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

Chien-Chung Huang, Naonori Kakimura

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

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

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

Abstract

In this paper, we consider the problem of maximizing a monotone submodular function subject to a knapsack constraint in a streaming setting. In such a setting, elements arrive sequentially and at any point in time, and the algorithm can store only a small fraction of the elements that have arrived so far. For the special case that all elements have unit sizes (i.e., the cardinality-constraint case), one can find a $(0.5-\varepsilon)$-approximate solution in $O(K\varepsilon^{-1})$ space, where K is the knapsack capacity (Badanidiyuru *et al.* KDD 2014). The approximation ratio is recently shown to be optimal (Feldman *et al.* STOC 2020). In this work, we propose a $(0.4-\varepsilon)$-approximation algorithm for the knapsack-constrained problem, using space that is a polynomial of K and ε. This improves on the previous best ratio of $0.363-\varepsilon$ with space of the same order. Our algorithm is based on a careful combination of various ideas to transform multiple-pass streaming algorithms into a single-pass one.

1 Introduction

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

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

A set function is *monotone* if $f(S) \leq f(T)$ for any $S \subseteq T$. Submodular functions play a fundamental role in combinatorial optimization, as they capture rank functions of matroids, edge cuts of graphs, and set coverage (to name but a few examples). In addition to their theoretical interests, submodular functions have also attracted much attention from the machine learning community because they can model a variety of such practical problems as online advertising [1, 26, 38], sensor location [27], text summarization [32, 33], and maximum entropy sampling [30].

Many of the abovementioned applications can be formulated as the problem of maximizing a monotone submodular function under a knapsack constraint. In this problem, we are given a monotone submodular function $f : 2^E \to \mathbb{R}_+$, a size function $c : E \to \mathbb{N}$, and an integer $K \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of positive integers. The problem is defined as

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

*A preliminary version appears in *The Algorithms and Data Structures Symposium (WADS) 2019*. The first author is supported by ANR-19-CE48-0016 and ANR-18-CE40-0025-01 from the French National Research Agency (ANR). The second author is supported by JSPS KAKENHI Grant Numbers JP17K00028 and JP18H05291.

Table 1: The knapsack-constrained problem. The algorithms [16, 39] are not for the streaming setting. See also [15, 28].

	approx. ratio	#passes	space	running time
Ours	$2/5 - \varepsilon$	1	$O\left(K\varepsilon^{-4}\log^4 K\right)$	$O\left(n\varepsilon^{-4}\log^4 K\right)$
Huang *et al.* [24]	$4/11 - \varepsilon$	1	$O\left(K\varepsilon^{-4}\log^4 K\right)$	$O\left(n\varepsilon^{-4}\log^4 K\right)$
Yu *et al.* [43]	$1/3 - \varepsilon$	1	$O\left(K\varepsilon^{-1}\log K\right)$	$O\left(n\varepsilon^{-1}\log K\right)$
Huang *et al.* [24]	$2/5 - \varepsilon$	2	$O\left(K\varepsilon^{-4}\log^4 K\right)$	$O\left(n\varepsilon^{-4}\log^4 K\right)$
Huang-Kakimura [23]	$1/2 - \varepsilon$	$O\left(\varepsilon^{-1}\right)$	$O\left(K\varepsilon^{-7}\log^2 K\right)$	$O\left(n\varepsilon^{-8}\log^2 K\right)$
Ene and Nguyễn [16]	$1 - e^{-1} - \varepsilon$	—	—	$O\left((1/\varepsilon)^{O(1/\varepsilon^4)} n \log n\right)$
Sviridenko [39]	$1 - e^{-1}$	—	—	$O\left(Kn^4\right)$

where we denote $c(S) = \sum_{e \in S} c(e)$ for a subset $S \subseteq E$. Note that the size $c(e)$ for each $e \in E$ is a positive integer. When $c(e) = 1$ for every item $e \in E$, the constraint coincides with a cardinality constraint. Hereafter, we assume that every item $e \in E$ satisfies $c(e) \leq K$, as otherwise we can simply discard it.

The problem of maximizing a monotone submodular function under a knapsack constraint or a cardinality constraint is classical and well-studied [21, 41]. The problem is known to be NP-hard but can be approximated within the factor of $1 - e^{-1}$; see, e.g., [3, 15, 22, 28, 39, 42].

In some applications, the amount of input data is much larger than the main memory capacity of individual computers. In such a case, we need to process data in a *streaming* fashion (see e.g., [34]). That is, we consider the situation in which each item in the ground set E arrives sequentially, and we are allowed to keep only a small number of items in the memory at any point. This setting effectively rules out most techniques provided in the literature, as they typically require random access to the data. In this work, we assume that an item can be stored using $O(1)$ space, and that the value oracle of f is available at any point in the process. Such an assumption is standard in the submodular function literature and in the context of a streaming setting [2, 13, 43].

Our main contribution is to propose a single-pass $(2/5 - \varepsilon)$-approximation algorithm for the problem (1), which improves on previous work [24, 43] (see Table 1). Space complexity here is independent of the number of items in E, which is denoted by n.

Theorem 1.1. *There exists a single-pass streaming $(2/5 - \varepsilon)$-approximation algorithm for the problem (1) requiring $O\left(K\varepsilon^{-4}\log^4 K\right)$ space.*

Our Technique.

Let us first describe approximation algorithms for the knapsack-constrained problem (1) in the offline setting. The simplest algorithm is greedy, i.e., it repeatedly takes that item with maximum ratio between its marginal return and its cost. Although the output of such a greedy algorithm does not guarantee any approximation, if we take the better of the output and every singleton, we can obtain a solution with ratio roughly 0.35 [41][1]. Sviridenko [39] showed that, by applying a greedy algorithm from each set of at most three items, we can find a $(1 - 1/e)$-approximate solution. Recently, it is shown in [37] that it suffices to enumerate all the sets of at most two items. To improve running time to nearly linear time, such partial enumeration has been replaced by more

[1]The approximation ratio is recently improved to 0.405 [40].

sophisticated multi-stage guessing strategies (in which fractional items are added on the basis of the technique of multilinear extension) [16]. For implementation, however, all of them require large space and/or a large number of passes.

For a streaming setting, Badanidiyuru *et al.* [2] proposed a single-pass thresholding algorithm that achieves a $(0.5-\varepsilon)$-approximation for the cardinality-constrained problem. The algorithm simply takes an arriving item e if its marginal return exceeds a certain threshold and its addition does not violate the feasibility constraint. This strategy, however, gives us only a $(1/3-\varepsilon)$-approximation for the knapsack-constrained problem. The drop in approximation ratio results from the fact that, under a knapsack constraint, a new item cannot always be added into the current set even if the latter's size is less than K.

To overcome this drawback, in [24] a branching technique is introduced in which one stops at some point of the thresholding algorithm and uses a different strategy to collect subsequent items. The ratio of this branching algorithm depends on the size of the largest item o_1 in the optimal solution; the ratio becomes worse when $c(o_1)$ is overly large. Overall, the proposed approach of [24] gives a $(4/11-\varepsilon)$-approximation.

How, then, might one improve the ratio further when $c(o_1)$ is large? One possible strategy is to find o_1 separately and run the thresholding algorithm to find the rest of the optimal solution $\mathrm{OPT}-o_1$, which is a similar approach to that used in the offline setting. However, finding o_1 is a difficult task in a streaming setting. One can certainly find an item whose size and f-value are close to those of o_1 by guessing the size $c(o_1)$ and the f-value $f(\{o_1\})$. The difficulty lies in how to identify such an item that, *together with the rest* $\mathrm{OPT}-o_1$, will guarantee a decent f-value. That is to say, we need a good substitute for o_1. In [24], a single-pass procedure, called PickOneItem, is designed to find such an item (see Section 2 for details). Their algorithm finds a constant number of items such that at least one among them will be a good substitute for o_1. Once equipped with such an item, it is not difficult to collect other items so as to improve the approximation ratio to $2/5-\varepsilon$. The down-side of this approach is that one needs multiple passes.

In this paper, we introduce new techniques to achieve the same ratio *without* the need to waste a pass in collecting a good substitute for o_1. We create a combination of PickOneItem and the thresholding algorithm in two different ways. The first is to perform both *dynamically*, that is, each time we find a candidate e for an approximation of o_1, we perform the thresholding algorithm starting from e with the current set, where the thresholding algorithm is to find an approximate solution to $\mathrm{OPT}-o_1$. We show that the approximation ratio depends on $c(o_1)$ and $c(o_2)$, where o_2 is the second largest item in OPT, and is at least $2/5-\varepsilon$ when $c(o_2)$ is at most $K/3$. In contrast to this, when $c(o_2)$ is greater than $K/3$, we deal with both o_1 and o_2 separately from the rest of the optimal solution. The second algorithm performs the thresholding algorithm to find an approximate solution to $\mathrm{OPT}-o_1-o_2$, *in parallel* with finding approximations of o_1 and o_2 using PickOneItem. We show that a combination of their results yields a $(2/5-\varepsilon)$-approximate solution when $c(o_2)$ is greater than $K/3$. Details regarding the two algorithms are described in Sections 3.2 and 3.3, respectively.

Related Work.

Maximizing a monotone submodular function subject to various constraints is a subject that has been extensively studied in the literature. We do not attempt to give a complete survey here and merely highlight the most relevant results. In addition to the knapsack constraint and cardinality constraint mentioned above, the problem has also been studied under (multiple) matroid constraints, p-system constraints, multiple knapsack constraints. See [9, 11, 12, 15, 20, 28, 31] and

the references therein.

For a streaming setting, Badanidiyuru *et al.* [2] proposed a single-pass $(0.5-\varepsilon)$-approximation algorithm with $O(K\varepsilon^{-1}\log K)$ space for the cardinality-constrained problem. The space complexity has been improved to $O(K\varepsilon^{-1})$ [25]. Recently, Feldman *et al.* [19] showed that the approximation ratio 0.5 is optimal in the sense that, to achieve an approximation ratio better than $0.5+\varepsilon$, one needs to use space $\Omega(\varepsilon n/K^3)$. Further, single-pass streaming algorithms have also been proposed for the problem with matroid constraints [10], p-matchoid constraints [18], and knapsack constraint [24, 43], and for that without monotonicity [13, 36]. *Multi-pass streaming algorithms*, which are allowed to read a stream of the input multiple times, have also been studied [3, 10, 23, 24]. Particularly notable is that Chakrabarti and Kale [10] gave an $O(\varepsilon^{-3})$-pass streaming algorithms for a generalization of the maximum matching problem and the submodular maximization problem with a cardinality constraint. Huang and Kakimura [23] designed an $O(\varepsilon^{-1})$-pass streaming algorithm with approximation guarantee $1/2-\varepsilon$ for the knapsack-constrained problem. In addition to the streaming setting, recent applications of submodular function maximization to large data sets have motivated researchers to pursue work in new directions on other computational models including such a parallel computation model as the MapReduce model [7, 6, 29] and on adaptivity analysis [4, 5, 14, 17].

The maximum coverage problem is a special case of monotone submodular maximization under a cardinality constraint for which the function is a set-covering function. For the special case, McGregor and Vu [35] and Batani *et al.* [8] gave a $(1-e^{-1}-\varepsilon)$-approximation algorithm in a multi-pass streaming setting.

2 Preliminaries

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

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

In the following sections, we review previous results w.r.t. the thresholding algorithm and the procedure PickOneItem.

2.1 Thresholding Algorithms

In this section, we present a thresholding algorithm with a single pass [2, 24, 43]. The algorithm simply takes an arriving item e when the marginal return exceeds a threshold. That is, when a new item e arrives, we decide to add e to our current set S if $c(S+e)\leq K$ and $f(e\mid S)\geq\alpha\frac{c(e)}{K}v$, where α is a parameter. See Algorithm 1. Performance depends on the following.

Lemma 2.1. *Let $S=\{e_1,e_2,\ldots,e_s\}$. Suppose that $f(e_i\mid\{e_1,e_2,\ldots,e_{i-1}\})\geq\alpha\frac{c(e_i)}{K}v$ for each $i=1,2,\ldots,s$. It holds, then, that*

$$f(S)\geq\alpha\frac{c(S)}{K}v.$$

Algorithm 1 Thresholding Algorithm [2, 24, 43]

procedure Thresholding(v) ▷ v is an approximation of $f(\mathrm{OPT})$
  $S := \emptyset$.
  while item e is arriving **do**
    If $f(e \mid S) \geq \alpha \frac{c(e)}{K} v$ and $c(S+e) \leq K$ **then** $S := S + e$.
  return S

Proof. This is because

$$f(S) = \sum_{i=1}^{s} f(e_i \mid \{e_1, e_2, \ldots, e_{i-1}\}) \geq \alpha \frac{c(S)}{K} v.$$

□ □

By Lemma 2.1, if the thresholding algorithm returns a set S of size K, then $f(S) \geq \alpha v$ holds. For the cardinality-constrained problem, we can see that, if S has a size less than K, then $f(S) \geq (1-\alpha)v$ by submodularity, which implies that, with a setting of $\alpha = 1/2$, the algorithm will find a set S such that $f(S) \geq v/2$ [2]. For the knapsack-constrained problem, setting $\alpha = 2/3$, together with taking, in parallel, a singleton with maximum return, we can find, with a single pass, a set S such that $f(S) \geq v/3$ [24].

2.2 Guessing the Large Item

We here consider a procedure for approximating the largest item o_1 in OPT. It is difficult to correctly identify o_1 among the items in E, but we can nonetheless find a reasonable approximation of it in a single pass. This procedure is used to design multi-pass streaming algorithms [23, 24]. Recall that we are given an approximate value v of $f(\mathrm{OPT})$ such that $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$.

Let us first consider the following:

Lemma 2.2 ([24]). *Let $E_1 \subseteq E$ such that $e^* \in E_1 \cap \mathrm{OPT}$. Let θ be a real number such that $\theta v/(1+\varepsilon) \leq f(e^*) \leq \theta v$. For a nonnegative integer t with $t > \frac{1}{\theta} - 2$, define*

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

Suppose that a set $X = \{e_1, e_2, \ldots, e_x\} \subseteq E_1$ satisfies that $f(e_i \mid \{e_1, e_2, \ldots, e_{i-1}\}) \geq (\theta - \lambda(i-1))v$ for each $i = 1, 2, \ldots, x$. Then the following holds:

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

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

Here $\Gamma : \mathbb{R} \to \mathbb{R}$ is the function defined by

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

Algorithm 2 Procedure to guess one optimal item [24]

procedure PickOneItem(v, θ, E_1) ▷ $\theta v/(1+\varepsilon) \le f(\mathrm{OPT}) \le \theta v$ and $\mathrm{OPT} \cap E_1 \neq \emptyset$
  Define t and λ from θ by Lemma 2.2.
  $X := \emptyset$.
  while item $e \in E_1$ is arriving **do**
    If $|X| < t+1$ and $f(e \mid X) \ge (\theta - \lambda|X|)v$ **then** $X := X + e$.
  return X

This lemma suggests the following procedure, called PickOneItem, for finding an item that resembles o_1, and we describe this in Algorithm 2. Suppose that we are given approximations $\underline{r}_1, \overline{r}_1$ of r_1 such that $\underline{r}_1 \le r_1 \le \overline{r}_1$ and $\overline{r}_1 \le (1+\varepsilon)\underline{r}_1$. Define $E_1 = \{e \in E \mid \underline{r}_1 K \le c(e) \le \overline{r}_1 K, \theta v/(1+\varepsilon) \le f(e) \le \theta v\}$. We see, then, that $o_1 \in E_1$. In a single pass, starting from $X = \emptyset$, we decide to add an item $e \in E_1$ to X if $f(e \mid X) \ge (\theta - \lambda|X|)v$. We stop making this decision when $|X| = t+1$. Then, in each step, X will always satisfy the assumption in Lemma 2.2; that is, $X = \{e_1, e_2, \ldots, e_x\} \subseteq E_1$ will satisfy that $f(e_i \mid \{e_1, e_2, \ldots, e_{i-1}\}) \ge (\theta - \lambda(i-1))v$ for each $i = 1, 2, \ldots, x$.

We are able to demonstrate that the output X contains an item $e \in X$ such that $f(\mathrm{OPT} - o_1 + e) \ge \Gamma(\theta)v - O(\varepsilon)v$. Let us consider the situation just before o_1 arrives. If the current set X has size $t+1$, then Lemma 2.2 (i) implies that there exists $e \in X$ such that $f(\mathrm{OPT} - o_1 + e) \ge \Gamma(\theta)v - O(\varepsilon)v$. If X has a size less than $t+1$, then either o_1 is put in X, or there exists $e \in X$ such that $f(\mathrm{OPT} - o_1 + e) \ge \Gamma(\theta)v - O(\varepsilon)v$ by Lemma 2.2 (ii). Hence, in any case, at least one item $e \in X$ will guarantee that $f(\mathrm{OPT} - o_1 + e) \ge \Gamma(\theta)v - O(\varepsilon)v$.

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

Theorem 2.3 ([24]). *Let $E_1 \subseteq E$ such that $e^* \in E_1 \cap \mathrm{OPT}$. Suppose that we are given a real number θ that satisfies $\theta v/(1+\varepsilon) \le f(e^*) \le \theta v$. Define t to be*

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

Then, with a single pass and $O(1)$ space, we can find a set $X \subseteq E_1$ such that $|X| \le t+1$ and some item $e \in X$ satisfies that $f(\mathrm{OPT} - e^* + e) \ge \Gamma(\theta)v - O(\varepsilon)v$, where

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

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

In this section, we present a single-pass $(2/5 - \varepsilon)$-approximation algorithm for the problem (1). We first show in Section 3.1 that, if $c(o_1)$ is at most $K/2$ or more than $2K/3$, then the algorithm in [24] can be used. We then focus on the case in which $c(o_1)$ is in $[K/2, 2K/3]$. For this case, we develop two algorithms by combining the technique used in Section 2.2 into the thresholding

algorithm seen in Section 2.1. The first algorithm, which is presented in Section 3.2, is useful when $c(o_2)$ is at most $K/3$, while the second in Section 3.3 is applied when $c(o_2)$ is more than $K/3$.

Hereafter, we often assume that we know in advance approximations of $r_1 = c(o_1)/K$ and $r_2 = c(o_2)/K$. That is, we are given $\underline{r}_\ell$, $\overline{r}_\ell$ such that $\underline{r}_\ell \le r_\ell \le \overline{r}_\ell$ and $\overline{r}_\ell \le (1+\varepsilon)\underline{r}_\ell$ for $\ell \in \{1, 2\}$. These values can be guessed from a geometric series of a certain interval; this will be described in greater detail for each algorithm.

3.1 Algorithm When $c(o_1) \le \frac{K}{2}$ or $c(o_1) > \frac{2K}{3}$

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

Theorem 3.1 ([24]). *Suppose that $c(o_1) \le K/2$. We can find a $(2/5-\varepsilon)$-approximate solution with a single pass for the problem (1). The space complexity of the algorithm is $O(K\varepsilon^{-4}\log^4 K)$.*

The algorithm for the above theorem can be extended for the problem of finding a set S of items that maximizes $f(S)$ subject to the relaxed constraint that the total size is at most pK, for a given number $p \ge 1$. Note here that a set S of items is a *(p,α)-approximate solution* if $c(S) \le pK$ and $f(S) \ge \alpha f(\mathrm{OPT})$, where OPT is an optimal solution of the original instance.

Theorem 3.2 ([24]). *For a constant number $p \ge 2r_1$, there exists a $\left(p, \frac{2p}{2p+3} - \varepsilon\right)$-approximation single-pass streaming algorithm. The space complexity of the algorithm is $O(K\varepsilon^{-3}\log^3 K)$.*

For example, when we are allowed to pack items up to $2K$ (i.e., when $p = 2$), the above algorithm achieves a $(2, 4/7-\varepsilon)$-approximation.

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

Corollary 3.3. *If $c(o_1) > 2K/3$, then we can find a $(2/5-\varepsilon)$-approximate solution with a single pass. The space complexity of the algorithm is $O(K\varepsilon^{-3}\log^3 K)$.*

Proof. Suppose that $c(o_1) > 2K/3$. We may assume that $f(o_1) < \frac{2}{5} f(\mathrm{OPT})$, as otherwise taking a singleton with maximum return would give a 2/5-approximation. We can see then that $f(\mathrm{OPT} - o_1) \ge f(\mathrm{OPT}) - f(o_1) > \frac{3}{5} f(\mathrm{OPT})$ and $c(\mathrm{OPT} - o_1) \le K - c(o_1) < K/3$. Consider maximizing $f(S)$ subject to $c(S) \le K/3$ in the set $\{e \in E \mid c(e) \le K/3\}$. The optimal value of this instance will be at least $f(\mathrm{OPT} - o_1) > \frac{3}{5} f(\mathrm{OPT})$, as $\mathrm{OPT} - o_1$ will be feasible for this instance. We can now apply Theorem 3.2 with $p = 3$ to this instance. The output S then has a size of at most K, and moreover, we have

$$f(S) \ge \left(\frac{2}{3} - \varepsilon\right)\frac{3}{5} f(\mathrm{OPT}) \ge \left(\frac{2}{5} - O(\varepsilon)\right) f(\mathrm{OPT}).$$

Thus we obtain a $(2/5 - O(\varepsilon))$-approximation. □ □

Corollary 3.4. *Suppose that $c(o_1) > K/2$. If $f(o_1) \le \frac{3}{10} f(\mathrm{OPT})$, then we can find a $(2/5-\varepsilon)$-approximate solution with a single pass. The space complexity of the algorithm is $O(K\varepsilon^{-3}\log^3 K)$.*

Proof. We observe that $f(\mathrm{OPT}-o_1) \geq f(\mathrm{OPT})-f(o_1) \geq \frac{7}{10}f(\mathrm{OPT})$ and $c(\mathrm{OPT}-o_1) \leq K-c(o_1) < K/2$. Consider maximizing $f(S)$ subject to $c(S) \leq K/2$ in the set $\{e \in E \mid c(e) \leq K/2\}$. The optimal value of this instance is at least $f(\mathrm{OPT}-o_1) \geq \frac{7}{10}f(\mathrm{OPT})$. We now apply Theorem 3.2 with $p=2$ to this instance. The output S then has a size of at most K, and, further, we have

$$f(S) \geq \left(\frac{4}{7}-\varepsilon\right)\frac{7}{10}f(\mathrm{OPT}) \geq \left(\frac{2}{5}-O(\varepsilon)\right)f(\mathrm{OPT}).$$

Thus we obtain a $(2/5-O(\varepsilon))$-approximation. □ □

3.2 Algorithm for Small $c(o_2)$

By Theorem 3.1 and Corollary 3.3, we may assume that $c(o_1)$ is in $[K/2, 2K/3]$. In this section, we describe a single-pass algorithm that works well when $c(o_2)$ is small. More specifically, we present the following theorem.

Theorem 3.5. *If $c(o_1) \in [K/2, 2K/3]$ and $c(o_2) \leq K/3$, then we can find a $(2/5-\varepsilon)$-approximate solution with a single pass. The space complexity is $O(K\varepsilon^{-4}\log^2 K)$.*

To prove the theorem, we first give an algorithm provided with an approximation v of the optimal value in Section 3.2.1, and then eliminate the assumption in Section 3.2.2.

3.2.1 Algorithm with Optimal Value

In this section, we suppose that we know in advance an approximate value v of $f(\mathrm{OPT})$, i.e., $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$. This assumption can be eliminated with a dynamic update technique using $O(\varepsilon^{-1}\log K)$ additional space in a way similar to that of [2, 24, 43], which will be discussed later in Section 3.2.2. In addition, we suppose that we are given θ_1 such that $\theta_1 v/(1+\varepsilon) \leq f(o_1) \leq \theta_1 v$. Define $E_1 = \{e \in E \mid c(e) \in [\underline{r}_1 K, \overline{r}_1 K], f(e) \in [\theta_1 v/(1+\varepsilon), \theta_1 v]\}$. We can assume that E is the disjoint union of E_1 and $\overline{E_1} = \{e \mid c(e) \leq \overline{r}_2 K\}$, as we can discard the other items. Note that $o_1 \in E_1$ and $\mathrm{OPT}-o_1 \subseteq \overline{E_1}$.

We propose a single-pass streaming algorithm, called $\mathsf{Dynamic}(v)$. The algorithm description is given in Algorithm 3.

In the algorithm $\mathsf{Dynamic}(v)$, we basically run the thresholding algorithm for $\overline{E_1}$ to collect a set S of items. In the same pass in parallel, we try to find a subset $X \subseteq E_1$ that contains a good approximation of o_1, on the basis of Lemma 2.2. That is, when an item e in E_1 arrives, we add e to X if $|X| < t+1$ and $f(e \mid X) \geq (\theta_1 - \lambda|X|)v$. Each time an item e is added to X, since e may be a good approximation of o_1, we create a new feasible set $S+e$, and start to run the thresholding algorithm, in parallel, from $S+e$. Thus each item e in X will generate a feasible set, and the family of these feasible sets will be maintained as $\mathcal{T}$ in the algorithm. Note that, in order to guarantee the approximation ratio of the algorithm starting from $S+e$, the thresholding condition needs to be satisfied for e in X: $f(e \mid S) \geq \alpha\frac{c(e)}{K}v$ for the current set S. Thus the above algorithm *dynamically* performs the thresholding algorithm to $\overline{E_1}$ and $\overline{E_1}+e$ for each $e \in X$.

The above strategy does not work, however, when the size of S becomes large. Indeed, as we perform the thresholding algorithm to $S+e$ for each $e \in X$, it is necessary for $S+e$ to be feasible, i.e., for $c(S) \leq K-c(e)$ when e arrives. Further, since we have the additional condition $f(e \mid S) \geq \alpha\frac{c(e)}{K}v$ for picking an item for X, as noted above, we may discard an approximation of o_1 when $f(e \mid S)$ is small (even if Lemma 2.2 is applicable). To avoid such issues, we adopt another

Algorithm 3

1: **procedure** Dynamic(v)
2: $S := \emptyset$; $S'_0 := \emptyset$; $\mathcal{T} := \emptyset$; $X := \emptyset$.
3: $\alpha := \frac{2}{5(1-\overline{r}_2)}$.
4: Define t and λ from θ_1 by (4) and (2).
5: **while** item e is arriving **do** ▷ First phase
6: **if** $e \in E_1$ **then**
7: **if** $|X| < t+1$ and $f(e \mid X) \geq (\theta_1 - \lambda|X|)v$ and $f(e \mid S) \geq \alpha\frac{c(e)}{K}v$ **then**
8: $\mathcal{T} := \mathcal{T} \cup \{S+e\}$ and $X := X + e$.
9: **else**
10: **if** $f(e \mid S) \geq \alpha\frac{c(e)}{K}v$ and $c(S+e) \leq K$ **then** $S := S + e$.
11: **for** each $T \in \mathcal{T}$ **do**
12: **if** $f(e \mid T) \geq \alpha\frac{c(e)}{K}v$ and $c(T+e) \leq K$ **then** $\mathcal{T} := \mathcal{T} \setminus \{T\} \cup \{T+e\}$.
13: **if** $c(S) \geq (1-\overline{r}_1-\overline{r}_2)K$ **then** $S'_0 := S$ **and break**.
14: $S' := S'_0$.
15: **while** item e is arriving **do** ▷ Second phase
16: **if** $e \in E_1$ **then**
17: **if** $f(S') < f(S'_0 + e)$ **then** $S' := S'_0 + e$.
18: **else**
19: **if** $f(e \mid S) \geq \alpha\frac{c(e)}{K}v$ and $c(S+e) \leq K$ **then** $S := S + e$.
20: **for** any $T \in \mathcal{T}$ **do**
21: **if** $f(e \mid T) \geq \alpha\frac{c(e)}{K}v$ and $c(T+e) \leq K$ **then** $\mathcal{T} := \mathcal{T} \setminus \{T\} \cup \{T+e\}$.
return the best among $\{S, S'\} \cup \mathcal{T}$.

strategy when $c(S)$ becomes large. Let S'_0 be the set we have the first time that $c(S)$ is at least $(1-\overline{r}_1-\overline{r}_2)K$. It follows from Lemma 2.1 that $f(S'_0)$ will be relatively large, as indicated in (6) below. Additionally, since $c(S'_0)$ will be at most $(1-\overline{r}_1)K$, we can add any item in E_1 to S'_0. In the remainder of a stream after having S'_0, we simply take one item $e \in E_1$ that maximizes $f(S'_0+e)$. At the same time, we continue to run the thresholding algorithms to S and every set in $\mathcal{T}$. In the end, the algorithm returns the best among all candidates.

Theorem 3.6. *Suppose that $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$. Then Algorithm* Dynamic(v) *returns a set S such that $c(S) \leq K$ and*

$$f(S) \geq \min\left\{\frac{2}{5},\ 1 - \frac{2}{5(1-\overline{r}_2)},\ \Gamma(\theta) - \frac{2}{5}\left(\frac{1-r_1}{1-\overline{r}_2}\right)\right\} v - O(\varepsilon)v. \tag{5}$$

The space complexity is $O(K)$.

Let $\tilde{S}$ be the set S at the end of the algorithm, and $\tilde{S}'$ be the output obtained by adding one item in E_1 to S'_0 (Line 17). Similarly, we denote by $\tilde{T}_e$ the set of $\mathcal{T}$ containing an item $e \in X$ at the end. Note that the sets $\tilde{S}$ and $\tilde{T}_e$ are obtained by adding an item satisfying the thresholding condition repeatedly. Also, $\tilde{S}'$ and $\tilde{T}_e$ contain exactly one item e in E_1.

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

Lemma 3.7. *It holds that $c(\tilde{S}) \leq K$, $c(\tilde{S}') \leq K$, and $c(\tilde{T}_e) \leq K$ for each $e \in X$.*

Proof. We add an item e only when the addition does not exceed the knapsack capacity K, except for Lines 8 and 17. In Line 8, since $c(S) < (1 - \overline{r}_1 - \overline{r}_2)K$, we have $c(S + e) \leq (1 - \overline{r}_2)K \leq K$ for $e \in E_1$. In Line 17, since $c(S'_0) < (1 - \overline{r}_1)K$, we can add an item in E_1. □ □

In the remainder of this subsection, we show (5) in Theorem 3.6. We first demonstrate that, by Lemma 2.1, we have

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

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

Lemma 3.8. *At the end of the algorithm, one of the following holds.*

- *There exists an item $e \in \tilde{X}$ such that $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$.*
- *$f(o_1 \mid \tilde{S}) < \alpha\overline{r}_1 v$, or*
- *$f(\tilde{S}') \geq 2v/5$.*

Proof. Suppose that o_1 arrives during the first while-loop. Note that this situation includes the case in which the algorithm ends in the first while-loop without creating S'_0. Let $X = \{e_1, e_2, \ldots, e_x\}$ be the set just before o_1 arrives such that items are sorted in the ordering of the addition. Then X satisfies $f(e_i \mid \{e_1, e_2, \ldots, e_{i-1}\}) \geq (\theta - \lambda(i-1))v$ for each $i = 1, 2, \ldots, x$. Note that, when o_1 will be added to X, clearly the first statement will hold. Thus we may assume that o_1 does not satisfy the condition in Line 7; that is, one of the following three conditions holds: $|X| = t + 1$, $f(o_1 \mid X) < (\theta - \lambda|X|)v$, or $f(o_1 \mid S) < \alpha c(o_1)v \leq \alpha\overline{r}_1 v$. It follows from Lemma 2.2 that, if one of the first two conditions holds, then at least one item $\overline{e} \in X$ will satisfy $f(\mathrm{OPT} - o_1 + \overline{e}) \geq \Gamma(\theta)v - O(\varepsilon)v$. If $f(o_1 \mid S) < \alpha\overline{r}_1 v$, then $f(o_1 \mid \tilde{S}) \leq f(o_1 \mid S) < \alpha\overline{r}_1 v$, by submodularity. Thus, since $X \subseteq \tilde{X}$, one of the first two statements of Lemma 3.8 will be satisfied.

Next suppose that S'_0 is constructed and o_1 arrives after that. We may assume that $f(\tilde{S}') < 2v/5$. From Line 17, we see that $f(S'_0 + o_1) \leq f(\tilde{S}') < 2v/5$. Hence we have

$$f(o_1 \mid S'_0) = f(S'_0 + o_1) - f(S'_0) < \frac{2}{5}v - f(S'_0).$$

By (6), it holds that

$$f(o_1 \mid S'_0) < \frac{2}{5}v - \alpha(1 - \overline{r}_1 - \overline{r}_2)v \leq \alpha\overline{r}_1 v,$$

where, we may recall, $\alpha = \frac{2}{5(1-\overline{r}_2)}$. Therefore, by submodularity, we obtain $f(o_1 \mid \tilde{S}) \leq f(o_1 \mid S'_0) < \alpha\overline{r}_1 v$. Thus the lemma follows. □ □

Let us next consider each case of Lemma 3.8.

Lemma 3.9. *Suppose that there exists $e \in \tilde{X}$ such that $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$. It holds, then, that*

$$f(\tilde{T}_e) \geq \min\left\{\frac{2}{5},\ \Gamma(\theta) - \frac{2}{5}\left(\frac{1 - r_1}{1 - \overline{r}_2}\right)\right\}v - O(\varepsilon)v.$$

Proof. If $c(\tilde{T}_e) \geq (1-\overline{r}_2)K$, then we obtain $f(\tilde{T}_e) \geq 2v/5$ by Lemma 2.1. Thus we may assume that $c(\tilde{T}_e) < (1-\overline{r}_2)K$. This implies that, during the algorithm, no items in $\mathrm{OPT} - o_1 - \tilde{T}_e$ are included in $\tilde{T}_e$ due to the thresholding condition, not the capacity constraint. Hence, for each item $o \in \mathrm{OPT} - o_1 - \tilde{T}_e$, we have $f(o \mid \tilde{T}_e) < \alpha c(o)v/K$. Therefore, it holds by monotonicity and submodularity that

$$\begin{aligned} f(\mathrm{OPT} - o_1 + e) &\leq f((\mathrm{OPT} - o_1 + e) \cup \tilde{T}_e) \leq f(\tilde{T}_e) + f(\mathrm{OPT} - o_1 - \tilde{T}_e \mid \tilde{T}_e) \\ &\leq f(\tilde{T}_e) + \sum_{o \in \mathrm{OPT} - o_1 - \tilde{T}_e} f(o \mid \tilde{T}_e) \\ &\leq f(\tilde{T}_e) + \alpha \frac{c(\mathrm{OPT} - o_1 - \tilde{T}_e)}{K} v. \end{aligned}$$

Since $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$ and $c(\mathrm{OPT} - o_1 - \tilde{T}_e) \leq (1-r_1)K$, we obtain

$$\Gamma(\theta)v - O(\varepsilon)v \leq f(\tilde{T}_e) + \alpha(1-r_1)v.$$

Thus the lemma follows, as $\alpha = \frac{2}{5(1-\overline{r}_2)}$. □ □

Lemma 3.10. *If $f(o_1 \mid \tilde{S}) < \alpha \overline{r}_1 v$, then we have*

$$f(\tilde{S}) \geq \min\left\{\frac{2}{5},\ 1-\alpha\right\} v - O(\varepsilon)v.$$

Proof. If $c(\tilde{S}) \geq (1-\overline{r}_2)K$, then $f(\tilde{S}) \geq 2v/5$ by Lemma 2.1. Thus we may assume that $c(\tilde{S}) < (1-\overline{r}_2)K$. In this case, during the algorithm, all items in $\mathrm{OPT} - o_1 - \tilde{S}$ are not included in $\tilde{S}$ due to the thresholding condition, not the capacity constraint. Hence, for each item $o \in \mathrm{OPT} - o_1 - \tilde{S}$, we have $f(o \mid \tilde{S}) < \alpha c(o)v/K$. This implies that

$$\begin{aligned} f(\mathrm{OPT}) - f(\tilde{S}) = f(\mathrm{OPT} - \tilde{S} \mid \tilde{S}) &\leq f(o_1 \mid \tilde{S}) + f(\mathrm{OPT} - o_1 - \tilde{S} \mid \tilde{S}) \\ &\leq \alpha \overline{r}_1 v + \sum_{o \in \mathrm{OPT} - o_1 - \tilde{S}} f(o \mid \tilde{S}) \\ &\leq \alpha \overline{r}_1 v + \alpha \frac{c(\mathrm{OPT} - o_1 - \tilde{S})}{K} v. \end{aligned}$$

Since $c(\mathrm{OPT} - o_1 - \tilde{S}) \leq (1-r_1)K$, we obtain

$$f(\mathrm{OPT}) - f(\tilde{S}) \leq \alpha \overline{r}_1 v + \alpha(1-r_1)v \leq \alpha v + \alpha \varepsilon r_1 v = \alpha v + O(\varepsilon)v.$$

Therefore, since $f(\mathrm{OPT}) \geq v$, we have $f(\tilde{S}) \geq (1-\alpha)v - O(\varepsilon)v$. □ □

By the above two lemmas, Theorem 3.6 holds, as may be seen below.

Proof of Theorem 3.6. It follows from Lemma 3.8 that, at the end of Algorithm Dynamic(v), one of the three conditions will be satisfied. If the first condition holds, then Lemma 3.9 implies that, for some $e \in \tilde{X}$,

$$f(\tilde{T}_e) \geq \min\left\{\frac{2}{5},\ \Gamma(\theta) - \frac{2}{5}\left(\frac{1-r_1}{1-\overline{r}_2}\right)\right\} v - O(\varepsilon)v.$$

If the second condition holds, then Lemma 3.10 implies that

$$f(\tilde{S}) \geq \min\left\{\frac{2}{5},\ 1-\alpha\right\} v - O(\varepsilon)v.$$

Since the output S of the algorithm is the best among $\{\tilde{S}, \tilde{S}'\} \cup \{\tilde{T}_e \mid e \in \tilde{X}\}$, we can see that (5) holds. Thus, since $c(S) \leq K$ by Lemma 3.7, the theorem holds. □ □

It turns out from Theorem 3.6 and Corollary 3.4 that we can find a $(2/5-\varepsilon)$-approximate solution when $c(o_2)$ is small.

Corollary 3.11. *Suppose that v satisfies $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$ and that $c(o_1) \in [K/2, 2K/3]$. If $c(o_2) \leq K/3$, then we can find, with a single pass, a set S such that $c(S) \leq K$ and $f(S) \geq (2/5 - O(\varepsilon))v$. The space complexity is $O(K\varepsilon^{-3}\log K)$.*

Proof. Recall that $\underline{r}_\ell, \overline{r}_\ell$ satisfy that $\underline{r}_\ell \leq r_\ell \leq \overline{r}_\ell$ and $\overline{r}_\ell \leq (1+\varepsilon)\underline{r}_\ell$ for $\ell \in \{1,2\}$. Also, θ_1 satisfies $\theta_1 v/(1+\varepsilon) \leq f(o_1) \leq \theta_1 v$. These values can be guessed from a geometric series of certain intervals. Specifically, since r_1 is in $[1/2, 2/3]$, $\overline{r}_1, \underline{r}_1$ can be taken from a geometric series of the interval $[1/2, 2/3]$: $\{(1+\varepsilon)^i \mid i \in \mathbb{Z}_+, 1/2 \leq (1+\varepsilon)^i \leq 2/3\} \cup \{1/2, 2/3\}$. Similarly, $\overline{r}_2, \underline{r}_2$ can be taken from a geometric series of the interval $[1/K, 1/3]$, as $c(o_2)$ is a positive integer and $r_2 = c(o_2)/K$. For θ_1, we may assume that θ_1 is in $[3/10, 2/5]$ by Corollary 3.4, because, if $\theta_1 \geq 2/5$, then taking a singleton e with maximum return will satisfy $f(e) \geq 2v/5$. This means that θ_1 can also be taken from a geometric series of the interval $[3/10, 2/5]$. We run Algorithm Dynamic(v) for each guessed value of $\underline{r}_\ell, \overline{r}_\ell$ $(\ell = 1, 2)$ and θ_1, and return the best one. The space complexity for guessing these values is $O(\varepsilon^{-3}\log K)$. Since Dynamic$(v)$ requires $O(K)$ space, total space complexity is $O(K\varepsilon^{-3}\log K)$.

Since $r_1 \geq 1/2$ and $\overline{r}_2 \leq 1/3$, it holds that

$$\frac{1-r_1}{1-\overline{r}_2} \leq \frac{1/2}{2/3} = \frac{3}{4} \text{ and } \frac{1}{1-\overline{r}_2} \leq \frac{3}{2}.$$

Since $\Gamma(\theta) \geq \frac{9}{10} - \frac{\theta}{2} \geq 0.7$ for $\theta \leq \frac{2}{5}$, (5) implies that

$$f(S) \geq \min\left\{\frac{2}{5},\ 1-\frac{2}{5}\cdot\frac{3}{2},\ 0.7-\frac{2}{5}\cdot\frac{3}{4}\right\} v - O(\varepsilon)v \geq \left(\frac{2}{5} - O(\varepsilon)\right) v,$$

which proves the lemma. □ □

3.2.2 Algorithm with Dynamic Update

Our algorithm Dynamic requires a good approximation v for $f(\mathrm{OPT})$. This requirement can be eliminated with dynamic updates in a way similar to that of [2, 24, 43]. We describe the idea briefly below.

Before describing the dynamic update technique, however, we should first note that, if we are given $m = \max_{e \in S} f(e)$ in advance, we can easily guess v. In fact, since $m \leq f(\mathrm{OPT}) \leq Km$, a value v with $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$ for $\varepsilon \in (0,1]$ will exist in the set $\mathcal{V} = \{(1+\varepsilon)^i \mid m \leq (1+\varepsilon)^i \leq Km, i \in \mathbb{Z}_+\} \cup \{m\}$. Hence we can run our algorithm for each $v \in \mathcal{V}$ in parallel and choose the best output. As the size of $\mathcal{V}$ is $O(\varepsilon^{-1}\log K)$, the total space complexity required is $O(K\varepsilon^{-4}\log^2 K)$, by Corollary 3.11.

To eliminate the assumption that we are given m in advance, we consider an algorithm which dynamically updates m to determine the range of guessed optimal values; it maintains the (tentative) maximum value $m' = \max f(e)$, where the maximum is taken over the items e arrived so far, and maintains v values in the interval between m' and Km'/α. More specifically, we perform our algorithm $\mathsf{Dynamic}(v)$ for each v in $\mathcal{V}' = \{(1+\varepsilon)^i \mid \frac{m'}{1+\varepsilon} \le (1+\varepsilon)^i \le Km'/\alpha, i \in \mathbb{Z}_+\}$. When a new item e arrives, we update m' and $\mathcal{V}'$, removing the data stored for $v \notin \mathcal{V}'$, and decide whether or not to add e to S and X for each $v \in \mathcal{V}'$.

We will see that it suffices to maintain v in $\mathcal{V}'$. Indeed, a value v with $v \le f(\mathrm{OPT}) \le (1+\varepsilon)v$ is clearly more than m', as $m' \le f(\mathrm{OPT})$. Further, if $v > Km'/\alpha$, then e is not selected, because, for any subset S, we have

$$f(e \mid S) \le f(e) \le m' < \frac{\alpha}{K} v \le \alpha \frac{c(e)}{K} v.$$

Therefore, we do not need to check the thresholding condition for $v \notin \mathcal{V}'$. The number of v's in $\mathcal{V}'$ is $O(\varepsilon^{-1} \log K)$, and for each v in $\mathcal{V}'$, our algorithm requires $O(K\varepsilon^{-3} \log K)$ space, by Corollary 3.11. Thus, the total space required is $O(K\varepsilon^{-4} \log^2 K)$. This proves Theorem 3.5.

3.3 Algorithm for Large $c(o_2)$

In this section, we propose our second algorithm, which is efficient when $c(o_2)$ is large. Since $o_1, o_2 \in \mathrm{OPT}$, it is clear that $c(o_1) + c(o_2) \le K$, and hence $r_2 \le 1 - r_1$, for which we may recall that $r_\ell = c(o_\ell)/K$ for $\ell = 1, 2$. As will be shown in the following lemma (see Section 3.3.3 for the proof), we can find a $(2/5 - \varepsilon)$-approximate solution when r_2 is very large, i.e., when $r_1 + r_2 \ge 1 - \varepsilon$.

Lemma 3.12. *If $c(o_1) + c(o_2) \ge (1-\varepsilon)K$, then we can find a $(2/5-\varepsilon)$-approximate solution with a single pass using $O(K\varepsilon^{-3} \log^3 K)$ space.*

Below, we assume that $r_2 \le 1 - r_1 - \varepsilon$, which implies that $\overline{r}_1 + \overline{r}_2 \le 1$. The goal of this section is to demonstrate the following theorem.

Theorem 3.13. *If $c(o_1) \in [K/2, 2K/3]$ and $c(o_2) \in [K/3, K - c(o_1) - \varepsilon]$, then we can find a $(2/5-\varepsilon)$-approximate solution with a single pass using $O(K\varepsilon^{-3} \log K)$ space.*

The proof is given in the following two subsections.

3.3.1 Algorithm with Optimal Value

In a way similar to that seen in the previous section, we assume here that we know in advance an approximate value v of $f(\mathrm{OPT})$, i.e., $v \le f(\mathrm{OPT}) \le (1+\varepsilon)v$. We also assume that we are given θ_ℓ such that $\theta_\ell v/(1+\varepsilon) \le f(o_\ell) \le \theta_\ell v$ for $\ell \in \{1, 2\}$. Define $E_\ell = \{e \in E \mid c(e) \in [\underline{r}_\ell K, \overline{r}_\ell K], f(e) \in [\theta_\ell v/(1+\varepsilon), \theta_\ell v]\}$ for $\ell \in \{1, 2\}$. Then $o_\ell \in E_\ell$ holds. We also define $\overline{E} = \{e \mid c(e) \le \overline{r}_2 K\}$. We may then assume that $E = E_1 \cup E_2 \cup \overline{E}$, as we can discard the other items.

In the algorithm (Algorithm 4), which we call $\mathsf{Parallel}$, we perform, *in parallel*, the thresholding algorithm and the procedure $\mathsf{PickOneItem}$ from Section 2.2. We apply $\mathsf{PickOneItem}$ to both E_1 and E_2 to obtain approximations of o_1 and o_2. The set X_ℓ will then include an approximation of o_ℓ for $\ell = 1, 2$. While finding X_1 and X_2, we check in Line 11 as to whether there exists a pair of items, one each from X_1 and X_2, respectively, whose f-value is more than $2v/5$. In parallel, we run the thresholding algorithm with $\alpha_\ell := \frac{2}{5(1-\overline{r}_\ell)}$ to $\overline{E}$ to obtain a set S_ℓ for $\ell = 1, 2$. If the output S_ℓ has

Algorithm 4

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

a size larger than $(1 - \overline{r}_\ell)K$ for some $\ell \in \{1, 2\}$, then the algorithm will return S_ℓ. In this case, Lemma 2.1 guarantees that $f(S_\ell)$ will be large. Otherwise, i.e., if $c(S_\ell)$ is small, there will be room for adding an item from X_ℓ. The algorithm returns the set that maximizes $f(S_\ell + e)$ for $e \in X_\ell$ and $\ell = 1, 2$. Intuitively speaking, the algorithm partitions the ground set E into three parts E_1, E_2 and $\overline{E}$, and it then returns the best set that can be obtained from two of the three parts.

Theorem 3.14. *Suppose that $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$. If $r_1 + r_2 \leq 1 - \varepsilon$, then Algorithm* Parallel(v) *returns a set S such that $c(S) \leq K$ and $f(S) \geq \gamma v - O(\varepsilon)v$, where*

$$\gamma = \min\left\{\frac{2}{5},\ \Gamma(\theta_2) + \theta_2 - \frac{2}{5}\left(\frac{2 - 2r_2 - r_1}{1 - \overline{r}_2}\right), \Gamma(\theta_1) + \theta_1 - \frac{2}{5}\left(\frac{2 - 2r_1 - r_2}{1 - \overline{r}_1}\right)\right\}. \tag{7}$$

The space complexity is $O(K)$.

In the rest of the section, we prove Theorem 3.14.

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

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

Let us next show (7) in Theorem 3.14. We consider the following two cases separately: the case in which o_2 arrives before o_1 and that in which o_1 arrives before o_2.

Case 1: Suppose that o_2 arrives before o_1. Consider the case in which $\ell = 2$. We may assume that the algorithm terminates at the end (not in Line 11). Further, if $c(\tilde{S}_2) \geq (1 - \overline{r}_2)K$, then $f(\tilde{S}_2) \geq 2v/5$ by Lemma 2.1. Thus we may assume that $c(\tilde{S}_2) < (1 - \overline{r}_2)K$.

Let $X_2 = \{e_1, e_2, \ldots, e_x\}$ be the set collected just before o_2 arrives. Then X_2 satisfies $f(e_j \mid \{e_1, e_2, \ldots, e_{j-1}\}) \geq (\theta_2 - \lambda_2(j-1))$ for each $j = 1, 2, \ldots, x$. When o_1 arrives, we return the set $\{e, o_1\}$ for some $e \in X_2$ if $f(\{o_1, e\}) \geq 2v/5$ at Line 11. Thus we may assume that $f(\{o_1, e\}) < 2v/5$ for any $e \in X_2$.

Lemma 3.15. *Suppose that $c(\tilde{S}_2) < (1-\overline{r}_2)K$ and that, for any $e \in X_2$, we have $f(\{o_1, e\}) < 2v/5$. Then there exists an item $e \in X_2$ such that*

$$f(\tilde{S}_2 + e) \geq \Gamma(\theta_2)v + \theta_2 v - \frac{2}{5}\left(\frac{2 - 2r_2 - r_1}{1 - \overline{r}_2}\right)v - O(\varepsilon)v.$$

Proof. By Lemma 2.2, we have $e \in X_2$ such that $f(\mathrm{OPT} - o_2 + e) \geq \Gamma(\theta_2)v - O(\varepsilon)v$. Since $f(\{o_1, e\}) < 2v/5$ and $f(e) \geq \theta_2 v/(1+\varepsilon)$, we can see that

$$f(o_1 \mid e) = f(\{o_1, e\}) - f(e) < \left(\frac{2}{5} - \theta_2 + O(\varepsilon)\right)v.$$

It then holds by submodularity that

$$\begin{aligned} f(\mathrm{OPT} - o_2 + e) &\leq f(o_1 \mid e) + f(\mathrm{OPT} - o_1 - o_2 + e) \\ &\leq \left(\frac{2}{5} - \theta_2 + O(\varepsilon)\right)v + f(\mathrm{OPT} - o_1 - o_2 + e). \end{aligned}$$

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

$$\Gamma(\theta_2)v - \left(\frac{2}{5} - \theta_2\right)v - O(\varepsilon)v \leq f(\mathrm{OPT} - o_1 - o_2 + e).$$

On the other hand, it follows from submodularity that

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

where the second to the last inequality follows from the fact that, since $c(\tilde{S}_2) \leq (1 - c(o_2))K$, any item $o \in \mathrm{OPT} - o_1 - o_2 - \tilde{S}_2$ will not be included in $\tilde{S}_2$ due to the thresholding condition, implying $f(o \mid \tilde{S}_2) < \alpha_2 \frac{c(o)}{K} v$. Combining these, we obtain

$$\begin{aligned} f(\tilde{S}_2 + e) &\geq \left(\Gamma(\theta_2) - \left(\frac{2}{5} - \theta_2\right) - \alpha_2(1 - r_1 - r_2)\right)v - O(\varepsilon)v \\ &\geq \left(\Gamma(\theta_2) + \theta_2 - \frac{2}{5}\frac{2 - 2r_2 - r_1}{1 - \overline{r}_2}\right)v - O(\varepsilon)v, \end{aligned}$$

where, we may recall, $\alpha_2 = \frac{2}{5(1-\overline{r}_2)}$. The proof is complete. □ □

Case 2: Suppose that o_1 arrives before o_2. Let X_1 be the set just before o_1 arrives. We can apply a symmetrical argument to Case 1. Here, we omit the proof.

Lemma 3.16. *Suppose that $c(\tilde{S}_1) < (1-\overline{r}_1)K$ and that, for any $e \in X_1$, we have $f(\{o_2, e\}) < 2v/5$. Then there exists an item $e \in X_1$ such that*

$$f(\tilde{S}_1 + e) \geq \Gamma(\theta_1)v + \theta_1 v - \frac{2}{5}\left(\frac{2 - 2r_1 - r_2}{1 - \overline{r}_1}\right)v.$$

We are now ready to prove Theorem 3.14.

Proof of Theorem 3.14. As mentioned at the beginning of the consideration of Case 1, we may assume that the algorithm terminates at the end (not in Line 11), and that $c(\tilde{S}_\ell) < (1 - \overline{r}_\ell)K$ for any $\ell \in \{1, 2\}$.

Suppose that o_2 arrives before o_1. Lemma 3.15 implies that there will exist an item $e \in X_2$ such that

$$f(\tilde{S}_2 + e) \geq \Gamma(\theta_2)v + \theta_2 v - \frac{2}{5}\left(\frac{2 - 2r_2 - r_1}{1 - \overline{r}_2}\right)v.$$

Hence we can see that $f(\tilde{S}'_2)$ will at least be the RHS.

Next suppose that o_1 arrives before o_2. In a way similar to that seen above, Lemma 3.16 implies that there will exist an item $e \in X_1$ such that

$$f(\tilde{S}_1 + e) \geq \Gamma(\theta_1)v + \theta_1 v - \frac{2}{5}\left(\frac{2 - 2r_1 - r_2}{1 - \overline{r}_1}\right)v.$$

Hence we see $f(\tilde{S}'_1)$ will at least be the RHS. This completes the proof. □ □

3.3.2 Proof of Theorem 3.13

In the algorithm $\mathsf{Parallel}(v)$, we need θ_ℓ for $\ell = 1, 2$, which are approximations of $f(o_\ell)/v$. We first observe that we may assume that θ_1 is in $[3/10, 2/5]$ by Corollary 3.4. We may further assume that θ_2 is in $[1/5, 2/5]$ by Theorem 3.2 as may be seen below.

Corollary 3.17. *Suppose that $c(o_1) \in [K/2, 2K/3]$ and $c(o_2) > K/3$. If $f(o_2) < f(\mathrm{OPT})/5$, then we can find, with a single pass, a set S such that $c(S) \leq K$ and $f(S) \geq (2/5 - O(\varepsilon))f(\mathrm{OPT})$. The space complexity is $O(K\varepsilon^{-4}\log^3 K)$.*

Proof. We may assume that we are given $\underline{r}_2, \overline{r}_2$ such that $\underline{r}_2 \leq r_2 \leq \overline{r}_2$ and $\overline{r}_2 \leq (1+\varepsilon)\underline{r}_2$. In fact, since $r_2 \in [1/3, 1/2]$, they can be guessed from a geometric series of the interval $[1/3, 1/2]$. The number of guessed values is $O(\varepsilon^{-1})$. For each guessed value, we do the following.

We see that $f(\mathrm{OPT} - o_2) \geq f(\mathrm{OPT}) - f(o_2) \geq \frac{4}{5}f(\mathrm{OPT})$ and $c(\mathrm{OPT} - o_2) \leq (1 - \underline{r}_2)K$. Consider maximizing $f(S)$ subject to $c(S) \leq (1 - \underline{r}_2)K$ in the set $\{e \in E \mid c(e) \leq (1 - \underline{r}_2)K\}$. The optimal value will be at least $f(\mathrm{OPT} - o_2) > \frac{4}{5}f(\mathrm{OPT})$. Now set $p = \frac{1}{1-\underline{r}_2}$. Since $\underline{r}_2 \geq 1/3$, we have $p \geq \frac{3}{2} \geq 2r_1$. Hence we can apply Theorem 3.2 to this instance in which $p \geq \frac{3}{2}$. The output S will then have a size of at most K, and, further,

$$\begin{aligned} f(S) &\geq \left(\frac{2p}{2p+3} - O(\varepsilon)\right)\left(\frac{4}{5}f(\mathrm{OPT})\right) \geq \left(\frac{1}{2} - O(\varepsilon)\right)\left(\frac{4}{5}f(\mathrm{OPT})\right) \\ &\geq \left(\frac{2}{5} - O(\varepsilon)\right)f(\mathrm{OPT}). \end{aligned}$$

Thus we obtain a $(2/5 - O(\varepsilon))$-approximation. □ □

Theorem 3.14, together with the above corollary, implies the following theorem.

Theorem 3.18. *Suppose that $v \le f(\mathrm{OPT}) \le (1+\varepsilon)v$. If $c(o_1) \in [K/2, 2K/3]$ and $c(o_2) \in [K/3, K - c(o_1) - \varepsilon]$, then we can find, with a single pass, a set S such that $c(S) \le K$ and $f(S) \ge (2/5 - O(\varepsilon))v$. The space complexity is $O(K\varepsilon^{-4}\log^3 K)$.*

Proof. In a way similar to that seen in the proof of Corollary 3.11, we guess necessary parameters from a geometric series of certain intervals, and run the algorithm $\mathsf{Parallel}(v)$ for each set of guessed parameters. Since r_1 is in $[1/2, 2/3]$ and r_2 is in $[1/3, 1/2]$, $\underline{r}_\ell$, $\overline{r}_\ell$ can be guessed from a geometric series of the intervals. This will require $O(\varepsilon^{-1})$ space for each $\ell = 1, 2$. Additionally, since we may assume that θ_1 is in $[3/10, 2/5]$ by Corollary 3.4, and that θ_2 is in $[1/5, 2/5]$ by Corollary 3.17, θ_ℓ can be guessed using $O(\varepsilon^{-1})$ space for each $\ell = 1, 2$. Since we can run the case of $\ell = 1, 2$ separately in $\mathsf{Parallel}(v)$, the space complexity for guessing parameters is $O(\varepsilon^{-2})$. Therefore, since $\mathsf{Parallel}(v)$ takes $O(K)$ space, the total space complexity is $O(K\varepsilon^{-2})$.

By Theorem 3.14, we can find a set S such that $f(S) \ge \gamma v - O(\varepsilon)v$, where

$$\gamma = \min\left\{\frac{2}{5},\ \Gamma(\theta_2) + \theta_2 - \frac{2}{5}\left(\frac{2 - 2r_2 - r_1}{1 - \overline{r}_2}\right),\ \Gamma(\theta_1) + \theta_1 - \frac{2}{5}\left(\frac{2 - 2r_1 - r_2}{1 - \overline{r}_1}\right)\right\}.$$

By the definition of Γ, since $\theta_\ell \le 2v/5$ for $\ell = 1, 2$, it holds that

$$\Gamma(\theta_\ell) \ge \frac{9}{10} - \frac{\theta_\ell}{2}.$$

Further, since $\theta_1 \ge 3v/10$ and $\theta_2 \ge v/5$, we see that

$$\Gamma(\theta_1) + \theta_1 \ge \frac{9}{10} - \frac{\theta_1}{2} + \theta_1 \ge 1.05$$
$$\Gamma(\theta_2) + \theta_2 \ge \frac{9}{10} - \frac{\theta_2}{2} + \theta_2 \ge 1.$$

Hence, since $2/3 \ge r_1 \ge 1/2$ and $r_2 \ge 1/3$, it holds that

$$\begin{aligned}\Gamma(\theta_1) + \theta_1 - \frac{2}{5}\left(\frac{2 - 2r_1 - r_2}{1 - \overline{r}_1}\right) &\ge 1.05 - \frac{2}{5}\left(2 - \frac{r_2}{1 - r_1}\right) - O(\varepsilon) \\ &\ge 0.51 - O(\varepsilon),\end{aligned}$$

where the minimum of $\frac{r_2}{1-r_1}$ is attained when $r_1 = 1/2$ and $r_2 = 1/3$. Further,

$$\begin{aligned}\Gamma(\theta_2) + \theta_2 - \frac{2}{5}\left(\frac{2 - 2r_2 - r_1}{1 - \overline{r}_2}\right) &\ge 1 - \frac{2}{5}\left(2 - \frac{r_1}{1 - r_2}\right) - O(\varepsilon) \\ &= 0.5 - O(\varepsilon),\end{aligned}$$

where the minimum of $\frac{r_1}{1-r_2}$ is attained when $r_1 = 1/2$ and $r_2 = 1/3$. Thus the ratio γ is at least $2/5 - O(\varepsilon)$. □ □

It is not difficult to see that the dynamic update technique in Section 3.2.2 can be applied directly by replacing α with the minimum of α_1 and α_2. Thus we can perform the algorithm without having v using $O(K\varepsilon^{-3}\log K)$ space. This proves Theorem 3.13.

3.3.3 Algorithm When $c(o_1) + c(o_2) \approx K$

Let us prove Lemma 3.12, which considers the case $r_2 \geq 1 - r_1 - \varepsilon$. In this case, we can use an idea similar to but simpler than that used in Section 3.3.1.

Let us first show that, if the optimal solution OPT has a size of two, we will be able to find a $(2/3 - \varepsilon)$-approximate solution with a single pass.

Lemma 3.19. *Suppose that* $\mathrm{OPT} = \{o_1, o_2\}$ *for some* $o_1, o_2 \in E$. *We can then find a set* S *using a single pass and* $O(K\varepsilon^{-2} \log K)$ *space such that* $c(S) \leq K$ *and*

$$f(S) \geq \left(\frac{2}{3} - O(\varepsilon)\right) f(\{o_1, o_2\}).$$

Proof. Let v' be an approximate value of $f(\{o_1, o_2\})$, i.e., $v' \leq f(\{o_1, o_2\}) \leq (1+\varepsilon)v'$. We may assume that $f(o_\ell) \geq v'/3$ and $f(o_\ell) \leq 2v'/3$, as otherwise taking a singleton e with maximum return would satisfy that $f(e) \geq 2v'/3$. For $\ell = 1, 2$, we guess approximations θ_ℓ of $f(o_\ell)$ from a geometric series of the interval $[1/3, 2/3]$, using $O(\varepsilon^{-1})$ space, i.e., $\theta_\ell/(1+\varepsilon) \leq f(o_\ell) \leq \theta_\ell$.

First, suppose that o_1 arrives before o_2. For each $i = 1, 2, \ldots, K/2$, define $F_i = \{e \in E \mid i \leq c(e) \leq K - i\}$. We observe that, if $i \leq c(o_2)$, then $o_1 \in F_i$.

We perform in parallel to collect items from F_i for each $i = 1, 2, \ldots, K/2$, using Lemma 2.2 with θ_1. Let $X_i \subseteq F_i$ be the collection from F_i when o_1 arrives. It then follows from Lemma 2.2 that, for each i with $i \leq c(o_2)$, there will exist $e^* \in X_i$ such that $c(e^*) \leq K - i$ and

$$f(\mathrm{OPT} - o_1 + e^*) = f(\{o_2, e^*\}) \geq (\Gamma(\theta_1) - O(\varepsilon))v' \geq (2/3 - O(\varepsilon))v'.$$

For each arriving e after o_1, we check whether or not there exists $e' \in X_{c(e)}$ such that $f(\{e, e'\}) \geq (2/3 - O(\varepsilon))v'$. Since o_2 arrives after constructing X_i, we will be able to find at least one pair satisfying the condition. Since $|X_i| = O(1)$, it will take a total of $O(\varepsilon^{-1}K)$ space and $O(\varepsilon^{-1}n)$ time.

Next suppose that o_2 arrives before o_1. The argument can be made symmetrical by setting $F_i = \{e \in E \mid 1 \leq c(e) \leq K - i\}$ for each $i = K/2, \ldots, K$ and using Lemma 2.2 with θ_2. Note that, if $i \leq c(o_1)$, then $o_2 \in F_i$.

Therefore, given an approximate value v' of $f(\{o_1, o_2\})$, we will be able, using a single pass and $O(K\varepsilon^{-1})$ space, to find a set S such that $c(S) \leq K$ and $f(S) \geq \left(\frac{2}{3} - O(\varepsilon)\right) v'$. The pseudo-code description is given in Algorithm 5.

Finally, we can, by means of a dynamic update technique with additional space $O(\varepsilon^{-1} \log K)$, eliminate the assumption of having the approximation v'. This completes the proof. □ □

The above lemma suggests the algorithm described below. We divide the ground set E into two parts $E' = \{e \in E \mid c(e) \in [\underline{r}_2 K, \overline{r}_1 K]\}$ and $\overline{E'} = \{e \mid c(e) \leq \overline{r}_2 K\}$. Then $\{o_1, o_2\}$ will be included in E', and we can apply the algorithm of Lemma 3.19 to E'. If $f(\{o_1, o_2\}) \geq \frac{3}{5} f(\mathrm{OPT})$, then it holds by Lemma 3.19 that the obtained solution is a $(2/5 - \varepsilon)$-approximation. Thus we may assume that $f(\{o_1, o_2\}) < \frac{3}{5} f(\mathrm{OPT})$, meaning that $f(\mathrm{OPT} - o_1 - o_2) \geq f(\mathrm{OPT}) - f(\{o_1, o_2\}) \geq \frac{2}{5} f(\mathrm{OPT})$. Now $f(\mathrm{OPT} - o_1 - o_2) \geq \frac{2}{5} f(\mathrm{OPT})$ but $c(\mathrm{OPT} - o_1 - o_2) \leq \varepsilon K$, which means that $\mathrm{OPT} - o_1 - o_2$ will be a "dense" set. Therefore, applying Theorem 3.2 to $\overline{E'}$, we can obtain a $(1 - \varepsilon)$-approximation for $\mathrm{OPT} - o_1 - o_2$.

We are now ready to prove Lemma 3.12.

Algorithm 5

procedure TwoItems(v, ℓ, θ_ℓ) ▷ $\ell \in \{1, 2\}$
  $F_i := \{e \in E \mid i \le c(e) \le K - i\}$ for $i = 1, 2, \ldots, K/2$.
  $F_i := \{e \in E \mid 1 \le c(e) \le K - i\}$ for $i = K/2, \ldots, K$.
  Set $t := 1$. Define λ_ℓ from θ_ℓ by Lemma 2.2 for $\ell = 1, 2$.
  $X_i := \emptyset$ for $i = 1, 2, \ldots, K$.
  while item e is arriving **do**
    If $\ell = 1$ **then** $I := \{1, 2, \ldots, K/2\}$ **else** $I := \{K/2, \ldots, K\}$.
    for each $i \in I$ **do**
      if $e \in E_i$ **then**
        if $|X_i| < t + 1$ and $f(e \mid X_i) \ge (\theta_\ell - \lambda_\ell |X_i|)v$ **then**
          $X_i := X_i + e$.
      else
        if $f(e + e') \ge 2/3v$ for some $e' \in X_{c(e)}$ **then return** $e + e'$.

Proof of Lemma 3.12. If $f(\{o_1, o_2\}) \ge \frac{3}{5} f(\mathrm{OPT})$, then we have finished using Lemma 3.19, while assuming otherwise implies that $f(\mathrm{OPT} - o_1 - o_2) \ge f(\mathrm{OPT}) - f(\{o_1, o_2\}) \ge \frac{2}{5} f(\mathrm{OPT})$.

Consider maximizing $f(S)$ subject to $c(S) \le \varepsilon K$ in the set $\{e \in E \mid c(e) \le \varepsilon K\}$. Since $c(\mathrm{OPT} - o_1 - o_2) \le \varepsilon K$, the optimal value of this instance will be at least $f(\mathrm{OPT} - o_1 - o_2) \ge \frac{2}{5} f(\mathrm{OPT})$. We then apply Theorem 3.2 with $p = 1/\varepsilon$ to this instance. The output S will then have a size of at most K, and further, we will have

$$\begin{aligned} f(S) &\ge \left(\frac{2p}{2p+3} - O(\varepsilon)\right)\left(\frac{2}{5} f(\mathrm{OPT})\right) \\ &= \left(\frac{1}{1+1.5\varepsilon} - O(\varepsilon)\right)\left(\frac{2}{5} f(\mathrm{OPT})\right) \ge \left(\frac{2}{5} - O(\varepsilon)\right) f(\mathrm{OPT}). \end{aligned}$$

In this way, we obtain a $(2/5 - O(\varepsilon))$-approximation. Space complexity here is $O(K\varepsilon^{-3} \log^3 K)$. □ □

References

[1] N. Alon, I. Gamzu, and M. Tennenholtz. Optimizing budget allocation among channels and influencers. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 381–388, 2012.

[2] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 671–680, 2014.

[3] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1497–1514, 2013.

[4] E. Balkanski, A. Rubinstein, and Y. Singer. An exponential speedup in parallel running time for submodular maximization without loss in approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 283–302, 2019.

[5] E. Balkanski and Y. Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, STOC 2018, pages 1138–1151, New York, NY, USA, 2018. ACM.

[6] R. Barbosa, A. Ene, H. Le Nguyen, and J. Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1236–1244. JMLR.org, 2015.

[7] R. D. P. Barbosa, A. Ene, H. L. Nguyen, and J. Ward. A new framework for distributed submodular maximization. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 645–654, Oct 2016.

[8] M. Bateni, H. Esfandiari, and V. Mirrokni. Almost optimal streaming algorithms for coverage problems. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '17, pages 13–23, New York, NY, USA, 2017. ACM.

[9] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[10] A. Chakrabarti and S. Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

[11] T.-H. H. Chan, Z. Huang, S. H.-C. Jiang, N. Kang, and Z. G. Tang. Online submodular maximization with free disposal: Randomization beats for partition matroids online. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1204–1223, 2017.

[12] T.-H. H. Chan, S. H.-C. Jiang, Z. G. Tang, and X. Wu. Online submodular maximization problem with vector packing constraint. In *Annual European Symposium on Algorithms (ESA)*, pages 24:1–24:14, 2017.

[13] C. Chekuri, S. Gupta, and K. Quanrud. Streaming algorithms for submodular function maximization. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 9134, pages 318–330, 2015.

[14] C. Chekuri and K. Quanrud. Submodular function maximization in parallel via the multilinear relaxation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 303–322, 2019.

[15] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014.

[16] A. Ene and H. L. Nguyễn. A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In *The 46th International Colloquium on Automata, Languages and Programming (ICALP 2019)*, number to appear, 2019.

[17] A. Ene and H. L. Nguyễn. Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 274–282, 2019.

[18] M. Feldman, A. Karbasi, and E. Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 730–740, 2018.

[19] M. Feldman, A. Norouzi-Fard, O. Svensson, and R. Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1363–1374. ACM, 2020.

[20] Y. Filmus and J. Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. *SIAM Journal on Computing*, 43(2):514–542, 2014.

[21] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions i. *Mathematical Programming*, pages 265–294, 1978.

[22] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions ii. *Mathematical Programming Study*, 8:73–87, 1978.

[23] C. Huang and N. Kakimura. Multi-pass streaming algorithms for monotone submodular function maximization. *CoRR*, abs/1802.06212, 2018.

[24] C.-C. Huang, N. Kakimura, and Y. Yoshida. Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. In *The 20th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems(APPROX2017)*, 2017.

[25] E. Kazemi, M. Mitrovic, M. Zadimoghaddam, S. Lattanzi, and A. Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *International Conference on Machine Learning (ICML2019)*, pages 3311–3320, 2019.

[26] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.

[27] A. Krause, A. P. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[28] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 545–554, 2013.

[29] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. *ACM Trans. Parallel Comput.*, 2(3):14:1–14:22, Sept. 2015.

[30] J. Lee. *Maximum Entropy Sampling*, volume 3 of *Encyclopedia of Environmetrics*, pages 1229–1234. John Wiley & Sons, Ltd., 2006.

[31] J. Lee, M. Sviridenko, and J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010.

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

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

[34] A. McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, May 2014.

[35] A. McGregor and H. T. Vu. Better streaming algorithms for the maximum coverage problem. In *International Conference on Database Theory (ICDT)*, 2017.

[36] B. Mirzasoleiman, S. Jegelka, and A. Krause. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. In *Proc. Conference on Artificial Intelligence (AAAI)*, Feburary 2018.

[37] Z. Nutov and E. Shoham. Practical budgeted submodular maximization, 2020.

[38] T. Soma, N. Kakimura, K. Inaba, and K. Kawarabayashi. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 351–359, 2014.

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

[40] J. Tang, X. Tang, A. Lim, K. Han, C. Li, and J. Yuan. Revisiting modified greedy algorithm for monotone submodular maximization with a knapsack constraint, 2020.

[41] L. Wolsey. Maximising real-valued submodular functions: primal and dual heuristics for location problems. *Mathematics of Operations Research*, 1982.

[42] Y. Yoshida. Maximizing a monotone submodular function with a bounded curvature under a knapsack constraint. https://arxiv.org/abs/1607.04527, 2016.

[43] Q. Yu, E. L. Xu, and S. Cui. Streaming algorithms for news and scientific literature recommendation: Submodular maximization with a d-knapsack constraint. *IEEE Global Conference on Signal and Information Processing*, 2016.