Simple Deterministic Approximation for Submodular Multiple Knapsack Problem

Xiaoming Sun
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China

Jialin Zhang
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China

Zhijie Zhang[1]
Center for Applied Mathematics of Fujian Province, School of Mathematics and Statistics, Fuzhou University, China

Abstract

Submodular maximization has been a central topic in theoretical computer science and combinatorial optimization over the last decades. Plenty of well-performed approximation algorithms have been designed for the problem over a variety of constraints. In this paper, we consider the submodular multiple knapsack problem (SMKP). In SMKP, the profits of each subset of elements are specified by a monotone submodular function. The goal is to find a feasible packing of elements over multiple bins (knapsacks) to maximize the profit. Recently, Fairstein et al. [ESA20] proposed a nearly optimal $(1-e^{-1}-\epsilon)$-approximation algorithm for SMKP. Their algorithm is obtained by combining configuration LP, a grouping technique for bin packing, and the continuous greedy algorithm for submodular maximization. As a result, the algorithm is somewhat sophisticated and inherently randomized. In this paper, we present an arguably simple deterministic combinatorial algorithm for SMKP, which achieves a $(1-e^{-1}-\epsilon)$-approximation ratio. Our algorithm is based on very different ideas compared with Fairstein et al. [ESA20].

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Submodular maximization, knapsack problem, deterministic algorithm

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.98

Related Version *Full Version*: https://arxiv.org/abs/2003.11450

Funding This work was supported in part by the National Natural Science Foundation of China Grants No. 61832003, 62272441.

1 Introduction

The *multiple knapsack problem* (MKP) is defined as follows. We are given a set N of n elements and a set M of m bins (knapsacks). Each element $u \in N$ has a positive cost $c(u) > 0$ and a positive profit $p(u) > 0$. The cost (profit) of a subset $S \subseteq N$ equals the sum of the costs (profits) of its elements. The j-th bin in M has a positive budget $B_j > 0$ for $1 \le j \le m$. A subset $S \subseteq N$ is feasible if there is a disjoint partition $\{S_j\}_{j=1}^m$ of S such that $c(S_j) \le B_j$ for $1 \le j \le m$. The goal is to find a feasible set S (and its partition $\{S_j\}_{j=1}^m$) whose profit is maximized. It is well-known that the problem admits a PTAS but no FPTAS assuming P $\neq$ NP [18, 5, 17].

In this paper, we consider the submodular generalization of the above problem, referred to as the *submodular multiple knapsack problem* (SMKP). In SMKP, the profit is in general non-additive and specified by a *monotone submodular* function $f : 2^N \to \mathbb{R}_+$. Here, a set function $f : 2^N \to \mathbb{R}$ is monotone if $f(S) \le f(T)$ for any $S \subseteq T$ and submodular if

[1] Corresponding author

31st Annual European Symposium on Algorithms (ESA 2023).
Editors: Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman; Article No. 98; pp. 98:1–98:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$ for any $S \subseteq T$ and $u \notin T$. The goal is again to find a feasible set S which maximizes the profit $f(S)$. When $m = 1$, the problem reduces to submodular maximization under a knapsack constraint, which enjoys an optimal $(1 - e^{-1})$-approximation [18, 25].

Submodular functions capture the effect of diminishing returns in the economy and generalize many well-known functions such as coverage functions, cut functions, matroid rank functions, and log determinants. By introducing a submodular objective, SMKP falls in the field of submodular maximization, which studies maximization problems with submodular objectives, including maximum coverage problem, maximum cut problem, submodular welfare problem [26], influence maximization [19]. The study of submodular maximization has lasted for more than forty years. As early as 1978, it was shown that for monotone submodular maximization, a greedy algorithm achieves a $(1 - e^{-1})$-approximation under the cardinality constraint [24] and a $1/2$ approximation under the matroid constraint [15]. On the other hand, even for the cardinality constraint, the problem does not admit an approximation ratio better than $1 - e^{-1}$ [23]. It was a longstanding open question whether the problem admits a $(1 - e^{-1})$-approximation under the matroid constraint. In 2008, Vondrák [26] made a big breakthrough and answered this question affirmatively by proposing the so-called *continuous greedy* algorithm. Since then, plenty of optimal or well-performed approximation algorithms have been proposed for submodular maximization over a variety of constraints [2, 3, 4, 7, 12, 14, 16, 21, 22, 27].

For SMKP, a nearly optimal $(1-e^{-1}-\epsilon)$-approximation algorithm based on the continuous greedy technique was recently proposed in [9]. Their algorithm relies on two key ideas. First, they showed that by defining a configuration LP, an SMKP instance whose all bins have the same budget can be reduced to submodular maximization under 2-dimensional packing constraints (SMPC). Second, they developed a grouping technique inspired by [6] to convert a general SMKP instance to a leveled instance where bins are partitioned into blocks and bins in the same block have the same budget. In this way, they are able to reduce a general SMKP instance to an SMPC instance. They finally finished their work by a refined analysis of the continuous greedy algorithm for SMPC.

The techniques adopted by [9] and the way to combine them are somewhat sophisticated, which makes their algorithm not easy to understand and implement. Besides, the continuous greedy technique involves a sampling process and therefore their algorithm is inherently randomized. To the best of our knowledge, no deterministic algorithm was known for SMKP. In this paper, we present a simple deterministic combinatorial algorithm for SMKP, which achieves a $(1 - e^{-1} - \epsilon)$-approximation ratio.

▶ **Theorem 1.** *For any $\epsilon > 0$, there exists a deterministic combinatorial algorithm for SMKP that achieves a $(1 - e^{-1} - \epsilon)$-approximation ratio and runs in polynomial time.*

1.1 Technique Overview

We start with solving SMKP instances under the *identical case*, where all the bins have the same budget B. Such instances can be reduced to exponential-size instances of submodular maximization subject to a cardinality constraint. Inspired by this observation, we design an algorithm for the identical case by mimicking the greedy algorithm for the cardinality constraint. See Section 1.1.1 for details.

For any general SMKP instance, we use the grouping technique developed by [9] to convert it to the so-called leveled instance. While Fairstein et al. [9] resorts to the configuration LP to solve the leveled instance, we present a simple $(1 - e^{-1} - \epsilon)$-approximation algorithm for it by exploiting its structure and invoking our algorithm for the identical case as a subroutine. See Section 1.1.2 for details.

1.1.1 The Identical Case

Under the identical case, SMKP can be regarded as an exponential-size instance of submodular maximization subject to a cardinality constraint. Specifically, let $\mathcal{I} = \{S \subseteq N \mid c(S) \leq B\}$. For any $\mathcal{T} \subseteq \mathcal{I}$, define $g(\mathcal{T}) = f(\cup_{S \in \mathcal{T}} S)$. It is easy to verify that g is a monotone submodular function. Then, $\max\{g(\mathcal{T}) \mid |\mathcal{T}| \leq m\}$ describes the SMKP instance under the identical case.

Inspired by the above observation, our algorithm packs bins one by one and manages to make each bin pack at least the average marginal value of the optimal solution over m bins. In other words, for the j-th bin, it aims to find a set S_j such that $f(S_j \mid \cup_{i=1}^{j-1} S_i) \geq \frac{1}{m} f(OPT \mid \cup_{i=1}^{j-1} S_i)$, where OPT denotes the optimal solution. This naturally leads to $(1 - e^{-1})$ approximation.

We take the first bin as an example and explain that it is possible to find a set S_1 such that $f(S_1) \geq \frac{1}{m} f(OPT)$ when m is large enough. If S_1 is obtained by packing elements in sequence greedily according to their marginal densities, then we can prove

$$f(S_1) \geq (1 - e^{-c(S_1)/c(OPT)}) \cdot f(OPT).$$

If we further allow S_1 to violate the budget constraint by adding one more element, then $c(S_1) \geq B$. Together with $c(OPT) \leq mB$, we have

$$f(S_1) \geq (1 - e^{-1/m}) \cdot f(OPT) \approx \frac{1}{m} f(OPT).$$

The story has not ended since the last element added to S_1 violates the budget constraint. To handle this issue, our algorithm divides elements into large and small elements according to their costs and then packs them in different ways. Specifically, an element $u \in N$ is large if $c(u) > \epsilon B$ and small otherwise. Our algorithm packs large elements by enumeration since there are polynomial ways to pack them in total. It packs small elements greedily as before. In this way, the last element added to S_1 has a cost less than ϵB and there are at most m such elements. Thus, all of them can be repacked using additional ϵm bins and all S_j's will then become feasible.

In Lemma 5, we show that $f(S_1) \geq \frac{1}{m} f(OPT)$ still holds although we introduce the enumeration step.

1.1.2 The General Case

Observe that a general SMKP instance can be reduced to an exponential-size instance of submodular maximization subject to a partition matroid constraint. Specifically, let $\mathcal{I}_j = \{S \subseteq N \mid c(S) \leq B_j\}$ be the feasible region for the j-th bin and $\mathcal{I} = \cup_{j=1}^{m} \mathcal{I}_j$. For any $\mathcal{T} \subseteq \mathcal{I}$, define $g(\mathcal{T}) = f(\cup_{S \in \mathcal{T}} S)$. Then, $\max\{g(\mathcal{T}) \mid |\mathcal{T} \cap \mathcal{I}_j| \leq 1, 1 \leq j \leq m\}$ describes the general SMKP instance. Recall that the optimal $(1 - e^{-1})$-approximation for the partition matroid constraint is obtained via the continuous greedy algorithm [26]. Thus, it is not a good idea to solve general SMKP instances directly.

The difficulty in solving general SMKP stems from that the budgets are distinct. Therefore, we first consider an "intermediate" instance where bins can be partitioned into r blocks $\{M_k\}_{k=1}^{r}$ such that block M_k contains sufficiently many bins and all of them have the same budget B_k. Clearly, this instance is slightly more general than the instance under the identical case. It can also be reduced to an exponential-size instance of submodular maximization subject to a partition matroid constraint. Specifically, let $\mathcal{I}_k = \{S \subseteq N \mid c(S) \leq B_k\}$ for $1 \leq k \leq r$ and $\mathcal{I} = \cup_{k=1}^{r} \mathcal{I}_k$. For any $\mathcal{T} \subseteq \mathcal{I}$, define $g(\mathcal{T}) = f(\cup_{S \in \mathcal{T}} S)$. Then, $\max\{g(\mathcal{T}) \mid |\mathcal{T} \cap \mathcal{I}_k| \leq |M_k|, 1 \leq k \leq r\}$ describes the above SMKP instance.

The above two reductions lead to different constraints $|\mathcal{T} \cap \mathcal{I}_j| \leq 1$ and $|\mathcal{T} \cap \mathcal{I}_k| \leq |M_k|$. For convenience, assume that $1/\epsilon$ is an integer, $|M_k| \geq 1/\epsilon$ and $\epsilon|M_k|$ is an integer for all $1 \leq k \leq r$. Our key observation is that for constraint $\{\mathcal{T} \subseteq \mathcal{I} \mid |\mathcal{T} \cap \mathcal{I}_k| \leq |M_k|, 1 \leq k \leq r\}$, there is a simple deterministic algorithm that can achieve $(1 - e^{-1} - \epsilon)$-approximation. The algorithm runs in $1/\epsilon$ iterations. In each iteration, block M_k is visited in sequence and the algorithm will pack $\epsilon|M_k|$ bins in M_k. This forms an SMKP instance under the identical case. Thus, we can invoke our algorithm for the identical case to solve it.

Finally, we apply a grouping technique from [9] to convert a general instance to a t-leveled instance which has blocks $\{M_k\}_{k=1}^r$ and bins in the same block have the same budget. Besides, each of the first t^2 blocks contains a single bin, and each of the remaining blocks contains at least t. This is very similar to the intermediate instance before and it is not difficult to handle the first t^2 blocks.

1.2 Related Work

MKP has been fully studied previously. Kellerer [18] proposed the first PTAS for the identical case of the problem. Soon after, Chekuri and Khanna [5] proposed a PTAS for the general case. The result was later improved to an EPTAS by Jansen [17]. On the other hand, it is easy to see that the problem does not admit an FPTAS even for the case of $m = 2$ bins unless P = NP [5].

SMKP contains submodular maximization subject to a knapsack constraint as a special case. For this problem, there is an optimal $(1 - e^{-1})$-approximation algorithm that runs in $O(n^5)$ time [18, 25]. Later, a fast algorithm was proposed in [1] that achieves a $(1 - e^{-1} - \epsilon)$-approximation ratio and runs in $n^2(\log n/\epsilon)^{O(1/\epsilon^8)}$ time[2]. This was recently improved in [8] by a new algorithm that runs in $(1/\epsilon)^{O(1/\epsilon^4)} n \log^2 n$ time. The last two algorithms are impractical due to their high dependence on $1/\epsilon$. Very recently, a $(1 - e^{-1})$-approximation algorithm was proposed in [20, 13], which runs in $O(n^4)$ time. This algorithm can be further accelerated to achieve $(1 - e^{-1} - \epsilon)$-approximation in $\widetilde{O}(n^3/\epsilon)$ time.

To the best of our knowledge, SMKP was first considered in Feldman's Ph. D thesis [11]. Feldman proposed a polynomial time $(1/9 - o(1))$-approximation algorithm and a pseudo-polynomial time 1/4 approximation algorithm for the general case of SMKP. For the identical case, he improved the results to a polynomial time $((e - 1)/(3e - 1) - o(1)) \approx 0.24$ approximation algorithm and a pseudo-polynomial time $(1 - e^{-1} - o(1))$-approximation algorithm. These algorithms are based on the *continuous greedy* technique and *contension resolution schemes* [27], and hence involve randomness inherently. Recently, Fairstein et al. [9] proposed a polynomial time randomized $(1 - e^{-1} - \epsilon)$-approximation algorithm for general SMKP.

1.3 Organization

In Section 2, we first formulate SMKP and introduce some notations. Then, we present a greedy algorithm that packs elements greedily according to their marginal densities. In Section 3, we present a $(1 - e^{-1} - \epsilon)$-approximation algorithm for SMKP under the identical case, assuming the number of bins $m \geq 1/(4\epsilon^3)$. In Section 4, we present a $(1 - e^{-1} - \epsilon)$-approximation algorithm for general SMKP. We conclude the paper and list some open problems in Section 5.

[2] As pointed out by [28, 8], the result in [1] has some issues.

Algorithm 1 GREEDY.

Input: elements N, budgets $\{B_j\}_{j=1}^m$, profit f, cost c.
$S_j = \emptyset$ for $1 \le j \le m$ and $S = \cup_{j=1}^m S_j$.
while *$N \setminus S \neq \emptyset$ and there exists $1 \le j \le m$ such that $c(S_j) < B_j$* **do**
  $u^* = \arg\max_{u \in N \setminus S} f(u \mid S)/c(u)$.
  $S_j = S_j + u^*$ and $S = S + u^*$.
end
return $S = \cup_{j=1}^m S_j$.

2 Preliminaries

An instance of the *submodular multiple knapsack problem* (SMKP) is defined as follows. We are given a set N of n elements and a set M of m bins (knapsacks). Each element $u \in N$ has a positive cost $c(u) > 0$. A subset $S \subseteq N$ of elements has a cost $c(S) = \sum_{u \in S} c(u)$. The j-th bin in M has a positive budget $B_j > 0$ for $1 \le j \le m$. A subset $S \subseteq N$ is feasible for the problem if there is a disjoint partition $\{S_j\}_{j=1}^m$ of S such that $c(S_j) \le B_j$ for $1 \le j \le m$. The profit of each subset $S \subseteq N$ of elements is specified by a *normalized*, *monotone* and *submodular* function $f : 2^N \to \mathbb{R}_+$. For a non-negative set function $f : 2^N \to \mathbb{R}_+$, it is called *normalized* if $f(\emptyset) = 0$, *monotone* if $f(S) \le f(T)$ for any $S \subseteq T$, and *submodular* if $f(S \cup \{u\}) - f(S) \ge f(T \cup \{u\}) - f(T)$ for any $S \subseteq T$ and $u \notin T$. The goal is to find a feasible set S (and its partition $\{S_j\}_{j=1}^m$) such that the profit $f(S)$ (or $f(\cup_{j=1}^m S_j)$) is maximized.

An SMKP instance is specified by $(N, M, \{B_j\}_{j \in M}, f, c)$. Throughout this paper, we use OPT to denote the optimal solution of an SMKP instance. Let $S + u$ be a shorthand for $S \cup \{u\}$. For the objective function f, we also use $f(u \mid S)$ and $f(T \mid S)$ to denote the marginal values $f(S + u) - f(S)$ and $f(S \cup T) - f(S)$, respectively. f is accessed via a value oracle that returns $f(S)$ when set $S \subseteq N$ is queried. The query complexity of any algorithm for SMKP should be polynomial in the size of the problem.

2.1 The Greedy Algorithm

We first present a greedy algorithm, which is depicted as Algorithm 1. It serves as a cornerstone for other algorithms in this paper. It returns a (possibly infeasible) set with a $(1 - 1/e)$ approximation ratio. It packs elements one by one greedily, according to their *densities*, namely the ratios of their marginal values to their costs. The process continues provided there exists some bin whose budget has not been exhausted yet. As a side effect, each bin may pack one more element whose addition exceeds the budget of that bin. For convenience, we refer to this element as a *reserved element*. Nonetheless, we show that the set returned by Algorithm 1 has a large profit.

▶ **Lemma 2.** *Let S be the set returned by Algorithm 1. For any set $X \subseteq N$, we have*

$$f(S) \ge \left(1 - e^{-c(S)/c(X)}\right) \cdot f(X).$$

Proof. If $c(S) < \sum_{j=1}^m B_j$, there is some j such that $c(S_j) < B_j$. It means that Algorithm 1 ended with $S = N$. Thus, the lemma follows by monotonicity.

Now consider the case where $c(S) \ge \sum_{j=1}^m B_j$. Assume that $S = \{u_1, u_2, \ldots, u_\ell\}$, and for $0 \le i \le \ell$, $S^i = \{u_1, u_2, \ldots u_i\}$ denotes the first i elements packed by Algorithm 1. Then, by the greedy rule,

$$\frac{f(u_i \mid S^{i-1})}{c(u_i)} \ge \frac{f(x \mid S^{i-1})}{c(x)}, \forall x \in X \setminus S^{i-1}.$$

By moving $c(x)$ to the left and summing over $x \in X \setminus S^{i-1}$,

$$c(X \setminus S^{i-1}) \cdot \frac{f(u_i \mid S^{i-1})}{c(u_i)} \geq \sum_{x \in X \setminus S^{i-1}} f(x \mid S^{i-1}) \geq f(X \setminus S^{i-1} \mid S^{i-1}).$$

The last inequality holds since f is submodular. This gives us

$$\frac{f(S^i) - f(S^{i-1})}{c(u_i)} \geq \frac{f(X \setminus S^{i-1} \mid S^{i-1})}{c(X \setminus S^{i-1})} \geq \frac{f(X) - f(S^{i-1})}{c(X)}. \tag{1}$$

The last inequality holds since f is monotone and $c(X \setminus S^{i-1}) \leq c(X)$.

Next, we assume that $f(X) > f(S^\ell)$, since otherwise the lemma already holds. Under this assumption, it must hold that $c(u_i) < c(X)$, since otherwise inequality (1) implies that $f(X) \leq f(S^i) \leq f(S^\ell)$. A contradiction! Now we can rearrange inequality (1) and obtain that

$$f(X) - f(S^i) \leq \left(1 - \frac{c(u_i)}{c(X)}\right)(f(X) - f(S^{i-1})).$$

By expanding the recurrence, we have

$$f(X) - f(S^i) \leq \prod_{j=1}^{i} \left(1 - \frac{c(u_j)}{c(X)}\right) \cdot f(X) \leq \prod_{j=1}^{i} e^{-\frac{c(u_j)}{c(X)}} \cdot f(X) = e^{-\frac{c(S^i)}{c(X)}} \cdot f(X).$$

The second inequality holds due to $e^x \geq 1 + x$. Hence we have

$$f(S^i) \geq \left(1 - e^{-c(S^i)/c(X)}\right) \cdot f(X).$$

The lemma follows by plugging $i = \ell$ into it. ◀

The above lemma immediately leads to the following corollary.

▶ **Corollary 3.** *The set S returned by Algorithm 1 satisfies $f(S) \geq (1 - e^{-1}) \cdot f(OPT)$.*

Proof. If $c(S) < \sum_{j=1}^m B_j$, there is some j such that $c(S_j) < B_j$. It means that Algorithm 1 ended with $S = N$. Thus, the corollary follows by monotonicity. If $c(S) \geq \sum_{j=1}^m B_j$, then $c(S) \geq c(OPT)$. The corollary follows from Lemma 2. ◀

3 The identical Case

In this section, we present a deterministic $(1 - e^{-1} - \epsilon)$ approximation algorithm for SMKP under the identical case, where all bins have the same budget. Our algorithm is depicted as Algorithm 2 and works when $m \geq 1/(4\epsilon^3)$. It packs bins one by one and manages to make each bin pack at least the average marginal value of the optimal solution over m bins. In other words, for the j-th bin, it aims to find a set S_j such that $f(S_j \mid \cup_{i=1}^{j-1} S_i) \geq \frac{1}{m} f(OPT \mid \cup_{i=1}^{j-1} S_i)$. This naturally leads to $(1 - e^{-1})$ approximation. For this purpose, Algorithm 2 divides elements into *large* and *small* elements according to their costs. Given input ϵ, an element $u \in N$ is large if $c(u) > \epsilon B$ and small otherwise. Let $N_\ell = \{u \in N \mid c(u) > \epsilon B\}$ be the set of large elements and $N_s = N \setminus N_\ell$. For the j-th bin, Algorithm 2 first enumerates all feasible subsets of large elements. Then, for every such subset, Algorithm 1 is invoked over small elements to augment it. Finally, the one with the maximum marginal value is assigned to S_j.

Due to the call of Algorithm 1, S_j might contain a reserved element, which is the last added into S_j and violates the budget. To remedy this issue, Algorithm 2 divides the bins into two classes: the first $(1 - \epsilon)m$ bins are called *working bins* and the last ϵm bins are

Algorithm 2 IDENTICAL-CASE.

```
   Input: elements N, budget B, number of bins m, profit f, cost c, constant ϵ > 0.
 1 Let the first (1 − ϵ)m bins be working bins and the last ϵm bins be reserved bins.
 2 Define N_ℓ = {u ∈ N | c(u) > ϵB} and let N_s = N \ N_ℓ.
 3 S_j = ∅ for 1 ≤ j ≤ m and T = ∪_{j=1}^m S_j.
 4 for j = 1 to (1 − ϵ)m do
 5 |  foreach subset E ⊆ N_ℓ such that c(E) ≤ B do
 6 |  |  G_E = GREEDY(N_s, B − c(E), f(· | T ∪ E), c(·)).
 7 |  end
 8 |  S_j = arg max_E f(E ∪ G_E | T) and T = ∪_{j=1}^m S_j.
 9 end
10 Repack the reserved elements in T into the reserved bins.
11 return T = ∪_{j=1}^m S_j.
```

called *reserved bins*. The procedure described above only proceeds with the working bins. After that, Algorithm 2 repacks all reserved elements into the reserved bins. We will show that in this way, Algorithm 2 produces a feasible solution and the loss of the profit is little even if it does not use the reserved bins to pack new elements.

We now give an analysis of Algorithm 2. For $1 \leq j \leq (1-\epsilon)m$, let S_j be defined as in line 8 of Algorithm 2 and $T_j = \cup_{i=1}^{j} S_i$. We first show that Algorithm 2 returns a feasible solution.

▶ **Lemma 4.** *Algorithm 2 produces a feasible solution.*

Proof. For $1 \leq j \leq (1-\epsilon)m$, observe that each S_j contains at most one reserved element due to the call of Algorithm 1. By repacking those reserved elements into the reserved bins, each S_j becomes feasible. Besides, the cost of each reserved element is at most ϵB since it is a small element. Thus, a reserved bin can pack at least $1/\epsilon$ reserved elements. Then, ϵm reserved bins can pack $m > (1-\epsilon)m$ reserved elements without exceeding their budgets. Therefore, Algorithm 2 produces a feasible solution. ◀

Next, we present Lemma 5 for Algorithm 2.

▶ **Lemma 5.** *Assume that $m \geq 1/(4\epsilon^3)$. For every $1 \leq j \leq (1-\epsilon)m$,*

$$f(S_j \mid T_{j-1}) \geq \frac{1-2\epsilon}{m} \cdot f(OPT \mid T_{j-1}).$$

Proof. For the sake of description, we define $g(\cdot) = f(\cdot \mid T_{j-1})$ and the lemma becomes $g(S_j) \geq \frac{1-2\epsilon}{m} \cdot g(OPT)$. Let $OPT_\ell = OPT \cap N_\ell$ and $OPT_s = OPT \setminus OPT_\ell$. We prove the lemma by case analysis, according to the cost and density of OPT_s.

Case 1: $c(OPT_s) \geq \epsilon m B$, namely OPT_s has a large cost. Let $OPT_\ell = \cup_{j=1}^m OPT_{\ell,j}$ and $OPT_s = \cup_{j=1}^m OPT_{s,j}$, where $OPT_{\ell,j}$ and $OPT_{s,j}$ are the large and small elements packed in the j-th bin, respectively. For each $1 \leq j \leq m$, since $c(OPT_{\ell,j}) \leq B$, $OPT_{\ell,j}$ will be enumerated during the **foreach** loop. Let G_j be the output of GREEDY (Algorithm 1) starting from $OPT_{\ell,j}$. We will show that one of $OPT_{\ell,j} \cup G_j$ satisfies the lemma.

If $c(G_j) < B - c(OPT_{\ell,j})$, it means that Algorithm 1 ended with $G_j = OPT_s$ and therefore $g(G_j \mid OPT_{\ell,j}) = g(OPT_s \mid OPT_{\ell,j})$. If $c(G_j) \geq B - c(OPT_{\ell,j})$, then $c(G_j) \geq c(OPT_{s,j})$.

By Lemma 2,

$$\begin{aligned}
g(G_j \mid OPT_{\ell,j}) &\geq \left(1 - e^{-c(G_j)/c(OPT_s)}\right) \cdot g(OPT_s \mid OPT_{\ell,j}) \\
&\geq \left(1 - e^{-c(OPT_{s,j})/c(OPT_s)}\right) \cdot g(OPT_s \mid OPT_{\ell,j}) \\
&\geq \left(\frac{c(OPT_{s,j})}{c(OPT_s)} - \frac{c(OPT_{s,j})^2}{2 \cdot c(OPT_s)^2}\right) \cdot g(OPT_s \mid OPT_{\ell,j}) \\
&\geq \left(\frac{c(OPT_{s,j})}{c(OPT_s)} - \frac{1}{2\epsilon^2 m^2}\right) \cdot g(OPT_s \mid OPT_{\ell,j}) \\
&\geq \left(\frac{c(OPT_{s,j})}{c(OPT_s)} - \frac{2\epsilon}{m}\right) \cdot g(OPT_s \mid OPT_{\ell,j}).
\end{aligned}$$

The third inequality holds since $1 - e^{-x} \geq x - x^2/2$ for $x \geq 0$. The fourth inequality holds since $c(OPT_{s,j})/c(OPT_s) \leq 1/(\epsilon m)$. The last inequality holds since $m \geq 1/(4\epsilon^3)$. By adding $g(OPT_{\ell,j})$ on both sides of the last inequality and summing over j,

$$\begin{aligned}
\sum_{j=1}^{m} g(OPT_{\ell,j} \cup G_j) &\geq \sum_{j=1}^{m} \left(\frac{c(OPT_{s,j})}{c(OPT_s)} - \frac{2\epsilon}{m}\right) \cdot g(OPT_s \mid OPT_{\ell,j}) + \sum_{j=1}^{m} g(OPT_{\ell,j}) \\
&\geq \sum_{j=1}^{m} \left(\frac{c(OPT_{s,j})}{c(OPT_s)} - \frac{2\epsilon}{m}\right) \cdot g(OPT_s \mid OPT_\ell) + g(OPT_\ell) \\
&= (1 - 2\epsilon) \cdot g(OPT_s \mid OPT_\ell) + g(OPT_\ell) \\
&\geq (1 - 2\epsilon) \cdot g(OPT).
\end{aligned}$$

Hence, the maximum of $OPT_{\ell,j} \cup G_j$ satisfies the lemma and so does S_j.

Case 2: $g(OPT_s) \geq (1 - e^{-B/c(OPT_s)})^{-1} \cdot \frac{g(OPT)}{m}$, namely the density of OPT_s is large. Consider one of the iterations of **foreach** loop where $E = \emptyset$. Note that it is augmented by $G_\emptyset$ via GREEDY (Algorithm 1). If $c(G_\emptyset) < B$, it means that Algorithm 1 ended with $G_\emptyset = OPT_s$. Then,

$$g(G_\emptyset) = g(OPT_s) \geq (1 - e^{-B/c(OPT_s)})^{-1} \cdot \frac{g(OPT)}{m} \geq \frac{g(OPT)}{m}.$$

If $c(G_\emptyset) \geq B$, by Lemma 2,

$$g(G_\emptyset) \geq \left(1 - e^{-B/c(OPT_s)}\right) \cdot g(OPT_s) \geq \frac{g(OPT)}{m}.$$

This implies that $G_\emptyset$ satisfies the lemma and so does S_j.

Case 3: $c(OPT_s) < \epsilon m B$ and $g(OPT_s) < \left(1 - e^{-B/c(OPT_s)}\right)^{-1} \cdot \frac{g(OPT)}{m}$, namely both the cost and density of OPT_s are small. We show that OPT_s only contributes a negligible value in OPT:

$$\begin{aligned}
g(OPT_s) &< (1 - e^{-1/\epsilon m})^{-1} \cdot \frac{g(OPT)}{m} \leq \left(\frac{1}{\epsilon m} - \frac{1}{2\epsilon^2 m^2}\right)^{-1} \frac{g(OPT)}{m} \\
&\leq \left(\frac{1}{2\epsilon m}\right)^{-1} \frac{g(OPT)}{m} = 2\epsilon \cdot g(OPT).
\end{aligned}$$

The first inequality holds since $(1 - e^{-B/x})^{-1}$ is monotone increasing. The second holds since $1 - e^{-x} \geq x - x^2/2$ for $x \geq 0$. The third holds as long as $m \geq 1/\epsilon$. Hence, by submodularity,

$$g(OPT_\ell) \geq g(OPT) - g(OPT_s) \geq (1 - 2\epsilon) \cdot g(OPT),$$

and

$$\frac{1}{m}\sum_{j=1}^{m} g(OPT_{\ell,j}) \geq \frac{1}{m}\cdot g(OPT_\ell) \geq \frac{1-2\epsilon}{m}\cdot g(OPT).$$

This implies that the maximum of $OPT_{\ell,j}$ satisfies the lemma and so does S_j. ◀

By expanding the recurrence in Lemma 5, we have

▶ **Lemma 6.** *Assume that $m \geq 1/(4\epsilon^3)$. For every $1 \leq j \leq (1-\epsilon)m$,*

$$f(T_j) \geq (1 - e^{-j(1-2\epsilon)/m}) \cdot f(OPT).$$

Proof. By Lemma 5, for $1 \leq j \leq (1-\epsilon)m$,

$$f(S_j \mid T_{j-1}) \geq \frac{1-2\epsilon}{m}\cdot f(OPT \mid T_{j-1}).$$

By monotonicity of f,

$$f(T_j) - f(T_{j-1}) \geq \frac{1-2\epsilon}{m}\cdot (f(OPT) - f(T_{j-1})).$$

By rearranging the above inequality,

$$\left(1 - \frac{1-2\epsilon}{m}\right)(f(OPT) - f(T_{j-1})) \geq f(OPT) - f(T_j).$$

By expanding the recurrence,

$$f(OPT) - f(T_j) \leq \left(1 - \frac{1-2\epsilon}{m}\right)^j f(OPT) \leq e^{-j(1-2\epsilon)/m}\cdot f(OPT).$$

The last inequality holds since $e^{-x} \geq 1 - x$. Thus, we have

$$f(T_j) \geq (1 - e^{-j(1-2\epsilon)/m}) \cdot f(OPT).$$ ◀

We now provide a theoretical guarantee for Algorithm 2.

▶ **Theorem 7.** *When $m \geq 1/(4\epsilon^3)$, Algorithm 2 achieves a $(1 - e^{-1} - O(\epsilon))$ approximation ratio and uses $O(mn^{3+1/\epsilon})$ queries.*

Proof. For the approximation ratio, by plugging $j = (1-\epsilon)m$ into Lemma 6,

$$f(T_{(1-\epsilon)m}) \geq (1 - e^{-(1-\epsilon)(1-2\epsilon)}) \cdot f(OPT).$$

For the query complexity, observe that during the **foreach** loop, the number of subsets $E \subseteq N_\ell$ such that $c(E) \leq B$ is at most

$$\sum_{i=0}^{1/\epsilon}\binom{n}{i} = O(n^{1/\epsilon+1}).$$

Since each E is augmented via Greedy, which uses $O(n^2)$ queries, the **foreach** loop uses $O(n^{1/\epsilon+3})$ in total. Then, Algorithm 2 overall uses $O(mn^{3+1/\epsilon})$ queries. ◀

4 The General Case

In this section, we present a deterministic $(1 - e^{-1} - \epsilon)$ approximation algorithm for solving general SMKP instances. A key difficulty is that the budgets of bins are distinct, which makes our technique for the identical case inapplicable. In Section 4.1, we introduce a grouping technique from [9], which reshapes any SMKP instance such that bins can be partitioned into *blocks* and almost every block contains sufficiently many bins with the same budget. Next, in Section 4.2, we show how one can design a nearly optimal algorithm for such instances.

4.1 Reshape the Instance

We first introduce a grouping technique from [9] to reshape any SMKP instance as follows.

▶ **Definition 8.** *A subset of bins $M' \subseteq M$ is called a block if for any $i, j \in M'$, $B_i = B_j$.*

▶ **Definition 9.** *For any $t \in \mathbb{N}_+$, a partition $\{M_k\}_{k=1}^r$ of bins M is t-leveled if for every $1 \le k \le r$, M_k is a block and $|M_k| = t^{\lfloor (k-1)/t^2 \rfloor}$.*

To gain some intuition, note that for every $1 \le k \le t^2$, block M_k contains a single bin, and for every $t^2 < k \le 2t^2$, block M_k contains t bins, etc. It follows that except for the first t^2 blocks, each of the remaining blocks contains at least t bins with the same budget.

▶ **Lemma 10** ([9]). *There is a polynomial-time algorithm, referred to as* Block*, that takes a set of bins M, budgets $\{B_j\}_{j \in M}$ and a parameter $t \in \mathbb{N}_+$ as input, and returns a new set of bins $\widetilde{M} \subseteq M$, budgets $\{\widetilde{B}_j\}_{j \in \widetilde{M}}$ and a t-leveled partition $\{\widetilde{M}_k\}_{k=1}^r$ of bins $\widetilde{M}$ such that*

- *For every $j \in \widetilde{M}$, $\widetilde{B}_j \le B_j$.*
- *For any SMKP instance $(N, M, \{B_j\}_{j \in M}, f, c)$ and a feasible solution $\{S_j\}_{j \in M}$ for it, there exists a feasible solution $\{\widetilde{S}_j\}_{j \in \widetilde{M}}$ for instance $(N, \widetilde{M}, \{\widetilde{B}_j\}_{j \in \widetilde{M}}, f, c)$ such that $f(\cup_{j \in \widetilde{M}} \widetilde{S}_j) \ge \left(1 - \frac{1}{t}\right) f(\cup_{j \in M} S_j)$ and $\cup_{j \in \widetilde{M}} \widetilde{S}_j \subseteq \cup_{j \in M} S_j$.*

The instance $(N, \widetilde{M}, \{\widetilde{B}_j\}_{j \in \widetilde{M}}, f, c)$ is called *t-leveled*. Lemma 10 tells us that any feasible solution for it is also feasible for the original instance $(N, M, \{B_j\}_{j \in M}, f, c)$, and an optimal solution for it causes a small loss in the profit.

4.2 The Final Algorithm

Now, we explain how one can design a nearly optimal algorithm for a t-leveled SMKP instance with bins $\widetilde{M}$, budgets $\{\widetilde{B}_j\}_{j \in \widetilde{M}}$ and a t-leveled partition $\{\widetilde{M}_k\}_{k=1}^r$ of $\widetilde{M}$.

For $t^2 < k \le r$, block $\widetilde{M}_k$ contains $|\widetilde{M}_k| \ge t$ bins with the same budget $\widetilde{B}_k$. The problem restricted to each block $\widetilde{M}_k$ can be regarded as an SMKP instance under the identical case. Thus, a natural idea is to pack each block $\widetilde{M}_k$ in sequence by invoking Algorithm 2. However, we fail to get an optimal approximation via this procedure. Instead, we develop a technique that is inspired by [1]. We run $1/\epsilon$ iterations in total (assume that $1/\epsilon$ is an integer). In each iteration, we pack each block $\widetilde{M}_k$ in sequence but only pack $\epsilon|\widetilde{M}_k|$ bins (assume that $\epsilon|\widetilde{M}_k|$ is an integer). This forms an instance under the identical case with $\epsilon|\widetilde{M}_k|$ bins and therefore we can invoke Algorithm 2 to solve it.

For $1 \le k \le t^2$, block $\widetilde{M}_k$ contains a single bin with budget $\widetilde{B}_k$. Basically, we can use Greedy to pack elements. Likewise, we do not use the full budget at a time. Instead, we also run $1/\epsilon$ iterations. In each iteration, we pack elements using budget $(\epsilon - \epsilon^2)\widetilde{B}_k$. To avoid exceeding the budget, we only pack small elements u satisfying $c(u) \le \epsilon^2 \widetilde{B}_k$. To ensure this, we need to enumerate large-valued and large-cost elements in this bin. The overall procedure is depicted as Algorithm 3.

Algorithm 3 The Final Algorithm for SMKP.

Input: elements N, bins M, budgets $\{B_j\}_{j\in M}$, profit f, cost c, constant $\epsilon > 0$.
Let $s = 1/(16\epsilon^9)$ and $t = 1/(4\epsilon^3)$.
Let $\mathcal{C} = \emptyset$.
$(\widetilde{M}, \{\widetilde{B}_j\}_{j\in\widetilde{M}}, \{\widetilde{M}_k\}_{k=0}^{r}) = \textsc{Block}(M, \{B_j\}_{j\in M}, t)$.
foreach *feasible solution* $\{E_j\}_{j=1}^{t^2}$ *such that* $|\cup_{j=1}^{m} E_j| \le s$ **do** \\ $E_j = \emptyset$ for $j > t^2$
  Let $E = \cup_{j=1}^{t^2} E_j$.
  Let $S_j = E_j$ for $1 \le j \le t^2$ and $S_j = \emptyset$ for $t^2 < j \le k$.
  for $i = 1$ *to* $1/\epsilon$ **do**
    for $k = 1$ *to* r **do** \\ handle blocks one by one
      if $k \le t^2$ **then** \\ each block contains a single bin
        Let $D = \{u \in N \setminus E \mid f(u \mid E) > \frac{1}{s} \cdot f(E)\}$.
        Let $L_k = \{u \in N \setminus E \mid c(u) > \epsilon^2(\widetilde{B}_k - c(E_k))\}$.
        $R_k = \textsc{Greedy}(N \setminus (E \cup D \cup L_k), (\epsilon - \epsilon^2)(\widetilde{B}_k - c(E_k)), f(\cdot \mid \cup_{j=1}^{m} S_j), c(\cdot))$.
        $S_{k+1} = S_{k+1} \cup R_k$.
      else \\ each block contains $\ge t$ bins
        $\{R_j\}_{j\in\widetilde{M}_k} = \textsc{Identical-case}(N \setminus E, \widetilde{B}_k, \epsilon|\widetilde{M}_k|, f(\cdot \mid \cup_{j=1}^{m} S_j), c(\cdot), \epsilon)$.
        $S_j = S_j \cup R_j$ for $j \in \widetilde{M}_k$.
      end
    end
  end
  $\mathcal{C} = \mathcal{C} \cup \{\{S_j\}_{j=1}^{m}\}$.
end
return $\arg\max\{f(\cup_{j=1}^{m} S_j) \mid \{S_j\}_{j=1}^{m} \in \mathcal{C}\}$.

▶ **Theorem 11.** *Algorithm 3 achieves a $1 - e^{-1} - O(\epsilon)$ approximation ratio and uses a polynomial number of queries.*

Proof. Let $\widetilde{OPT} = \cup_{j=1}^{m} \widetilde{OPT}_j$ be the optimal solution of the SMKP instance with bins $\widetilde{M}$ and budgets $\{\widetilde{B}_j\}_{j\in\widetilde{M}}$. Let $OPT' = \cup_{j=1}^{t^2} \widetilde{OPT}_j$. Order elements in OPT' greedily according to their marginal values such that $o_1 = \arg\max_{o\in OPT'} f(o)$, $o_2 = \arg\max_{o\in OPT'\setminus\{o_1\}} f(o \mid o_1)$, etc. Denote by E the first s elements in OPT' (if $|OPT'| < s$, then $E = OPT'$). Let $E_j = E \cap \widetilde{OPT}_j$ for $1 \le j \le t^2$. Then, $\{E_j\}_{j=1}^{t^2}$ will be enumerated during the **foreach** loop. In the following, we focus on this particular set.

Let $D = \{u \in N \mid f(u \mid E) > \frac{1}{s} \cdot f(E)\}$. Since E is the first s elements in OPT', we have $f(o \mid E) \le \frac{1}{s} \cdot f(E)$ for any $o \in OPT' \setminus E$. Thus, $D \cap (OPT' \setminus E) = \emptyset$ and therefore $OPT' \setminus E$ will not be excluded from the execution of Greedy over $N \setminus (E \cup D)$. Besides, $\{\widetilde{OPT}_j \setminus E_j\}_{j=1}^{t^2}$ is a feasible solution given budgets $\{\widetilde{B}_j - c(E_j)\}_{j=1}^{t^2}$.

For $1 \le j \le t^2$, let $L_j = \{u \in N \setminus E \mid c(u) > \epsilon^2(\widetilde{B}_j - c(E_j))\}$. Define OPT^* as follows. For $1 \le j \le t^2$, $OPT_j^* = \widetilde{OPT}_j \setminus L_j$. For $j > t^2$, $OPT_j^* = \widetilde{OPT}_j$. Then, $OPT^* = \cup_{j=1}^{m} OPT_j^*$. We have

$$\begin{aligned} f(OPT^* \mid E) &= f((\cup_{j=1}^{t^2} \widetilde{OPT}_j \setminus L_j) \cup (\cup_{j=t^2+1}^{m} \widetilde{OPT}_j) \mid E) \\ &\ge f(\cup_{j=1}^{m} \widetilde{OPT}_j \mid E) - f(\cup_{j=1}^{t^2} \widetilde{OPT}_j \cap L_j \mid E) \end{aligned}$$

$$\begin{aligned}
&\geq f(\widetilde{OPT} \mid E) - \sum_{j=1}^{t^2} \sum_{u \in (\widetilde{OPT}_j \setminus E) \cap L_j} f(u \mid E) \\
&\geq f(\widetilde{OPT} \mid E) - \frac{t^2}{\epsilon^2 s} \cdot f(E) \\
&= f(\widetilde{OPT} \mid E) - \epsilon \cdot f(E).
\end{aligned}$$

The first two inequalities are due to submodularity. The third inequality holds since by definition of L_j, $\widetilde{OPT}_j \setminus E$ contains at most $1/\epsilon^2$ elements in L_j, and $f(u \mid E) \leq \frac{1}{s} f(E)$ due to $D \cap (\widetilde{OPT}_j \setminus E) = \emptyset$. The last equality follows from the choices of t and s. This implies that invoking Greedy over $N \setminus (E \cup D \cup L_j)$ for $1 \leq j \leq t^2$ only incurs little loss in the profit.

Now we are prepared to provide a theoretical bound for Algorithm 3. Let $g(\cdot) = f(\cdot \mid E)$. For $1 \leq i \leq 1/\epsilon$ and $1 \leq k \leq r$, let R_{ik} be the set returned in line 12 if $k \leq t^2$ and $R_{ik} = \cup_{j \in \widetilde{M}_k} R_j$ otherwise, where $\{R_j\}_{j \in \widetilde{M}_k}$ is the set returned in line 15. Then, the k-th block $\widetilde{M}_k$ packs $\cup_{i=1}^{1/\epsilon} R_{ik}$ by the end of Algorithm 3. Define $T_0 = \emptyset$ and $T_i = T_{i-1} \cup (\cup_{k=1}^{r} R_{ik})$ for $1 \leq i \leq 1/\epsilon$.

For $1 \leq i \leq 1/\epsilon$ and $1 \leq k \leq t^2$, by Lemma 2,

$$\begin{aligned}
g(R_{ik} \mid T_{i-1} \cup (\cup_{k'=1}^{k-1} R_{ik'})) &\geq (1 - e^{-(\epsilon - \epsilon^2)}) \cdot g(OPT_k^* \mid T_{i-1} \cup (\cup_{k'=1}^{k-1} R_{ik'})) \\
&\geq (\epsilon - 2\epsilon^2) \cdot g(OPT_k^* \mid T_i).
\end{aligned}$$

The last inequality holds due to $1 - e^{-x} \geq x - x^2/2$ for $x \geq 0$ and submodularity.

For $1 \leq i \leq 1/\epsilon$ and $t^2 < k \leq r$, by Lemma 6,

$$\begin{aligned}
g(R_{ik} \mid T_{i-1} \cup (\cup_{k'=1}^{k-1} R_{ik'})) &\geq (1 - e^{-\epsilon(1-2\epsilon)}) \cdot g(OPT_k^* \mid T_{i-1} \cup (\cup_{k'=1}^{k-1} R_{ik'})) \\
&\geq (\epsilon - 3\epsilon^2) \cdot g(OPT_k^* \mid T_i).
\end{aligned}$$

Again, the last inequality holds due to $1 - e^{-x} \geq x - x^2/2$ for $x \geq 0$ and submodularity.

Summing up over $1 \leq k \leq r$, we have

$$\begin{aligned}
g(T_i) - g(T_{i-1}) &= \sum_{k=1}^{r} g(R_{ik} \mid T_{i-1} \cup (\cup_{k'=1}^{k-1} R_{ik'})) \\
&\geq \sum_{k=1}^{r} (\epsilon - 3\epsilon^2) \cdot g(OPT_k^* \mid T_i) \\
&\geq (\epsilon - 3\epsilon^2) \cdot g(OPT^* \mid T_i) \\
&\geq (\epsilon - 3\epsilon^2) \cdot (g(OPT^*) - g(T_i)).
\end{aligned}$$

The last two inequalities are due to submodularity and monotonicity, respectively. By adding $g(OPT^*)$ to both sides and move $g(T_i)$ to the right in the above inequality,

$$g(OPT^*) - g(T_{i-1}) \geq (1 + \epsilon - 3\epsilon^2)(g(OPT^*) - g(T_i)).$$

This leads to

$$g(OPT^*) - g(T_i) \leq \frac{1}{(1 + \epsilon - 3\epsilon^2)^i} \cdot g(OPT^*).$$

Hence, by plugging $i = 1/\epsilon$,

$$\begin{aligned}
g(T_{1/\epsilon}) &\geq \left(1 - \frac{1}{(1 + \epsilon - 3\epsilon^2)^{1/\epsilon}}\right) \cdot g(OPT^*) = \left(1 - e^{-\frac{1}{\epsilon} \ln(1+\epsilon-3\epsilon^2)}\right) \cdot g(OPT^*) \\
&\geq (1 - e^{-\frac{1}{\epsilon}(\epsilon - 3\epsilon^2 - (\epsilon - 3\epsilon^2)^2/2)}) \cdot g(OPT^*) \geq (1 - e^{-1} - O(\epsilon)) \cdot g(OPT^*).
\end{aligned}$$

The second inequality holds since $\ln(1+x) \geq x - x^2/2$ for $x > 0$. Finally, recall that $g(\cdot) = f(\cdot \mid E)$, we have

$$\begin{aligned} f(T_{1/\epsilon}) &= f(E) + f(T_{1/\epsilon} \mid E) \geq f(E) + (1 - e^{-1} - O(\epsilon)) \cdot f(OPT^* \mid E) \\ &\geq f(E) + (1 - e^{-1} - O(\epsilon)) \cdot (f(\widetilde{OPT} \mid E) - \epsilon f(E)) \\ &\geq (1 - e^{-1} - O(\epsilon)) \cdot f(OPT). \end{aligned}$$ ◀

5 Conclusion

In this paper, we present a deterministic $(1 - e^{-1} - \epsilon)$-approximation algorithm for SMKP. Our algorithm is inspired by the viewpoint regarding SMKP instances as exponential-size instances of submodular maximization subject to a cardinality or partition matroid constraint. Thus our algorithm is conceptually much simpler than that of Fairstein et al. [9].

As pointed out by [9], it remains open to remove the loss of ϵ in the approximation ratio. As a first step, we present a $(1 - e^{-1})$-approximation algorithm for SMKP when the number of bins m is constant in the full version of this paper. Recently, a randomized 0.385-approximation algorithm for *non-monotone* SMKP was proposed in [10]. It is an interesting question to design deterministic algorithms for this problem.

References

1 Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1497–1514, 2014. doi:10.1137/1.9781611973402.110.

2 Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 392–403, 2016. doi:10.1137/1.9781611974331.ch29.

3 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 649–658, 2012. doi:10.1109/FOCS.2012.73.

4 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1433–1452, 2014. doi:10.1137/1.9781611973402.106.

5 Chandra Chekuri and Sanjeev Khanna. A PTAS for the multiple knapsack problem. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 213–222, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338254.

6 Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within 1+epsilon in linear time. *Combinatorica*, 1(4):349–355, 1981. doi:10.1007/BF02579456.

7 Alina Ene and Huy L. Nguyen. Constrained submodular maximization: Beyond 1/e. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 248–257, 2016. doi:10.1109/FOCS.2016.34.

8 Alina Ene and Huy L. Nguyen. A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 53:1–53:12, 2019. doi:10.4230/LIPIcs.ICALP.2019.53.

9 Yaron Fairstein, Ariel Kulik, Joseph (Seffi) Naor, Danny Raz, and Hadas Shachnai. A (1-e^{-1}-ϵ)-approximation for the monotone submodular multiple knapsack problem. In *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 44:1–44:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.44.

10 Yaron Fairstein, Ariel Kulik, and Hadas Shachnai. Modular and submodular optimization with multiple knapsack constraints via fractional grouping. In *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.41.

11 Moran Feldman. *Maximization problems with submodular objective functions*. Technion-Israel Institute of Technology, Faculty of Computer Science, 2013.

12 Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 570–579, 2011. doi:10.1109/FOCS.2011.46.

13 Moran Feldman, Zeev Nutov, and Elad Shoham. Practical budgeted submodular maximization. *Algorithmica*, 85(5):1332–1371, 2023. doi:10.1007/s00453-022-01071-2.

14 Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 659–668, 2012. doi:10.1109/FOCS.2012.55.

15 Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. An analysis of approximations for maximizing submodular set functions - II. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

16 Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1098–1116, 2011. doi:10.1137/1.9781611973082.83.

17 Klaus Jansen. Parameterized approximation scheme for the multiple knapsack problem. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 665–674, 2009. doi:10.1137/1.9781611973068.73.

18 Hans Kellerer. A polynomial time approximation scheme for the multiple knapsack problem. In *Randomization, Approximation, and Combinatorial Algorithms and Techniques, RANDOM-APPROX'99, Berkeley, CA, USA, August 8-11, 1999, Proceedings*, pages 51–62, 1999. doi:10.1007/978-3-540-48413-4_6.

19 David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 137–146. ACM, 2003. doi:10.1145/956750.956769.

20 Ariel Kulik, Roy Schwartz, and Hadas Shachnai. A refined analysis of submodular greedy. *Oper. Res. Lett.*, 49(4):507–514, 2021. doi:10.1016/j.orl.2021.04.006.

21 Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 323–332. ACM, 2009. doi:10.1145/1536414.1536459.

22 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Math. Oper. Res.*, 35(4):795–806, 2010. doi:10.1287/moor.1100.0463.

23 George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978. `doi:10.1287/moor.3.3.177`.

24 George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Math. Program.*, 14(1):265–294, 1978. `doi:10.1007/BF01588971`.

25 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004. `doi:10.1016/S0167-6377(03)00062-2`.

26 Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 67–74, 2008. `doi:10.1145/1374376.1374389`.

27 Jan Vondrák, Chandra Chekuri, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 783–792, 2011. `doi:10.1145/1993636.1993740`.

28 Yuichi Yoshida. Maximizing a monotone submodular function with a bounded curvature under a knapsack constraint. *SIAM J. Discret. Math.*, 33(3):1452–1471, 2019. `doi:10.1137/16M1107644`.