $j = 1, 2$ by Lemma 20. Therefore, $f(OPT - o_1 - o_2 \mid Y) \geq 0.54v - f(Y)$ holds as $f(OPT - o_1 - o_2 \mid Y) \geq f(OPT \mid Y) - f(o_1 \mid Y) - f(o_2 \mid Y)$ and $f(OPT \mid Y) \geq v - f(Y)$.

We can prove Case 3 in a similar way to Case 2. Since $\bar{c}_3 \leq 1/3$, we can assume that $2/3K \leq c(Y) \leq 0.7K$. Then, similarly to the above claim, we can show that, if $c(Y) \geq 2/3K$ and $c(T') \geq 0.22K$, then $f(Y) \geq (0.5 - O(\varepsilon))v$. Hence we may assume that $c(T') < 0.22K$. This implies that $c(Y') = c(Y) - c(T') \geq (2/3 - 0.22)K > 0.44K$. Hence, by Lemma 20, it holds that $f(o_j \mid Y) < 0.144v$ for $j = 1, 2, 3$. Therefore, $f(OPT - o_1 - o_2 - o_3 \mid Y) \geq 0.567v - f(Y)$ holds from submodularity and the fact that $f(OPT \mid Y) \geq v - f(Y)$.

Packing the Remaining Space Let $Y$ be a set found by ModifiedSimple($I; v$). After taking $Y$, we consider the problem (20) to fill in the remaining space. We approximate $OPT - o_1$, $OPT - o_1 - o_2$, and $OPT - o_1 - o_2 - o_3$, respectively, depending on the size $c(Y)$ of $Y$. Recall that $c(Y) < 0.7K$. 

 Springer
Case 1: \((1 - \bar{c}_2)K > c(Y) \geq (1 - \bar{c}_1)K\). By Lemma 21, it holds that

\[ f(\text{OPT} - o_1 \mid Y) \geq 0.693v - f(Y). \]  

(32)

Let \(v' = 0.693v - f(Y)\). Define \(g(\cdot) = f(\cdot \mid Y)\). Consider the problem (20) to approximate \(\text{OPT} - o_1\). We set \(W' = (1 - \bar{c}_1)K\) and \(K' = K - c(Y)\).

If we can find a set \(\tilde{S}\) such that \(c(\tilde{S}) \leq K - c(Y)\) and \(g(\tilde{S}) \geq \kappa v'\), then \(Y \cup \tilde{S}\) is a feasible set to the original instance, and it holds by Lemma 19 and (32) that

\[ f(Y \cup \tilde{S}) \geq (1 - e^{-y})v + \kappa \left(0.693 - (1 - e^{-y})\right) v - O(\varepsilon)v, \]  

(33)

where \(v = c(Y) / K\).

We shall use Lemma 13 to find such a set \(\tilde{S}\). Since \(0.3 \leq \xi_1\) and \(y \leq 0.7\), the ratio \(\eta\) of \(W'\) and \(K'\) is

\[ \eta = \frac{W'}{K'} = \frac{1 - \xi_1}{1 - y} \leq \frac{0.7}{0.3} \leq 2.5. \]

(i) \(\eta \in [2, 2.5]\) In this case, we see that \(\eta \geq 2\) if and only if \(y \geq \frac{1 + \xi_1}{2} \geq 0.65\), since \(\xi_1 \geq 0.3\). It follows from Lemma 13(d) that we can find a set \(\tilde{S}\) such that \(c(\tilde{S}) \leq K - c(Y)\) and \(g(\tilde{S}) \geq 0.178v'\). Hence, since \(y \geq 0.65\), (33) implies that

\[ f(Y \cup \tilde{S}) \geq (1 - e^{-y})v + 0.178 \cdot (0.693v - (1 - e^{-y})v) - O(\varepsilon)v \geq (0.51 - O(\varepsilon))v. \]

(ii) \(\eta \in [1.5, 2]\) We see that \(\eta \geq 1.5\) if and only if \(y \geq \frac{0.5 + \xi_1}{1.5} = \frac{1 + 2\xi_1}{3}\). Also, since \(c(Y) \geq (1 - \bar{c}_1)K\), we have

\[ y \geq \max\left\{ \frac{1 + 2\xi_1}{3}, 1 - \bar{c}_1 \right\} \geq 0.6 - O(\varepsilon), \]

where the lower bound is achieved when both the terms are equal. It follows from Lemma 13(e) that we can find a set \(\tilde{S}\) such that \(c(\tilde{S}) \leq K - c(Y)\) and \(g(\tilde{S}) \geq 0.218v'\). Hence, by (33), we obtain

\[ f(Y \cup \tilde{S}) \geq (1 - e^{-y})v + 0.218 \cdot (0.693v - (1 - e^{-y})v) - O(\varepsilon)v \geq (0.50 - O(\varepsilon))v, \]

as \(y \geq 0.6 - O(\varepsilon)\).

(iii) \(\eta \in [1.4, 1.5]\) It means that \(y \geq \frac{0.4 + \xi_1}{1.4} = \frac{2 + 5\xi_1}{7}\). Also, since \(c(Y) \geq (1 - \bar{c}_1)K\), we have

\[ y \geq \max\left\{ \frac{2 + 5\xi_1}{7}, 1 - \bar{c}_1 \right\} \geq \frac{7}{12} - O(\varepsilon). \]

It follows from Lemma 13(b) that we can find a set \(\tilde{S}\) such that \(c(\tilde{S}) \leq K - c(Y)\) and \(g(\tilde{S}) \geq 0.283v'\). Hence, by (33), we obtain

\[ f(Y \cup \tilde{S}) \geq (1 - e^{-y})v + 0.283 \cdot (0.693v - (1 - e^{-y})v) - O(\varepsilon)v \geq (0.51 - O(\varepsilon))v, \]

as \(y \geq 7/12 - O(\varepsilon)\).
(iv) $\eta \in [1, 1.4]$ It follows from Lemma 13(a) that we can find a set $\tilde{S}$ such that $c(\tilde{S}) \leq K - c(Y)$ and $g(\tilde{S}) \geq 0.315v'$. Hence, by (33), we obtain

$$f(Y \cup \tilde{S}) \geq (1 - e^{-\eta})v + 0.315 \cdot (0.693v - (1 - e^{-\eta})v) - O(\varepsilon)v.$$ 

This is at least $(0.5 - O(\varepsilon))v$ if $y \geq 0.53$. Thus we may suppose that $c(Y) < 0.53K$. Since $c(Y) \geq (1 - \bar{c}_1)K$, we see $\bar{c}_1 \geq 1 - 0.53 = 0.47$. Moreover, since $2.4(1 - \bar{c}_1 - \bar{c}_2) > (1 - \bar{c}_1)$ by the assumption, we have $\bar{c}_2 \leq 0.31$. Hence we have that

$$1 - c_2 \geq 1 - \bar{c}_1 1 - \bar{c}_2 \geq (1 - \delta) \frac{0.69}{0.5} \geq 1.38(1 - \delta),$$

as $\bar{c}_1 \leq 1 - \varepsilon/\delta$. Therefore, by Corollary 6, we can find a $(0.5 - O(\varepsilon))$-approximation.

(v) $\eta \in [0, 1]$ It follows from Lemmas 11 and 12 that we can find a set $\tilde{S}$ such that $c(\tilde{S}) \leq K - c(Y)$ and $g(\tilde{S}) \geq 0.46v'$. By (33), we have

$$f(Y \cup \tilde{S}) \geq (1 - e^{-\eta})v + 0.46 \cdot (0.693v - (1 - e^{-\eta})v) - O(\varepsilon)v \geq (0.53 - O(\varepsilon))v,$$

since $y \geq 0.5$.

Therefore, in each case, the algorithm in Lemma 13 yields a $(0.5 - O(\varepsilon))$-approximation. The space required is $O(K/e^{3}\log K)$ and the total running time is $O(ne^{-4}\log K)$. Thus Lemma 18 holds for Case 1.

Case 2: $(1 - \bar{c}_3)K > c(Y) \geq (1 - \bar{c}_2)K$. Recall that $c(Y) \leq 0.7K$. Since $c(Y) \geq (1 - \bar{c}_2)K$, we have $0.3 \leq \bar{c}_2$. Also $c(Y) \geq (1 - \bar{c}_2)K \geq 0.63K$ holds since $\bar{c}_2 \leq 0.37$.

Define $g(\cdot) = f(\cdot | Y)$, and consider the problem (20) to approximate $\text{OPT} - o_1 - o_2$. By Lemma 21, it holds that

$$g(\text{OPT} - o_1 - o_2) \geq 0.54v - f(Y).$$

Let $v' = 0.54v - f(Y)$. In a way similar to Case 1, if we can find a set $\tilde{S}$ such that $c(\tilde{S}) \leq K - c(Y)$ and $g(\tilde{S}) \geq \kappa v'$, then $Y \cup \tilde{S}$ is a feasible set to the original instance, and it holds by Lemma 19 that

$$f(Y \cup \tilde{S}) \geq (1 - e^{-\eta})v + \kappa \left(0.54 - (1 - e^{-\eta})\right)v - O(\varepsilon)v. \tag{34}$$

We denote $W' = (1 - \xi_1 - \xi_2)K$, $K' = K - c(Y)$, and $y = c(Y)/K$. Since $y \leq 0.7$ and $\xi_1 + \xi_2 \geq (1 - \epsilon)(\bar{c}_1 + \bar{c}_2) \geq 0.6(1 - \epsilon)$, it holds that

$$\eta = \frac{W'}{K'} \leq \frac{1 - \xi_1 - \xi_2}{1 - y} \leq \frac{4}{3} + 2\varepsilon \leq 1.4,$$

where the last inequality follows because we may suppose that $\varepsilon$ is small.

(i) $\eta > 1$ In this case, it holds that $y \geq \xi_1 + \xi_2$. Since $y \geq 1 - \bar{c}_2$, we have

$$y \geq \max(\xi_1 + \xi_2, 1 - \bar{c}_2) \geq \frac{2}{3} - O(\varepsilon).$$

By Lemma 13, we can find a set $\tilde{S}$ such that $c(\tilde{S}) \leq K - c(Y)$ and $g(\tilde{S}) \geq 0.315v'$. Hence, by (34), we obtain

$$f(Y \cup \tilde{S}) \geq (1 - e^{-\eta})v + 0.315 \cdot (0.54v - (1 - e^{-\eta})v) - O(\varepsilon)v \geq (0.50 - O(\varepsilon))v$$

when $y \geq 2/3 - O(\varepsilon)$. 
It follows from Lemmas 11 and 12 that we can find a set $\tilde{S}$ such that $c(\tilde{S}) \leq K - c(Y)$ and $g(\tilde{S}) \geq 0.46v'$. By (34), we have
\[
f(Y \cup \tilde{S}) \geq (1 - e^{-y})v + 0.46 \cdot (0.54v - (1 - e^{-y})v) - O(\varepsilon)v \geq (0.51 - O(\varepsilon))v,
\]
since $y \geq 0.63$.

Therefore, in each case, the algorithm in Lemma 13 yields a $(0.5 - O(\varepsilon))$-approximation. The space required is $O(K\varepsilon^{-3} \log K)$ and the total running time is $O(n\varepsilon^{-4} \log K)$. Thus Lemma 18 holds for Case 2.

**Case 3: $c(Y) \geq (1 - \tilde{c}_3)K$.** In this case, we may assume that $\tilde{c}_3 \geq 0.3$ since $c(Y) \leq 0.7K$, and hence $\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3 \geq 0.9$.

Define $g(\cdot) = f(\cdot | Y)$, and consider the problem (20) to approximate $\text{OPT} - o_1 - o_2 - o_3$. By Lemma 21, it holds that
\[
g(\text{OPT} - o_1 - o_2 - o_3) \geq 0.567v - f(Y).
\]
Let $v' = 0.567v - f(Y)$. We set $W' = (1 - \tilde{c}_1 - \tilde{c}_2 - \tilde{c}_3)K$ and $K' = K - c(Y)$. Then, since $\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3 \geq 0.9(1 - \varepsilon)$, we have $W' \leq (0.1 + 0.9\varepsilon)K$. In addition, since $c(Y) \leq 0.7K$, we see $K' \geq 0.3K$. Since $W' \leq K'$, the algorithm in Section 4 is applicable, and we can find a set $\tilde{S}$ such that $c(\tilde{S}) \leq K - c(Y)$ and $g(\tilde{S}) \geq 0.46v'$. Since $y = c(Y)/K \geq 2/3$, we obtain by Lemma 19
\[
f(Y \cup \tilde{S}) \geq (1 - e^{-y})v + 0.46 \cdot (0.567v - (1 - e^{-y})v) - O(\varepsilon)v \geq (0.52 - O(\varepsilon))v.
\]

Therefore, since the algorithm in Section 4 runs in $O(K\varepsilon^{-3} \log K)$ space and $O(n\varepsilon^{-4} \log K)$ time, provided the approximated optimal value, Lemma 18 holds for Case 3.

### 5.4 Proof of Lemma 13

In this subsection, we prove Lemma 13. Recall that $W' = \eta K'$ for some $\eta > 1$ and that $c(e) \leq K'$ for any $e \in X$. Note that Simple would work even if $\eta \geq 1$, and, by Corollary 1, Simple can find a set $S$ such that
\[
f(S) \geq \left(1 - e^{-1-c_1} - O(\varepsilon)\right)v. \tag{35}
\]

Moreover, when $\eta \leq 1$, we can obtain a $(0.46 - O(\varepsilon))$-approximate solution by LargeFirst in Section 4. This algorithm runs in $O(K\varepsilon^{-3} \log K')$ space and $O(n\varepsilon^{-4} \log K')$ time using $O(\varepsilon^{-1})$ passes, provided the approximated optimal value $v$.

**a) $\eta \in [1,1.4]$** If there exists an item $e$ such that $f(e) \geq 0.315v$, then taking a singleton with maximum return admits a 0.315-approximation. Thus we may assume that $f(e) \leq 0.315v$ for any item $e \in E$. If $c_1 \leq \eta - 1$, then the set $S$ in (35) satisfies that
\[
f(S) \geq \left(1 - e^{-1-c_1} - O(\varepsilon)\right)v \geq \left(1 - e^{-3} - O(\varepsilon)\right)v \geq (0.348 - O(\varepsilon))v.
\]

Otherwise, we consider approximating $\text{OPT} - o_1$. Since $c(\text{OPT} - o_1) \leq K' - (\eta - 1)K' \leq \eta K' = W'$, we can use a $(0.46 - O(\varepsilon))$-approximation algorithm in
Section 4. Since $f(\text{OPT} - o_1) \geq v - f(o_1) \geq 0.685v$, we can find a set $S$ such that 
\[
    f(S) \geq 0.685(0.46 - O(\varepsilon))v \geq (0.315 - O(\varepsilon))v.
\]
Thus the statement holds.

(b) $\eta \in [1.4, 1.5]$ The proof is similar to (a). We may assume that $f(e) \leq 0.283v$ for any item $e \in E$. If $c_1 \leq 1 - \eta$, then the set $S$ in (35) satisfies that 
\[
    f(S) \geq \left(1 - e^{-\frac{1-c_1}{\eta}} - O(\varepsilon)\right)v \geq \left(1 - e^{-\frac{1}{3}} - O(\varepsilon)\right)v \geq (0.283 - O(\varepsilon))v.
\]
Otherwise, apply a $(0.46 - O(\varepsilon))$-approximation algorithm to approximate $\text{OPT} - o_1$. Since $f(\text{OPT} - o_1) \geq v - f(o_1) \geq 0.717v$, the ratio of the output $S$ is 
\[
    f(S) \geq 0.717(0.46 - O(\varepsilon))v \geq (0.329 - O(\varepsilon))v.
\]
Thus the statement holds.

(c) $\eta \in [1.5, 2]$ We will use the above argument in (b) recursively. We may assume that $f(e) < 0.22v$ for any $e \in E$. If $c_1 < 0.5$, then the set $S$ in (35) satisfies that 
\[
    f(S) \geq \left(1 - e^{-\frac{1-c_1}{\eta}} - O(\varepsilon)\right)v \geq \left(1 - e^{-\frac{0.5}{2}} - O(\varepsilon)\right)v \geq (0.22 - O(\varepsilon))v.
\]
So consider the case when $c_1 \geq 0.5$. In this case, we approximate $\text{OPT} - o_1$. Since $c(\text{OPT} - o_1) \leq 2K' - 0.5K' \leq 1.5K'$ and $f(\text{OPT} - o_1) \geq 0.78v$, the algorithm in (b) can find a set $S$ such that 
\[
    f(S) \geq 0.78(0.28 - O(\varepsilon))v \geq (0.218 - O(\varepsilon))v.
\]
Thus the statement holds.

(d) $\eta \in [2, 2.5]$ We may assume that $f(e) < 0.18v$ for any item $e \in E$. If $c_1 < 0.5$, then the set $S$ in (35) satisfies that 
\[
    f(S) \geq \left(1 - e^{-\frac{1-c_1}{\eta}} - O(\varepsilon)\right)v \geq \left(1 - e^{-\frac{0.5}{2.5}} - O(\varepsilon)\right)v \geq (0.18 - O(\varepsilon))v.
\]
So consider the case when $c_1 \geq 0.5$. In this case, we approximate $\text{OPT} - o_1$. Since $c(\text{OPT} - o_1) \leq 2.5K' - 0.5K' \leq 2.0W'$ and $f(\text{OPT} - o_1) \geq 0.82v$, the algorithm in (c) can find a set $S$ such that 
\[
    f(S) \geq 0.82 \cdot (0.218 - O(\varepsilon))v \geq (0.178 - O(\varepsilon))v.
\]
Thus the statement holds.

**Appendix A: Proof of Lemma 7**

We discuss how to obtain a $(0.5 - O(\varepsilon))$-approximation when $c_1 + c_2$ is almost 1.
Lemma 22 Suppose that \( f(o_1 + o_2) \geq v' \). We can find a set \( S \) using two passes and \( O(\varepsilon^{-1} K) \) space such that \(|S| = 2\) and \( f(S) \geq \left( \frac{2}{3} - \varepsilon \right) v' \).

We begin by reviewing the algorithm\(^3\) in [27].

**Theorem 8** Let \( E_R \subseteq E \) be a subset of the ground set (and we call \( E_R \) red items). Let \( X \subseteq E \) such that \( v \leq f(X) \leq (1 + \varepsilon)v \). Assume that there exists \( x \in X \cap E_R \) such that \( \tau v \leq f(x) \leq \tau v \). Then we can find a set \( Y \subseteq E_R \) of red items, in one pass and \( O(n) \) time, with \(|Y| = O(\log_{1+\varepsilon} \frac{\tau}{3}) \) such that some item \( e^* \in Y \) satisfies \( f(X - x + e^*) \geq (2/3 - O(\varepsilon))v \).

**Proof of Lemma 22** For each \( t = 1, 2, \ldots, K/2, \) define \( E_t = \{ e \in E \mid t \leq c(e) \leq K - t \} \) as the red items. The critical thing to observe is that, if \( t \leq c(o_2) \), we see \( o_1 \in E_t \).

The above observation suggests the following implementation. In the first pass, for each set \( E_t \), apply Theorem 8 to collect a set \( X_t \subseteq E_t \) (apparently we can set \( \tau = 2/3 \) and \( \tau = 1/3 \)). Since \(|X_t| = O(\log_{1+\varepsilon} 2) = O(\varepsilon^{-1})\), it takes \( O(\varepsilon^{-1} K) \) space and \( O(n) \) time in total. Then it follows from Theorem 8 that, for each \( t \) with \( t \leq c(o_2) \), there exists \( e^* \in X_t \) such that \( f(o_2 + e^*) \geq (2/3 - O(\varepsilon))v \) and \( c(e^*) \leq K - c(o_2) \). In the second pass, for each item \( e \) in \( E \), check whether there exists \( e' \) in \( X_{c(e)} \) such that \( c(e + e') \leq K \) and \( f(e + e') \geq (2/3 - O(\varepsilon))v \). It follows that there exists at least one pair of \( e \) and \( e' \) satisfying the condition. The second pass also takes \( O(\varepsilon^{-1} K) \) space as we keep \( X_t \)'s. Since \(|X_t| = O(\varepsilon^{-1})\), the second phase takes \( O(\varepsilon^{-1} n) \) time.

Suppose that \( v \leq f(OPT) \leq (1 + \varepsilon)v \). If \( f(o_1 + o_2) \geq 0.75v \), then we are done using Lemma 22. So assume otherwise, meaning that \( f(OPT - o_1 - o_2) \geq 0.25v \). Notice that we can also assume that \( f(OPT - o_1) \geq 0.5v \). Now consider two possibilities.

**Claim** If \( c_1 \geq 1 - \sqrt{\varepsilon} \), then we can find a set \( S \) in \( O(\varepsilon^{-1}) \) passes and \( O(K) \) space such that \( c(S) \leq K \) and \( f(S) \geq (0.5 - O(\varepsilon))v \).

**Proof** Since \( c_1 \geq 1 - \sqrt{\varepsilon} \), we have \( c(OPT - o_1) \leq \sqrt{\varepsilon} K \). Consider the problem (1) to approximate \( OPT - o_1 \). Then the largest item in \( OPT - o_1 \) is \( c(o_2) \) which is at most \( \sqrt{\varepsilon} K \). By Corollary 1, Simple(\( I; 0.5v, \sqrt{\varepsilon} K \)) can obtain a set \( S \) satisfying that \( f(S) \geq 0.5 \left( 1 - e^{-\frac{1 - \sqrt{\varepsilon}}{\sqrt{\varepsilon}}} - O(\varepsilon) \right) v \geq (0.5 - O(\varepsilon))v \),

where the last inequality follows because \( e^{-\frac{1 - \sqrt{\varepsilon}}{\sqrt{\varepsilon}}} \leq \varepsilon \) when \( \varepsilon \leq 1 \). \( \Box \)

**Claim** If \( c_1 < 1 - \sqrt{\varepsilon} \), then we can find a set \( S \) in \( O(\varepsilon^{-1}) \) passes and \( O(K) \) space such that \( c(S) \leq K \) and \( f(S) \geq (0.5 - O(\varepsilon))v \).

\(^3\)This theorem is essentially a rephrasing of Theorem 5.
Proof Consider the problem:

\[
\text{maximize } f(S) \text{ subject to } c(S) \leq \sqrt{\epsilon} K, \quad S \subseteq E,
\]

to approximate \( \text{OPT} - o_1 - o_2 \). Let \( I' \) be the corresponding instance. Since \( f(\text{OPT} - o_1 - o_2) \geq 0.25v \) and \( c(\text{OPT} - o_1 - o_2) \leq \epsilon K \), Corollary 1 implies that Simple\( (I'; 0.25v, \epsilon K) \) can obtain a set \( Y \) satisfying that

\[
f(Y) \geq 0.25 \left( 1 - e^{-\frac{\sqrt{\epsilon} - \epsilon}{\epsilon}} - O(\epsilon) \right) v \geq (0.25 - O(\epsilon)) v,
\]

since the largest item in \( \text{OPT} - o_1 - o_2 \) has size at most \( \epsilon \). After taking the set \( Y \), we still have space for packing either \( o_1 \) or \( o_2 \), since \( c(Y) \leq \epsilon K < K - c(o_1) \).

Define \( g := f(\cdot | Y) \). If some element \( e \) satisfies \( c(Y) + c(e) \leq K \) and \( f(Y + e) \geq 0.5v \), then we are done. Thus we may assume that no such element exists, implying that \( f(Y + o_\ell) < 0.5v \) for \( \ell = 1, 2 \). Hence it holds that

\[
g(\text{OPT} - o_1) \geq g(\text{OPT}) - g(o_1) \geq (f(\text{OPT}) - f(Y)) - (f(Y + o_1) - f(Y)) \geq 0.5v,
\]

This implies that

\[
g(\text{OPT} - o_1 - o_2) \geq g(\text{OPT} - o_1) - g(o_2) \\
\geq 0.5v - (f(Y + o_1) - f(Y)) \geq f(Y) \geq (0.25 - O(\epsilon)) v.
\]

Consider the problem:

\[
\text{maximize } g(S) \text{ subject to } c(S) \leq K - c(Y), \quad S \subseteq E,
\]

to approximate \( \text{OPT} - o_1 - o_1 \). Denote by \( I'' \) the corresponding instance. Since \( K - c(Y) \geq (1 - \sqrt{\epsilon}) K \) and \( g(\text{OPT} - o_1 - o_2) \geq (0.25 - O(\epsilon)) v \), Corollary 1 implies that Simple\( (I''; (0.25 - O(\epsilon)) v, \epsilon K) \) can obtain a set \( S \) satisfying that

\[
f(S) \geq (0.25 - O(\epsilon)) \left( 1 - e^{-\frac{\sqrt{\epsilon} - \epsilon}{\epsilon}} - O(\epsilon) \right) v \geq (0.25 - O(\epsilon)) v.
\]

Therefore, \( Y \cup S \) satisfies that \( c(Y \cup S) \leq K \) and

\[
f(Y \cup S) = f(Y) + g(S) \geq (0.5 - O(\epsilon)) v.
\]

For a given \( v \), the above can be done in \( O(\epsilon^{-1} K) \) space using \( O(\epsilon^{-1}) \) passes. The total running time is \( O(n \epsilon^{-1}) \). This completes the proof of Lemma 7.

Acknowledgements The authors thank the referees for their valuable comments on this manuscript.

References

1. Ahn, K.J., Guha, S.: Linear programming in the semi-streaming model with application to the maximum matching problem. Inform. Comput. 222, 59–79 (2013). 38th International Colloquium on Automata Languages and Programming (ICALP 2011)
2. Alaluf, N., Ene, A., Feldman, M., Nguyen, H.L., Suh, A.: Optimal streaming algorithms for submodular maximization with cardinality constraints. In: Czumaj, A., Dawar, A., Merelli, E. (eds.) 47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference), LIPIcs, vol. 168, pp. 6:1–6:19 (2020). Schloss Dagstuhl - Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.ICALP.2020.6

3. Alon, N., Gamzu, I., Tennenholz, M.: Optimizing budget allocation among channels and influencers. In: Proceedings of the 21st international conference on world wide web (WWW), pp. 381–388 (2012)

4. Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., Krause, A.: 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), pp. 671–680 (2014)

5. Badanidiyuru, A., Vondrák, J.: Fast algorithms for maximizing submodular functions. In: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1497–1514 (2013)

6. Balkanski, E., Rubinstein, A., Singer, Y.: An exponential speedup in parallel running time for submodular maximization without loss in approximation. In: Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms SODA, pp. 283–302 (2019)

7. Balkanski, E., Singer, Y.: The adaptive complexity of maximizing a submodular function. In: Proceedings of the 50th Annual ACM Symposium on Theory of Computing, STOC 2018, pp. 1138–1151. ACM, New York (2018)

8. Barbosa, R.D.P., Ene, A., Nguyen, H.L., Ward, J.: A new framework for distributed submodular maximization. In: 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pp. 645–654 (2016)

9. Batnari, M., Esfandiari, H., Mirrokni, V.: Almost optimal streaming algorithms for coverage problems. In: Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, pp. 13–23 (2017)

10. 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)

11. Chakrabarti, A., Kale, S.: Submodular maximization meets streaming: matchings, matroids, and more. Math. Program. 154(1-2), 225–247 (2015)

12. 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: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1204–1223 (2017)

13. Chan, T.H.H., Jiang, S.H.C., Tang, Z.G., Wu, X.: Online Submodular Maximization Problem with Vector Packing Constraint. In: Annual European Symposium on Algorithms (ESA), pp. 24:1–24:14 (2017)

14. Chekuri, C., Gupta, S., Quanrud, K.: Streaming algorithms for submodular function maximization. In: Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP), vol. 9134, pp. 318–330 (2015)

15. Chekuri, C., Quanrud, K.: Submodular function maximization in parallel via the multilinear relaxation. In: Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 303–322 (2019)

16. 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)

17. Ene, A., Nguyen, H.L.: A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In: Baier, C., Chatzigiannakis, I., Flocchini, P., Leonardi, S. (eds.) 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), LIPIcs, vol. 132, pp. 53:1–53:12 (2019)

18. Ene, A., Nguyen, H.L.: Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. In: Proceedings of the 30th annual ACM-SIAM symposium on discrete algorithms (SODA), pp. 274–282 (2019)

19. Feldman, M., Karbasi, A., Kazemi, E.: Do less, get more: streaming submodular maximization with subsampling. In: Annual Conference on Neural Information Processing Systems (NeurIPS), pp. 730–740 (2018)

20. Feldman, M., Norouzi-Fard, A., Svensson, O., Zenklusen, R.: The one-way communication complexity of submodular maximization with applications to streaming and robustness. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC), pp. 1363–1374 (2020)

21. Filmus, Y., Ward, J.: A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. SIAM J. Comput. 43(2), 514–542 (2014)
22. Fisher, M.L., Nemhauser, G.L., Wolsey, L.A.: An analysis of approximations for maximizing submodular set functions i. Math. Program., pp. 265–294 (1978)
23. Fisher, M.L., Nemhauser, G.L., Wolsey, L.A.: An analysis of approximations for maximizing submodular set functions ii. Math Prog Study 8, 73–87 (1978)
24. Haba, R., Kazemi, E., Feldman, M., Karbasi, A.: Streaming submodular maximization under a k-set system constraint. In: H.D. III, Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 119, pp. 3939–3949 (2020). PMLR. http://proceedings.mlr.press/v119/haba20a.html
25. Huang, C., Kakimura, N.: Improved streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. Algorithmica 83(3), 879–902 (2021). https://doi.org/10.1007/s00453-020-00786-4
26. Huang, C.C., Kakimura, N., Mauras, S., Yoshida, Y.: Approximability of Monotone Submodular Function Maximization under Cardinality and Matroid Constraints in the Streaming Model, SIAM Journal on Discrete Mathematics, to appear (2021)
27. Huang, C.C., Kakimura, N., Yoshida, Y.: Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. Algorithmica 82(4), 1006–1032 (2020)
28. Kale, S., Tirodkar, S.: Maximum matching in two, three, and a few more passes over graph streams. In: The 20th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX2017) (2017)
29. Kazemi, E., Mitrovic, M., Zadimoghaddam, M., Lattanzi, S., Karbasi, A.: Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In: Proceedings of the 36th International Conference on Machine Learning (ICML), pp. 3311–3320 (2019)
30. Kempe, D., Kleinberg, J., Tardos, É.: 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), pp. 137–146 (2003)
31. 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)
32. Kulik, A., Shachnai, H., Tamir, T.: Maximizing submodular set functions subject to multiple linear constraints. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 545–554 (2013)
33. 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)
34. Lee, J.: Maximum entropy sampling, encyclopedia of environmetrics, vol. 3, pp. 1229–1234. Wiley, New York (2006)
35. Lee, J., Sviridenko, M., Vondrák, J.: Submodular maximization over multiple matroids via generalized exchange properties. Math. Oper. Res. 35(4), 795–806 (2010)
36. Lin, H., Bilmes, J.: 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), pp. 912–920 (2010)
37. Lin, H., Bilmes, J.: 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), pp. 510–520 (2011)
38. McGregor, A.: Finding graph matchings in data streams. In: Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems, APPROX05, pp. 170–181 (2005)
39. McGregor, A.: Graph stream algorithms: a survey. SIGMOD Rec. 43(1), 9–20 (2014)
40. McGregor, A., Yu, H.T.: Better streaming algorithms for the maximum coverage problem. In: International Conference on Database Theory (ICDT) (2017)
41. Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., Krause, A.: Lazier than lazy greedy. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, AAAI2015, pp. 1812–1818. AAAI Press (2015)
42. Mirzasoleiman, B., Jegelka, S., Krause, A.: Streaming non-monotone submodular maximization: personalized video summarization on the fly. In: Proc. Conference on Artificial Intelligence (AAAI) (2018)
43. Norouzi-Fard, A., Tarnawski, J., Mitrovic, S., Zandieh, A., Mousavifar, A., Svensson, O.: Beyond 1/2-approximation for submodular maximization on massive data streams. In: Proceedings of the 35th International Conference on Machine Learning (ICML), pp. 3826–3835 (2018)
44. Nutov, Z., Shoham, E.: Practical budgeted submodular maximization (2020)
45. da Ponte Barbosa, R., Ene, A., Nguyễn, H.L., Ward, J.: The power of randomization: Distributed submodular maximization on massive datasets. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, JMLR workshop and conference proceedings, vol. 37, pp. 1236–1244 (2015). JMLR.org. http://proceedings.mlr.press/v37/barbosa15.html
46. Soma, T., Kakimura, N., Inaba, K., Kawarabayashi, K.: Optimal budget allocation: Theoretical guarantee and efficient algorithm. In: Proceedings of the 31st International Conference on Machine Learning (ICML), pp. 351–359 (2014)
47. Sviridenko, M.: A note on maximizing a submodular set function subject to a knapsack constraint. Oper. Res. Lett. 32(1), 41–43 (2004)
48. Wolsey, L.: Maximising real-valued submodular functions: primal and dual heuristics for location problems. Math. Oper. Res. (1982)
49. Yaroslavtsev, G., Zhou, S., Avdiukhin, D.: “Bring Your Own Greedy”+Max: near-optimal 1/2-approximations for submodular knapsack. In: International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 3263–3274 (2020)
50. Yoshida, Y.: Maximizing a monotone submodular function with a bounded curvature under a knapsack constraint. SIAM J. Discrete Math. 33(3), 1452–1471 (2019)
51. Yu, Q., Xu, E.L., Cui, S.: Streaming algorithms for news and scientific literature recommendation: Submodular maximization with a d-knapsack constraint. IEEE Global Conf Signal Inf Process (2016)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.