SIAM J. DISCRETE MATH.
Vol. 33, No. 3, pp. 1121–1145

SUBMODULAR MAXIMIZATION WITH UNCERTAIN KNAPSACK CAPACITY*

YASUSHI KAWASE†, HANNA SUMITA‡, AND TAKURO FUKUNAGA§

Abstract. We consider the maximization problem of monotone submodular functions under an uncertain knapsack constraint. Specifically, the problem is discussed in the situation where the knapsack capacity is not given explicitly and can be accessed only through an oracle that answers whether or not the current solution is feasible when an item is added to the solution. Assuming that cancellation of the last item is allowed when it overflows the knapsack capacity, we discuss the robustness ratios of adaptive policies for this problem, which are the worst case ratios of the objective values achieved by the output solutions to the optimal objective values. We present a randomized policy of robustness ratio $(1-1/e)/2$ and a deterministic policy of robustness ratio $2(1-1/e)/21$. We also consider a universal policy that chooses items following a precomputed sequence. We present a randomized universal policy of robustness ratio $(1-1/\sqrt[4]{e})/2$. When cancellation is not allowed, no randomized adaptive policy achieves a constant robustness ratio. Because of this hardness, we assume that a probability distribution of the knapsack capacity is given, and we consider computing a sequence of items that maximizes the expected objective value. We present a polynomial time randomized algorithm of approximation ratio $(1-1/\sqrt[4]{e})/4-\epsilon$ for any small constant $\epsilon > 0$.

Key words. submodular maximization, knapsack problem, robust optimization

AMS subject classifications. 68W25, 90C27, 68R05

DOI. 10.1137/18M1174428

1. Introduction.

Submodular maximization is one of the most well-studied combinatorial optimization problems. Since it captures an essential part of decision-making situations, it has a huge number of applications in diverse areas of computer science. Nevertheless, the standard setting of the submodular maximization problem fails to capture several realistic situations. For example, let us consider choosing several items to maximize a reward represented by a submodular function subject to a resource limitation. When the amount of the available resource is exactly known, this problem is formulated as the submodular maximization problem with a knapsack constraint. However, in many practical cases, precise information on the available resource is not given. Thus, algorithms for the standard submodular maximization problem cannot be applied to this situation. Motivated by this fact, we study robust maximization of submodular functions with an uncertain knapsack capacity. Besides its practical applications, this problem is interesting to study because it shows how much robustness can be achieved for an uncertain knapsack capacity in submodular maximization.

More specifically, we study the *submodular maximization problem with an*

*Received by the editors March 8, 2018; accepted for publication (in revised form) March 4, 2019; published electronically July 2, 2019. A preliminary version of this paper was presented at the 13th Latin American Theoretical Informatics Symposium (LATIN 2018).

https://doi.org/10.1137/18M1174428

Funding: The first author was supported by JSPS KAKENHI grant JP16K16005, Japan. The second author was supported by JSPS KAKENHI grant JP17K12646 and JST ERATO grant JPMJER1201, Japan. The third author was supported by JSPS KAKENHI grant JP17K00040, JST ERATO grant JPMJER1201, and JST PRESTO grant JPMJPR1759, Japan.

†Tokyo Institute of Technology, Tokyo 152-8550, Japan (kawase.y.ab@m.titech.ac.jp).

‡Tokyo Metropolitan University, Tokyo 192-0364, Japan (sumita@tmu.ac.jp).

§RIKEN Center for Advanced Intelligence Project, Tokyo 103-0027, Japan, and JST PRESTO, Tokyo 102-0076, Japan (takuro.fukunaga@riken.jp).

unknown knapsack capacity (SMPUC). In this problem, we are given a set I of items and a monotone nonnegative submodular function $f\colon 2^I \to \mathbb{R}_+$ such that $f(\emptyset) = 0$, where each item $i \in I$ is associated with a size $s(i)$. The objective is to find a set of items that maximizes the submodular function subject to a knapsack constraint, but we assume that the knapsack capacity is unknown. We have access to the knapsack capacity through an oracle; we add items to the knapsack one by one, and we see whether or not the selected item violates the knapsack constraint only after the addition. If a selected item fits the knapsack, the selection of this item is irrevocable. When the total size of the selected items exceeds the capacity, there are two settings depending on whether or not the last selection can be canceled. If cancellation is allowed, then we remove the last selected item from the knapsack and continue adding the remaining items to the knapsack. In the other setting, we stop the selection, and the final output is defined as the item set in the knapsack before adding the last item.

For the setting where cancellation is allowed, we consider an *adaptive policy*, which is defined as a decision tree to decide which item to pack into the knapsack next. A *randomized* policy is a probability distribution over adaptive policies. Performance of an adaptive policy is evaluated by the robustness ratio defined as follows. For any number $C \in \mathbb{R}_+$, let OPT_C denote the optimal item set when the capacity is C, and let ALG_C denote an output of the policy. Note that if the policy is a randomized one, then ALG_C is a random variable. We call the adaptive policy *α-robust*, for some $\alpha \le 1$, if for any $C \in \mathbb{R}_+$, the expected objective value of the policy's output is within a ratio α of the optimal value, i.e., $\mathbb{E}[f(\mathrm{ALG}_C)]/f(\mathrm{OPT}_C) \ge \alpha$. We also call the ratio α the *robustness ratio* of the policy.

One main purpose of this paper is to present algorithms that produce adaptive policies of constant robustness ratios for SMPUC. Moreover, we consider a *universal* policy, which selects items following a precomputed order of items regardless of the observations made while packing. Thus, a universal policy is identified with a sequence of the given items. This is in contrast to general adaptive policies, where the next item to try can vary with the observations made up to that point. We present an algorithm that produces a randomized universal policy that achieves a constant robustness ratio.

If cancellation is not allowed, then there is no difference between adaptive and universal policies because the selection terminates once a selected item does not fit the knapsack. In this case, we observe that no randomized adaptive policy achieves a constant robustness ratio. Due to this hardness, we consider a *stochastic* knapsack capacity when cancellation is not allowed. In this situation, we assume that the knapsack capacity is determined according to some probability distribution and that information about the distribution is available. Based on this assumption, we compute a sequence of items as a solution. When the knapsack capacity is realized, the items in the prefix of the sequence are selected so that their total size does not exceed the realized capacity. The objective of the problem is to maximize the expected value of the submodular function f for the selected items. We address this problem as the *submodular maximization problem with a stochastic knapsack capacity* (SMPSC). We say that the *approximation ratio* of a sequence is α (≤ 1) if its expected objective value is at least α times the maximum expected value of f for any instance. The sequence computed in an α-robust policy for SMPUC achieves an α-approximation ratio for SMPSC. However, the opposite does not hold, and an algorithm of a constant approximation ratio may exist for SMPSC even though no randomized adaptive policy achieves a constant robustness ratio for SMPUC. Indeed, we present such an algorithm.

1.1. Related studies. There are a huge number of studies on the submodular maximization problems (e.g., [23]), but we are aware of no previous work on SMPUC or SMPSC. Regarding studies on the stochastic setting of the problem, several papers proposed concepts of submodularity for random set functions and discussed adaptive policies to maximize those functions [2, 13]. There are also studies on the submodular maximization over an item set in which each item is activated stochastically [1, 3, 12, 15].

When the objective function is modular (i.e., the function returns the sum of the values associated with the selected items), the submodular maximization problem with a knapsack constraint is equivalent to the classic knapsack problem. We call the special case of SMPSC with modular objective functions *the knapsack problem with a stochastic capacity* (KPSC). For the classic knapsack problem, there are numerous studies on the problem with stochastic sizes and rewards of items [7, 14, 20], which is called the stochastic knapsack problem. Note that the stochastic knapsack problem is different from KPSC, and there is no direct relationship between them. However, most of the algorithms for the stochastic knapsack problem can be applied to KPSC with slight changes. Indeed, one of our algorithms for SMPSC is based on the idea of Gupta et al. [14] for the stochastic knapsack problem.

The covering version of KPSC is represented as a single machine scheduling problem with nonuniform processing speed. This machine scheduling problem is known to be strongly NP-hard [16], which implies that pseudopolynomial time algorithms are unlikely to exist. This is in contrast to the fact that the classic knapsack problem and its covering version admit pseudopolynomial time algorithms. Megow and Verschae [22] gave a polynomial time approximation scheme (PTAS) for the machine scheduling problem. Moreover, Epstein et al. [9] showed that a deterministic sequence of jobs achieves $(4 + \epsilon)$-approximation for any processing speed function simultaneously, and the approximation ratio can be improved to $e+\epsilon$ by considering randomized sequences.

To the best of our knowledge, KPSC itself has not been well studied. The only previous study we are aware of is the thesis of Dabney [6], wherein a PTAS is presented for the problem. Since the knapsack problem and its covering version are equivalent in the existence of exact algorithms, the strongly NP-hardness of the covering version implies the same hardness for KPSC.

Regarding the *knapsack problem with an unknown capacity* (KPUC), Megow and Mestre [21] mentioned that no deterministic policy achieves a constant robustness ratio when cancellation is not allowed. They presented an algorithm that constructs for each instance a policy whose robustness ratio is arbitrarily close to that of an optimal policy that achieves the largest robustness ratio for the instance. When cancellation is allowed, Disser et al. [8] provided a deterministic 1/2-robust universal policy for KPUC. They also proved that no deterministic adaptive policy achieves a robustness ratio better than 1/2, which means that the robustness ratio of their deterministic universal policy is best possible even for any deterministic adaptive policy.

Several previous studies [17, 18] discuss another robustness of the knapsack problem. In that setting, both a knapsack constraint and a cardinality constraint are given, and the robustness is defined for the cardinality constraint. Bernstein, Disser, and Gross [4] considered incremental algorithms for a general cardinality constrained maximization problem, where each item is associated with a unit size and the objective function is monotone subadditive. Since a submodular function is subadditive, their problem includes SMPUC with uniform item sizes. For this special case, their

TABLE 1
Summary of main results in this paper. Numbers represent robustness ratios for the unknown capacity case and approximation ratios for the stochastic capacity case.

	With cancellation	Without cancellation
Unknown capacity	Rand. adaptive $\frac{1-1/e}{2}$ (> 0.316) Rand. universal $\frac{1-1/\sqrt[4]{e}}{2}$ (> 0.110) Det. adaptive $\frac{2(1-1/e)}{21}$ (> 0.060)	No constant robustness ratio
Stochastic capacity		Rand. pseudopoly $\frac{1}{4} - o(1)$ Rand. $\frac{1-1/\sqrt[4]{e}}{4} - \epsilon$ ($> 0.055 - \epsilon$)

incremental algorithm gives a universal policy of robustness ratio $1/(1+\phi)$ (< 0.382), where ϕ is the golden ratio.

We note that some of these previous studies use different terminologies from ours. For example, the robustness ratio is called *competitive ratio* in [4]. A universal policy is called a *nonadaptive* policy in other papers, e.g., [3, 15].

1.2. Contributions. Contributions in this paper are summarized in Table 1. For the case where cancellation is allowed, we present three polynomial time algorithms for SMPUC. These algorithms produce

- a randomized adaptive policy of robustness ratio $(1 - 1/e)/2$ (section 3.1),
- a deterministic adaptive policy of robustness ratio $2(1-1/e)/21$ (section 3.2), and
- a randomized universal policy of robustness ratio $(1 - 1/\sqrt[4]{e})/2$ (section 3.3).

Our algorithms are constructed based on a simple greedy algorithm [19] for the monotone submodular maximization problem with a knapsack constraint. The greedy algorithm outputs a better solution between two candidates, one constructed greedily based on the increase in the objective function value per unit of size, and the other based on the increase in the objective function value. In our randomized adaptive policy, we achieve the robustness ratio $(1 - 1/e)/2$ by guaranteeing that each of the two candidate solutions is output by our policy with probability $1/2$.

We convert this randomized policy into a deterministic one by mixing the two strategies that correspond to the two candidate solutions. We remark that the same approach is taken for KPUC to construct a deterministic $1/2$-robust universal policy by Disser et al. [8]. They call an item a *swap item* if it forms a single-item solution better than a greedy solution for some knapsack capacity. A key idea in their policy is to pack swap items earlier than the others. However, their technique fully relies on the property that the objective function is modular, and their choice of swap items is not suitable for SMPUC. In the present paper, we introduce a new notion of *single-valuable items*. This enables us to design a deterministic $2(1 - 1/e)/21$-robust policy. We remark that our proof technique is also different from the standard one used in related work. Moreover, we modify the randomized adaptive policy to obtain the randomized universal policy. A key idea here is to guess a capacity by a doubling strategy.

We also show that no randomized adaptive policy achieves a robustness ratio better than $(25+15\cdot 2^{1/3}+9\cdot 2^{2/3})/71$ (< 0.820) for KPUC (section 4.1). It is known that the robustness ratio achieved by deterministic policies for this problem is at most $1/2$ [8], but there was no upper bound on the robustness ratio for randomized policies. Disser et al. [8] mentioned that it is an interesting open problem to improve the ratio $1/2$ by randomized policies. Our upper bound implies that, even if randomized policies are considered, the ratio cannot be improved better than 0.820. In addition, we show that no deterministic policy achieves a robustness ratio better than $(1 + \sqrt{5})/4$, and

no randomized policy achieves a robustness ratio better than $(5+\sqrt{5})/8$ for SMPUC even when each item has a unit size (section 4.3).

When cancellation is not allowed, it is already known that KPUC admits no deterministic universal policy of a constant robustness ratio [21]. We advance this hardness result by showing that no randomized adaptive policy achieves a constant robustness ratio (section 4.2).

For SMPSC without cancellation, we present the following two algorithms:

- a pseudopolynomial time randomized algorithm of approximation ratio $1/4-o(1)$ (section 5.3), and
- a polynomial time randomized algorithm of approximation ratio $(1-1/\sqrt[4]{e})/4-\epsilon$ for any small constant $\epsilon > 0$ (section 5.4).

The former algorithm is based on the observation that the problem can be reduced to the submodular maximization problem with an interval independent constraint. The latter algorithm is based on the idea of Gupta et al. [14] for the stochastic knapsack problem. Gupta et al. regarded the knapsack capacity as a time limit and showed that a rounding algorithm for a time-index linear program (LP) gives an adaptive policy for the stochastic knapsack problem. Although the formulation size of the time-index LP is not polynomial, a simple doubling technique reduces the formulation size to polynomial with a loss of the approximation ratio. In our algorithm for SMPSC, we first introduce a time-index convex relaxation of the problem using the multilinear extension of the objective function, and we show that the rounding algorithm of Gupta et al. is a monotone contention resolution scheme for any realization of the knapsack capacity. This observation gives a pseudopolynomial time algorithm of approximation ratio $(1-1/\sqrt[4]{e})/2-o(1)$ for SMPSC. We then transform it into a polynomial time algorithm. This transformation requires a careful sketching of knapsack capacity, which was not necessary for the stochastic knapsack problem.

1.3. Organization. The rest of this paper is organized as follows. Section 2 gives a formal definition of SMPUC. Section 3 presents the adaptive policies for SMPUC with cancellation. Section 4 provides the upper bounds on the robustness ratios. Section 5 presents a formal definition of SMPSC without cancellation and the approximation algorithms for it. Section 6 concludes the paper.

2. Setting of SMPUC. In this section, we present a formal definition of problem SMPUC and related terminologies used in this paper.

2.1. Maximization of monotone submodular functions. The inputs of the problem are a set I of n items and a nonnegative set function $f : 2^I \to \mathbb{R}_+$. In this paper, we assume that (i) f satisfies $f(\emptyset) = 0$, (ii) f is *submodular* (i.e., $f(X)+f(Y) \geq f(X \cup Y) + f(X \cap Y)$ for any $X, Y \subseteq I$), and (iii) f is *monotone* (i.e., $f(X) \leq f(Y)$ for any $X, Y \subseteq I$ with $X \subseteq Y$). The function f is given as an oracle that returns the value of $f(X)$ for any query $X \subseteq I$. Let $\mathcal{I} \subseteq 2^I$ be any family such that $X \subseteq Y \in \mathcal{I}$ implies $X \in \mathcal{I}$. The $\mathcal{I}$-constrained submodular maximization problem seeks $X \in \mathcal{I}$ that maximizes $f(X)$.

We focus on the case where $\mathcal{I}$ corresponds to a knapsack constraint. Namely, each item $i \in I$ is associated with a size $s(i)$, and $\mathcal{I}$ is defined as $\{X \subseteq I\colon \sum_{i \in X} s(i) \leq C\}$ for some knapsack capacity $C > 0$. We denote $\sum_{i \in X} s(i)$ by $s(X)$ for any $X \subseteq I$.

2.2. Problem SMPUC. In SMPUC, the knapsack capacity C is unknown. We can see whether an item set fits the knapsack only when an item is added to the item set.

A solution for SMPUC is an adaptive policy $\mathcal{P}$, which is represented as a binary

decision tree that contains every item at most once along each path from the root to a leaf. Each node of the decision tree is an item to try packing into the knapsack. A *randomized* policy is a probability distribution over binary decision trees. One of the decision trees is selected according to the probability distribution. For a fixed capacity C, the output of a policy $\mathcal{P}$ is an item set denoted by $\mathcal{P}(C) \subseteq I$ obtained as follows. We start with $\mathcal{P}(C) = \emptyset$ and check whether the item r at the root of $\mathcal{P}$ fits the knapsack, i.e., whether $s(r) + s(\mathcal{P}(C)) \leq C$. If the item fits, then we add r to $\mathcal{P}(C)$ and continue packing recursively with the left subtree of r. Otherwise, we have two options: when cancellation is allowed, we discard r and continue packing recursively with the right subtree of r; when cancellation is not allowed, we discard r and output $\mathcal{P}(C)$ to terminate the process.

When a policy does not depend on the observation made while packing, we call such a policy *universal*. Since every path from the root to a leaf in a universal policy is identical, we can identify a universal policy with a sequence $\Pi = (\Pi_1, \ldots, \Pi_n)$ of items in I. For a fixed capacity C, the output of a universal policy, denoted by $\Pi(C)$, is constructed as follows. We start with $\Pi(C) = \emptyset$ and add items to $\Pi(C)$ in n iterations. In the ith iteration, we check whether $s(\Pi(C)) + s(\Pi_i) \leq C$ holds. If true, then Π_i is added to $\Pi(C)$. Otherwise, Π_i is discarded, and we proceed to the next iteration when cancellation is allowed, and we terminate the process when cancellation is not allowed.

3. Adaptive policies for SMPUC with cancellation.

3.1. Randomized $(1 - 1/e)/2$-robust policy. In this subsection, we present a randomized adaptive policy for SMPUC in the situation that cancellation is allowed. The idea of our algorithm is based on a simple greedy algorithm [19] for the submodular maximization problem with a knapsack constraint. The greedy algorithm generates two candidate item sets. One set is obtained greedily by repeatedly inserting an item maximizing the increase in the objective function value per unit of size. The other is obtained similarly by packing an item maximizing the increase in the objective function value. Then, the algorithm returns the set with the larger value, which leads to a $(1 - 1/e)/2$-approximation solution.

The idea of choosing a better solution is not suitable for SMPUC, in which we cannot remove items from the knapsack. We resolve this issue by generating at random one of two policies $\mathcal{P}^1$ and $\mathcal{P}^2$ that are analogous to the above two greedy methods. Policy $\mathcal{P}^1$ corresponds to the greedy algorithm based on the increase in the objective function value per unit of size. We formally present this policy as Algorithm 1. The item i_j corresponds to a node of depth $j-1$ in $\mathcal{P}^1$. We remark that Algorithm 1 chooses i_j independently of the knapsack capacity, but the choice depends on which items fit the knapsack and which did not so far. For generality of the algorithm, we assume that the algorithm receives an initial state U of the knapsack, which is defined as a subset of I (this will be used in section 3.2).

The policy $\mathcal{P}^2$ tries to pack items greedily based on the increase in the objective function value. Our algorithm is summarized in Algorithm 2. We remark that Algorithm 2 chooses the item i_j for each iteration j in polynomial time with respect to the cardinality of I.

We analyze the robustness ratio of Algorithm 2. In the execution of Algorithm 1 for (I, U) under some capacity C, we call the order $(i_1, \ldots, i_{|I \setminus U|})$ of items in $I \setminus U$ the *greedy order* for (I, U) with capacity C, where i_j is the jth selected item at line 3. We remark that the greedy order depends on C. For example, suppose that $I = \{a, b, c\}$, $s(a) = 2$, $s(b) = 1$, $s(c) = 1$, $J_a = \{1, 2, 3\}$, $J_b = \{1, 2\}$, $J_c = \{4\}$, and

Algorithm 1: Greedy algorithm $\mathcal{P}^1$ for (I, U).

$X \leftarrow U$, $R \leftarrow I \setminus U$;
foreach $j = 1, \ldots, |I \setminus U|$ **do**
  let $i_j \in \arg\max \{(f(X \cup \{i\}) - f(X))/s(i) \colon i \in R\}$;
  if *i_j fits the knapsack (i.e., $s(X) + s(i_j) \le C$)* **then** `// left subtree`
    $X \leftarrow X \cup \{i_j\}$;
  else `// right subtree`
    discard i_j;
  $R \leftarrow R \setminus \{i_j\}$;
return X;

Algorithm 2: Randomized $(1 - 1/e)/2$-robust adaptive policy.

flip a coin;
if *heads* **then** execute Algorithm 1 for $(I, \emptyset)$; `// policy` $\mathcal{P}^1$
else `// policy` $\mathcal{P}^2$
  $X \leftarrow \emptyset$, $R \leftarrow I$;
  foreach $j = 1, \ldots, |I|$ **do**
    let $i_j \in \arg\max \{f(X \cup \{i\}) - f(X) \colon i \in R\}$;
    if *i_j fits the knapsack (i.e., $s(X) + s(i_j) \le C$)* **then** `// left subtree`
      $X \leftarrow X \cup \{i_j\}$;
    else `// right subtree`
      discard i_j;
    $R \leftarrow R \setminus \{i_j\}$;
  return X;

$f(X) = |\bigcup_{e \in X} J_e|$, which is submodular because it is a coverage function. If $C \ge 2$, then Algorithm 1 for $(I, \emptyset)$ tries to insert items in the order (a, c, b), while the order is (a, b, c) if $C < 2$. A key concept in the analysis of Algorithm 2 is to focus on the first item in the greedy order that is a member of OPT_C but is spilled from $\mathcal{P}^1(C)$. The following lemma is useful in the analysis of Algorithm 2 and also algorithms given in subsequent sections.

LEMMA 3.1. *Let C, C' be any positive numbers. When Algorithm* 1 *is executed for (I, U) with capacity C, we have, for any index $j \in \{1, \ldots, |I \setminus U|\}$, that*

$$f\Big(\big((\mathcal{P}^1(C) \cup \mathrm{OPT}_{C'}) \cap \{i_1, \ldots, i_j\}\big) \cup U\Big) \ge \left(1 - \exp\left(-\frac{s((\mathcal{P}^1(C) \cup \mathrm{OPT}_{C'}) \cap \{i_1, \ldots, i_j\})}{C'}\right)\right) \cdot f(\mathrm{OPT}_{C'}).$$

Moreover, we have $(\mathcal{P}^1(C) \cup \mathrm{OPT}_{C'}) \cap \{i_1, \ldots, i_j\} = \mathcal{P}^1(C) \cap \{i_1, \ldots, i_j\}$ for any $j < q$ where q is the smallest index such that $i_q \in \mathrm{OPT}_{C'}$ and $i_q \notin \mathcal{P}^1(C)$ ($q = \infty$ if there is no such index).

To prove Lemma 3.1, we show the following two lemmas.

LEMMA 3.2. *For any $X \subseteq Y \subseteq I$, we have*

$$f(Y) \leq f(X) + \sum_{i \in Y \setminus X} (f(X \cup \{i\}) - f(X)).$$

Proof. Suppose that $Y \setminus X = \{a_1, \ldots, a_l\}$, where $l = |Y \setminus X|$. Let $X_j = X \cup \{a_1, \ldots, a_j\}$ for each $j = 1, \ldots, l$. Note that $X_0 = X$ and $X_l = Y$. Since f is submodular, we have $f(X_j) - f(X_{j-1}) \leq f(X \cup \{a_j\}) - f(X)$. By summing both sides over j, we obtain $f(Y) - f(X) = \sum_{j=1}^{l} (f(X_j) - f(X_{j-1})) \leq \sum_{j=1}^{l} (f(X \cup \{a_j\}) - f(X))$, which proves the lemma. ∎

LEMMA 3.3. *Let C be any positive number. For any $X \subseteq Y \subseteq I$ and any $i \in I \setminus Y$ such that $\mathrm{OPT}_C \setminus X \subseteq I \setminus Y$ and*

$$\frac{f(X \cup \{i\}) - f(X)}{s(i)} = \max \left\{ \frac{f(X \cup \{v\}) - f(X)}{s(v)} \colon v \in I \setminus Y \right\},$$

it holds that

$$f(X \cup \{i\}) - f(X) \geq \frac{s(i)}{C} (f(\mathrm{OPT}_C) - f(X)).$$

Proof. By applying Lemma 3.2 with $Y = \mathrm{OPT}_C \cup X$ ($\supseteq X$), we have $f(\mathrm{OPT}_C \cup X) \leq f(X) + \sum_{v \in \mathrm{OPT}_C \setminus X} (f(X \cup \{v\}) - f(X))$. In addition, $f(\mathrm{OPT}_C) \leq f(\mathrm{OPT}_C \cup X)$ holds by monotonicity. Thus, we have

$$\begin{aligned} f(\mathrm{OPT}_C) - f(X) &\leq \sum_{v \in \mathrm{OPT}_C \setminus X} (f(X \cup \{v\}) - f(X)) \\ &= \sum_{v \in \mathrm{OPT}_C \setminus X} s(v) \cdot \frac{f(X \cup \{v\}) - f(X)}{s(v)} \\ &\leq \left(\sum_{v \in \mathrm{OPT}_C \setminus X} s(v) \right) \cdot \frac{f(X \cup \{i\}) - f(X)}{s(i)} \\ &\leq C \cdot \frac{f(X \cup \{i\}) - f(X)}{s(i)}. \end{aligned}$$

∎

Proof of Lemma 3.1. We denote $Y_j = U \cup \{i_1, \ldots, i_j\}$ and $Z_j = (\mathcal{P}^1(C) \cup \mathrm{OPT}_{C'}) \cap \{i_1, \ldots, i_j\}$ for each j. Let $Y_0 = U$ and $Z_0 = \emptyset$. We first prove by induction on j that $f(Z_j \cup U) \geq (1 - \exp(-s(Z_j)/C')) \cdot f(\mathrm{OPT}_{C'})$ for any j. For $j = 0$, the statement clearly holds because $1 - \exp(-s(Z_0)/C') = 0$ and $f(Z_0 \cup U) \geq f(\emptyset) = 0$.

Suppose that $f(Z_{j-1} \cup U) \geq (1 - \exp(-s(Z_{j-1})/C')) \cdot f(\mathrm{OPT}_{C'})$ holds for some $j \geq 0$. If $i_j \notin \mathcal{P}^1(C) \cup \mathrm{OPT}_{C'}$, then we have $Z_j = Z_{j-1}$, and hence it is easy to see that $f(Z_j \cup U) \geq (1 - \exp(-s(Z_j)/C')) \cdot f(\mathrm{OPT}_{C'})$. In the following, we may assume that $i_j \in \mathcal{P}^1(C) \cup \mathrm{OPT}_{C'}$. Note that $Z_{j-1} \cup \{i_j\} = Z_j$. We observe that by the choice of i_j in Algorithm 1, it holds that

$$\frac{f(Z_{j-1} \cup U \cup \{i_j\}) - f(Z_{j-1} \cup U)}{s(i_j)} = \max_{v \in I \setminus Y_{j-1}} \frac{f(Z_{j-1} \cup U \cup \{v\}) - f(Z_{j-1} \cup U)}{s(v)}.$$

Because $\mathrm{OPT}_{C'} \setminus (Z_{j-1} \cup U) \subseteq I \setminus Y_{j-1}$, we can apply Lemma 3.3 with $X = Z_{j-1} \cup U$, $Y = Y_{j-1}$, and $i = i_j$ to derive

$$f(Z_j \cup U) - f(Z_{j-1} \cup U) \geq \frac{s(i_j)}{C'} (f(\mathrm{OPT}_{C'}) - f(Z_{j-1} \cup U)).$$

Therefore, by using this inequality, we see that

$$\begin{aligned}
f(Z_j \cup U) &\geq f(Z_{j-1} \cup U) + \frac{s(i_j)}{C'}(f(\mathrm{OPT}_{C'}) - f(Z_{j-1} \cup U)) \\
&= \left(1 - \frac{s(i_j)}{C'}\right) f(Z_{j-1} \cup U) + \frac{s(i_j)}{C'} f(\mathrm{OPT}_{C'}) \\
&\geq \left(1 - \frac{s(i_j)}{C'}\right) \cdot \left(1 - \exp\left(-\frac{s(Z_{j-1})}{C'}\right)\right) \cdot f(\mathrm{OPT}_{C'}) + \frac{s(i_j)}{C'} f(\mathrm{OPT}_{C'}) \\
&= \left(1 - \left(1 - \frac{s(i_j)}{C'}\right) \cdot \exp\left(-\frac{s(Z_{j-1})}{C'}\right)\right) \cdot f(\mathrm{OPT}_{C'}) \\
&\geq \left(1 - \exp\left(-\frac{s(i_j)}{C'}\right) \cdot \exp\left(-\frac{s(Z_{j-1})}{C'}\right)\right) \cdot f(\mathrm{OPT}_{C'}) \\
&= \left(1 - \exp\left(-\frac{s(Z_j)}{C'}\right)\right) \cdot f(\mathrm{OPT}_{C'}),
\end{aligned}$$

where the last inequality holds because $1 - x \leq \exp(-x)$ for any x.

It remains to show that $Z_j = \mathcal{P}^1(C) \cap \{i_1, \ldots, i_j\}$ for any $j < q$. The inclusion $\supseteq$ is clear. For any $j \leq q$, we have $\mathrm{OPT}_{C'} \cap \{i_1, \ldots, i_j\} \subseteq \mathcal{P}^1(C) \cap \{i_1, \ldots, i_j\}$ by the choice of q. Thus, $Z_j = (\mathrm{OPT}_{C'} \cap \{i_1, \ldots, i_j\}) \cup (\mathcal{P}^1(C) \cap \{i_1, \ldots, i_j\}) \subseteq \mathcal{P}^1(C) \cap \{i_1, \ldots, i_j\}$. This completes the proof. ∎

We are ready to analyze the robustness ratio of Algorithm 2. For any $C \in \mathbb{R}_+$, we denote $I_C = \{i \colon s(i) \leq C\}$ and denote by i^C an item with the largest value in this set, i.e., $i^C \in \arg\max\{f(\{i\}) \colon i \in I_C\}$.

THEOREM 3.4. *Algorithm 2 is a randomized* $(1 - 1/e)/2 > 0.316$*-robust adaptive policy.*

Proof. Let $\mathcal{P}^1$ (respectively, $\mathcal{P}^2$) be the adaptive policy in Algorithm 2 when the coin comes up heads (respectively, tails). Suppose that the given capacity is C. The expected value of the output by Algorithm 2 is $f(\mathrm{ALG}_C) = (f(\mathcal{P}^1(C)) + f(\mathcal{P}^2(C)))/2$. If $\mathrm{OPT}_C \subseteq \mathcal{P}^1(C)$, then we get $f(\mathrm{ALG}_C) \geq f(\mathcal{P}^1(C))/2 \geq f(\mathrm{OPT}_C)/2$. Assume that $\mathrm{OPT}_C \not\subseteq \mathcal{P}^1(C)$. Let $(i_1, i_2, \ldots, i_n)$ be the greedy order for $(I, \emptyset)$ with capacity C. Let q be the smallest index such that $i_q \in \mathrm{OPT}_C$ and $i_q \notin \mathcal{P}^1(C)$. We have $f(\mathcal{P}^2(C)) \geq f(\{i^C\}) \geq f(\{i_q\})$ because $i^C \in \mathcal{P}^2(C)$ and $i_q \in I_C$. By the monotonicity and submodularity of f, we have

$$f(\mathrm{ALG}_C) \geq \frac{f(\mathcal{P}^1(C) \cap \{i_1, \ldots, i_{q-1}\}) + f(\{i_q\})}{2} \geq \frac{f((\mathcal{P}^1(C) \cap \{i_1, \ldots, i_{q-1}\}) \cup \{i_q\})}{2}.$$

Note that by the choice of q, we have $(\mathcal{P}^1(C) \cap \{i_1, \ldots, i_{q-1}\}) \cup \{i_q\} = (\mathcal{P}^1(C) \cup \mathrm{OPT}_C) \cap \{i_1, \ldots, i_q\}$ and $s((\mathcal{P}^1(C) \cap \{i_1, \ldots, i_{q-1}\}) \cup \{i_q\}) > C$. Thus, Lemma 3.1 implies that $f((\mathcal{P}^1(C) \cup \mathrm{OPT}_C) \cap \{i_1, \ldots, i_q\}) \geq (1 - 1/e) f(\mathrm{OPT}_C)$, and hence we have $f(\mathrm{ALG}_C) \geq (1 - 1/e) f(\mathrm{OPT}_C)/2$. ∎

3.2. Deterministic $2(1 - 1/e)/21$-robust policy. In this subsection, we present a deterministic adaptive policy for SMPUC by modifying Algorithm 2. To this end, let us review the result of Disser et al. [8] for KPUC, which is identical to SMPUC with modular objective functions. They obtained a deterministic 1/2-robust universal policy for KPUC based on the greedy order for $(I, \emptyset)$ with the given capacity. Their policy first tries to insert items with large values, which are called *swap items*. For a greedy order $(i_1, \ldots, i_n)$, an item i_j is called a swap item if $f(\{i_j\}) \geq f(\mathcal{P}^1(C))$

for some capacity C such that i_j is the first item that overflows the knapsack when the items are packed in the greedy order. The key property is that the greedy order does not depend on the capacity C when f is modular. This enables one to determine swap items from the unique greedy order and to obtain a deterministic universal policy.

On the other hand, it is hard to apply the idea of [8] for our purpose. The difficulty is that the greedy order depends on the capacity when f is submodular. Thus, the notion of swap items is not suitable in SMPUC for choosing the items that should be tried first. In this paper, we introduce *single-valuable items*, which are items i satisfying $f(\{i\}) \geq 2 \cdot f(\mathrm{OPT}_{s(i)/2})$ $(= 2 \cdot \max\{f(X)\colon s(X) \leq s(i)/2\})$. Since computing $f(\mathrm{OPT}_{s(i)/2})$ is NP-hard, we use a γ-approximation algorithm for it; for example, γ can be set to $1 - 1/e$ if we use the polynomial time algorithm of Sviridenko [23]. We guess the set S of single-valuable items with the aid of the γ-approximation algorithm. Thus, S satisfies $f(\{i\}) \geq 2\gamma \cdot f(\mathrm{OPT}_{s(i)/2})$ for any item $i \in S$ and $f(\{i\}) \leq 2 \cdot f(\mathrm{OPT}_{s(i)/2})$ for any item $i \notin S$.

Our algorithm first inserts the most valuable item in S that fits the knapsack (if it exists) and then it executes Algorithm 1 with the remaining items. We remark that, unlike the algorithm for KPUC [8], our algorithm executes Algorithm 1 once a single-valuable item fits the knapsack. We summarize our algorithm in Algorithm 3.

Algorithm 3: Deterministic $2\gamma/21$-robust policy.

```
1  U ← ∅, R ← I, S ← ∅;
2  foreach i ∈ I do
3  |  Let L be a γ-approximate solution to max{f(X): s(X) ≤ s(i)/2, X ⊆ I};
4  |  if f({i}) ≥ 2f(L) then S ← S ∪ {i};
5  while U = ∅ and S ∩ R ≠ ∅ do
6  |  let i ∈ arg max{f({i}): i ∈ S ∩ R};
7  |  if i fits the knapsack (i.e., s(i) ≤ C) then U ← {i}; // left subtree
8  |  else discard i; // right subtree
9  |  R ← R \ {i};
10 execute Algorithm 1 for (R ∪ U, U);
```

Note that our algorithm constructs S and decides which item to try in polynomial time. In the rest of this subsection, we let R, U, and S denote the item sets maintained when Algorithm 1 is invoked in Algorithm 3. Then U is empty if $S \cap I_C = \emptyset$, and U consists of exactly one item $i^* \in \arg\max\{f(\{i\})\colon i \in I_C \cap S\}$ otherwise. The following theorem is the main result of this subsection.

THEOREM 3.5. *Algorithm* 3 *using a γ-approximation algorithm in line* 3 *is a $\min\{2\gamma/21, (1 - 1/\sqrt[3]{e})/3\}$-robust deterministic adaptive policy. In particular, it is $2(1 - 1/e)/21 > 0.0602$-robust when $\gamma = 1 - 1/e$, and it is $(1 - 1/\sqrt[3]{e})/3 > 0.0944$-robust when $\gamma = 1$.*

We remark that Theorem 3.5 implies that there always exists a deterministic adaptive policy of robustness ratio at least $(1-1/\sqrt[3]{e})/3$ even though we do not know how to find it in polynomial time.

We describe the proof idea. We discuss the behavior of Algorithm 3 when the given capacity is C. Since the output of Algorithm 3 is $\mathcal{P}^1(C)$ for $(R \cup U, U)$, one can think of a proof similar to that of Theorem 3.4. However, we may not be able to use the value $f(\{i_q\})$ in the evaluation of $f(\mathcal{P}^1(C))$ here because $\mathcal{P}^1(C)$ may not contain

any items that bound $f(\{i_q\})$. We show the theorem using a different approach. A basic idea is to divide OPT_C into several subsets $A_1, \ldots, A_k$ and derive a bound $f(\mathrm{OPT}_C) \leq f(A_1) + \cdots + f(A_k)$ by the submodularity of f. A key idea is to evaluate each $f(A_i)$ using properties of single-valuable items, which are shown as the following two lemmas.

LEMMA 3.6. *It holds that* $f(\{i\}) \leq \max\{f(U), 2f(\mathrm{OPT}_{s(i)/2})\}$ *for any item* $i \in I_C$.

Proof. The lemma follows because $f(\{i\}) \leq f(U)$ if $i \in S$ and $f(\{i\}) \leq 2f(\mathrm{OPT}_{s(i)/2})$ if $i \notin S$. ∎

LEMMA 3.7. *Let* $s^* = s(U)$. *If* $s^* \leq C/2$, *then, for any number* $x \in [s^*, C/2]$, *it holds that* $f(\mathrm{OPT}_{2x}) \leq 3f(\mathrm{OPT}_x)$.

Proof. First, suppose that OPT_{2x} contains some item i with $s(i) > x$ and $i \in S$. Then we have $f(\mathrm{OPT}_{2x}) \leq f(\{i\}) + f(\mathrm{OPT}_{2x} \setminus \{i\})$ by the submodularity of f. Since the existence of item i implies $I_C \cap S \neq \emptyset$, U consists of exactly one item from S. We denote this item by i^*. Any item $i' \in S$ with $f(\{i'\}) > f(\{i^*\})$ does not fit the knapsack with capacity C ($\geq 2x$). This implies that $f(\{i\}) \leq f(\{i^*\})$. Moreover, we have $f(\{i^*\}) \leq f(\mathrm{OPT}_x)$ because $s(i^*) = s^* \leq x$. Hence, $f(\{i\}) \leq f(\mathrm{OPT}_x)$ follows. On the other hand, $f(\mathrm{OPT}_{2x} \setminus \{i\}) \leq f(\mathrm{OPT}_{2x-s(i)}) \leq f(\mathrm{OPT}_x)$ holds, where the last inequality follows from $s(i) > x$. Therefore, we see that

$$f(\mathrm{OPT}_{2x}) \leq f(\{i\}) + f(\mathrm{OPT}_{2x} \setminus \{i\}) \leq 2f(\mathrm{OPT}_x).$$

Second, suppose that OPT_{2x} contains some item i with $s(i) > x$ and $i \notin S$. Recall that $i \notin S$ implies $f(\{i\}) \leq 2f(\mathrm{OPT}_{s(i)/2})$. Since $s(i)/2 \leq x < s(i)$, we see that

$$f(\mathrm{OPT}_{2x}) \leq f(\{i\}) + f(\mathrm{OPT}_{2x} \setminus \{i\}) \leq 2f(\mathrm{OPT}_{s(i)/2}) + f(\mathrm{OPT}_{2x-s(i)}) \leq 3f(\mathrm{OPT}_x).$$

Finally, assume that all items in OPT_{2x} have size at most x. We can divide OPT_{2x} into three sets A_1, A_2, A_3 with $s(A_j) \leq x$ ($j = 1, 2, 3$). Therefore, we have

$$f(\mathrm{OPT}_{2x}) \leq f(A_1) + f(A_2) + f(A_3) \leq 3f(\mathrm{OPT}_x).$$

The lemma follows from the arguments on the three cases. ∎

Proof of Theorem 3.5. Let $\mathcal{P}$ be the deterministic policy described as Algorithm 3. Suppose that the given capacity is C. We remark that $\mathcal{P}(C) = \mathcal{P}^1(C)$ for $(R \cup U, U)$. We may assume that $\mathrm{OPT}_C \not\subseteq \mathcal{P}(C)$, since otherwise $f(\mathcal{P}(C)) = f(\mathrm{OPT}_C)$. Let $s^* = s(U)$. We split the analysis into two cases: (a) $s^* < C/3$ and (b) $s^* \geq C/3$.

Case (a). We claim that

$$f(\mathcal{P}(C)) \geq (1 - 1/\sqrt[3]{e}) \cdot f(\mathrm{OPT}_C)/3.$$

Since $s^* < C/3 < C/2$, Lemma 3.7 indicates that $f(\mathrm{OPT}_{C/2}) \geq f(\mathrm{OPT}_C)/3$. We evaluate $f(\mathrm{OPT}_{C/2})$ by using Lemma 3.1 with $C' = C/2$. We may assume that $\mathrm{OPT}_{C/2} \not\subseteq \mathcal{P}(C)$; otherwise $f(\mathcal{P}(C)) \geq f(\mathrm{OPT}_{C/2})$ holds, which implies that $f(\mathcal{P}(C)) \geq f(\mathrm{OPT}_C)/3$. Let $(i_1, \ldots, i_{|R|})$ be the greedy order for $(R \cup U, U)$ with capacity C. Let q' be the smallest index such that $i_{q'} \in \mathrm{OPT}_{C/2}$ and $i_{q'} \notin \mathcal{P}^1(C)$. We denote $Z = \mathcal{P}^1(C) \cap \{i_1, \ldots, i_{q'-1}\}$. As $s(i_{q'}) \leq C/2$, we see that $s(Z) > C - s^* - s(i_{q'}) \geq C/6$. Then Lemma 3.1 implies that

$$f(\mathcal{P}(C)) \geq f(Z \cup U) \geq (1 - 1/\sqrt[3]{e}) \cdot f(\mathrm{OPT}_{C/2}),$$

and hence the claim follows.

Case (b). In this case, we prove the following two claims. Note that U is nonempty since $s^* > 0$. Let i^* denote the unique item in U.

CLAIM 3.8. $f(\mathrm{OPT}_C) \leq 7f(\mathrm{OPT}_{s^*})$.

Proof. Let $T' = \{i \in \mathrm{OPT}_C : s(i) > s^*\}$. Since $C \leq 3s^*$, we observe that T' has at most two items. We evaluate $f(\mathrm{OPT}_C)$ by splitting OPT_C depending on T'.

Suppose that $T' = \{i\}$ and $2s^* \leq s(i)$ $(\leq C)$. We note that $f(\mathrm{OPT}_C) \leq f(\{i\}) + f(\mathrm{OPT}_C \setminus \{i\})$. By Lemma 3.6, $f(\{i\}) \leq \max\{f(U), 2f(\mathrm{OPT}_{s(i)/2})\} \leq \max\{f(\mathrm{OPT}_{s^*}), 2f(\mathrm{OPT}_{s(i)/2})\}$. Since $s(i)/2 \leq C/2 < 2s^*$, it follows that $\max\{f(\mathrm{OPT}_{s^*}), 2f(\mathrm{OPT}_{s(i)/2})\} \leq 2f(\mathrm{OPT}_{2s^*})$, which is at most $6f(\mathrm{OPT}_{s^*})$ by Lemma 3.7 together with $s^* \leq s(i)/2 \leq C/2$. Moreover, we have $f(\mathrm{OPT}_C \setminus \{i\}) \leq f(\mathrm{OPT}_{C-s(i)}) \leq f(\mathrm{OPT}_{s^*})$ since $C - s(i) \leq 3s^* - 2s^* = s^*$. Thus, we obtain

$$f(\mathrm{OPT}_C) \leq f(\{i\}) + f(\mathrm{OPT}_C \setminus \{i\}) \leq 6f(\mathrm{OPT}_{s^*}) + f(\mathrm{OPT}_{s^*}) = 7f(\mathrm{OPT}_{s^*}).$$

Assume that $T' = \{i\}$ and $s(i) < 2s^*$. We observe that $f(\{i\}) \leq 2f(\mathrm{OPT}_{s^*})$ by Lemma 3.6 and $s(i)/2 < s^*$. Note that all items in $\mathrm{OPT}_C \setminus \{i\}$ have size at most s^* and their total size is at most $2s^*$. Thus, we can divide $\mathrm{OPT}_C \setminus \{i\}$ into three sets A_1, A_2, A_3 with $s(A_j) \leq s^*$ $(j = 1, 2, 3)$. Hence, we have

$$f(\mathrm{OPT}_C) \leq f(\{i\}) + f(\mathrm{OPT}_C \setminus \{i\}) \leq 2f(\mathrm{OPT}_{s^*}) + 3f(\mathrm{OPT}_{s^*}) = 5f(\mathrm{OPT}_{s^*}).$$

If $T' = \{i, i'\}$, then $s(i), s(i') < 2s^*$ and $C - s(i) - s(i') < s^*$. In this case, by Lemma 3.6 and $s(i)/2, s(i')/2 < s^*$, we have $f(\{i\}), f(\{i'\}) \leq 2f(\mathrm{OPT}_{s^*})$. This implies that

$$f(\mathrm{OPT}_C) \leq f(\{i\}) + f(\{i'\}) + f(\mathrm{OPT}_C \setminus \{i, i'\}) \leq (2+2+1)f(\mathrm{OPT}_{s^*}) = 5f(\mathrm{OPT}_{s^*}).$$

Finally, if $T' = \emptyset$, i.e., $s(i) \leq s^*$ for all $i \in \mathrm{OPT}_C$, then we can divide OPT_C into five sets $A_1, \ldots, A_5$ with $s(A_j) \leq s^*$ for all j, and hence we have $f(\mathrm{OPT}_C) \leq f(A_1) + \cdots + f(A_5) \leq 5f(\mathrm{OPT}_{s^*})$. Therefore, the claim holds. ∎

CLAIM 3.9. $f(\mathrm{OPT}_{s^*}) \leq \frac{3}{2\gamma} \cdot f(\{i^*\})$.

Proof. Recall that $f(\{i^*\}) \geq 2\gamma \cdot f(\mathrm{OPT}_{s^*/2})$. Take an arbitrary item i with the largest size in OPT_{s^*}.

Assume that $s(i) > s^*/2$ and $i \in S$. We have $f(\{i\}) \leq \max\{f(\{i'\}) : i' \in I_C \cap S\} = f(\{i^*\})$ since $i \in S$ and $s(i) \leq s^* \leq C$. Moreover, $f(\mathrm{OPT}_{s^*} \setminus \{i\}) \leq f(\mathrm{OPT}_{s^*-s(i)}) \leq f(\mathrm{OPT}_{s^*/2})$ follows from $s(i) > s^*/2$. Hence, we have $f(\mathrm{OPT}_{s^*}) \leq f(\{i\}) + f(\mathrm{OPT}_{s^*} \setminus \{i\}) \leq f(\{i^*\}) + f(\mathrm{OPT}_{s^*/2}) \leq \left(1 + \frac{1}{2\gamma}\right) f(\{i^*\})$.

Suppose that $s(i) > s^*/2$ and $i \notin S$. Then, $f(\{i\}) \leq 2f(\mathrm{OPT}_{s(i)/2}) \leq 2f(\mathrm{OPT}_{s^*/2})$ holds by $i \notin S$ and $s(i) \leq s^*$, and $f(\mathrm{OPT}_{s^*} \setminus \{i\}) \leq f(\mathrm{OPT}_{s^*/2})$ holds by $s(i) > s^*/2$. These imply that $f(\mathrm{OPT}_{s^*}) \leq f(\{i\}) + f(\mathrm{OPT}_{s^*} \setminus \{i\}) \leq 2f(\mathrm{OPT}_{s^*/2}) + f(\mathrm{OPT}_{s^*/2}) \leq \frac{3}{2\gamma} f(\{i^*\})$.

If $s(i) \leq s^*/2$, then we can divide $\mathrm{OPT}_{s^*} \setminus \{i\}$ into two sets A_1, A_2 such that $s(A_j) \leq s^*/2$ $(j = 1, 2)$, and hence it follows that $f(\mathrm{OPT}_{s^*}) \leq f(\mathrm{OPT}_{s(A_1)}) + f(\mathrm{OPT}_{s(A_2)}) + f(\mathrm{OPT}_{s(i)}) \leq 3f(\mathrm{OPT}_{s^*/2}) \leq \frac{3}{2\gamma} \cdot f(\{i^*\})$. Therefore, the claim follows since $3/(2\gamma) \geq 1 + 1/(2\gamma)$. ∎

Combining these claims gives $f(\mathcal{P}(C)) \geq (2\gamma/21) \cdot f(\mathrm{OPT}_C)$. Hence the theorem is proved. ∎

Algorithm 4: Randomized $(1-1/\sqrt[4]{e})/2$-robust universal policy.

$\Pi \leftarrow ()$;
flip a coin;
if *heads* **then**
  $l \leftarrow 1$, $s_{\min} \leftarrow \min_{i \in I} s(i)$;
  for $k \leftarrow 0$ **to** $\lceil \log_2(\sum_{i \in I} s(i)/s_{\min}) \rceil$ **do**
    let $Y^{(k)}$ be the output $\mathcal{P}^1(2^k \cdot s_{\min})$ of the policy given in Algorithm 1 for $(I, \emptyset)$;
    foreach $i \in Y^{(k)} \setminus \bigcup_{j=0}^{k-1} Y^{(j)}$ **do** $\Pi_l \leftarrow i$, $l \leftarrow l+1$;
else let Π be a nonincreasing order of items $i \in I$ in value $f(\{i\})$;
return Π;

3.3. Randomized $(1-1/\sqrt[4]{e})/2$-robust universal policy. In this subsection, we devise a randomized $(1-1/\sqrt[4]{e})/2$-robust universal policy by modifying Algorithm 2. Note that we cannot use directly Algorithm 1 in universal policies, because they do not use the observation while packing items so far. Instead we guess the capacity by the doubling strategy and emulate the execution of Algorithm 1. Our algorithm iteratively finds an item set X maximizing $f(X)$ under a bound C' on the total size $s(X)$, and then appends items of X to the sequence. The initial value of C' is set according to the input items. To compute the set X, we use Algorithm 1 in which the capacity is set to C'. We double the bound after each iteration.

Also, we cannot use the other adaptive policy $\mathcal{P}^2$ in Algorithm 2. We remark that the proof of Theorem 3.4 uses only the fact that $f(\mathcal{P}^2(C))$ contains i^C regarding the policy $\mathcal{P}^2$. This implies that in the worst case analysis, there is no difference between $\mathcal{P}^2$ and a universal policy that inserts items based on a nonincreasing order according to the values of f. Thus, we replace $\mathcal{P}^2$ with the universal policy. Our algorithm is summarized in Algorithm 4.

We remark that Algorithm 4 constructs a sequence of items in polynomial time with respect to the input size. We can prove the following result by using Lemma 3.1.

THEOREM 3.10. *Algorithm 4 is a* $(1-1/\sqrt[4]{e})/2 > 0.110$*-robust randomized universal policy.*

Proof. Let Π^1 (respectively, Π^2) be the sequence of items in I returned by Algorithm 4 when the coin comes up heads (respectively, tails). Suppose that the given capacity is C. The expected value of the output of Algorithm 4 is $f(\mathrm{ALG}_C) = (f(\Pi^1(C)) + f(\Pi^2(C)))/2$. We assume that $C \geq s_{\min}$, since otherwise $f(\mathrm{OPT}_C) = \mathrm{ALG}_C = 0$. Recall that $s_{\min} = \min_{i \in I} s(i)$. Let k be the number satisfying $2^{k-1} \cdot s_{\min} \leq C < 2^k \cdot s_{\min}$. We let $i_C \in \arg\max\{f(\{i\}) \colon s(i) \leq C\}$.

When $k = 1$ (i.e., $C < 2s_{\min}$), we have $\mathrm{OPT}_C = \{i^C\}$ because we can put only one item into the knapsack in this case. We also have $\Pi^2(C) = \{i^C\}$. Hence, it holds that $f(\mathrm{OPT}_C) = f(\Pi^2(C)) \leq 2 \cdot (f(\Pi^1(C)) + f(\Pi^2(C)))/2 = 2 \cdot f(\mathrm{ALG}_C)$.

In what follows, we assume $k \geq 2$. We observe that the total size of items in $\bigcup_{j=0}^{k-2} Y^{(j)}$ is at most $s\left(\bigcup_{j=0}^{k-2} Y^{(j)}\right) \leq \sum_{j=0}^{k-2} s(Y^{(j)}) \leq \sum_{j=0}^{k-2} 2^j \cdot s_{\min} \leq 2^{k-1} \cdot s_{\min} \leq C$. Thus, all items in $\bigcup_{j=0}^{k-2} Y^{(j)}$, in particular those in $Y^{(k-2)}$, are contained in $\Pi^1(C)$. We also observe that $f(\Pi^2(C)) \geq f(\{i^C\})$ because $i^C \in \Pi^2(C)$. Hence, it holds that $2f(\mathrm{ALG}_C) = f(\Pi^1(C)) + f(\Pi^2(C)) \geq f(Y^{(k-2)}) + f(\{i^C\})$.

We denote $C' = 2^{k-2} \cdot s_{\min}$. Recall that $Y^{(k-2)}$ is the output of the greedy algorithm for $(I, \emptyset)$ when the capacity is C'. Let $(i_1, i_2, \ldots, i_n)$ be the greedy order for $(I, \emptyset)$ with capacity C'. Let q be the smallest index such that $i_q \in \mathrm{OPT}_C$ and $i_q \notin Y^{(k-2)}$. Since this definition implies $i_q \in I_C$, we have $f(\{i^C\}) \geq f(\{i_q\})$. By the monotonicity of f, we also have $f(Y^{(k-2)}) \geq f(Y^{(k-2)} \cap \{i_1, \ldots, i_{q-1}\})$. Hence,

$$f(\mathrm{ALG}_C) \geq \frac{f(Y^{(k-2)} \cap \{i_1, \ldots, i_{q-1}\}) + f(\{i_q\})}{2} \geq \frac{f((Y^{(k-2)} \cap \{i_1, \ldots, i_{q-1}\}) \cup \{i_q\})}{2}.$$

We evaluate $f((Y^{(k-2)} \cap \{i_1, \ldots, i_{q-1}\}) \cup \{i_q\})$ using Lemma 3.1. For notational convenience, we denote $Y' = (Y^{(k-2)} \cup \mathrm{OPT}_C) \cap \{i_1, \ldots, i_q\}$. In a similar way to the proof of Theorem 3.4, we can see that $(Y^{(k-2)} \cap \{i_1, \ldots, i_{q-1}\}) \cup \{i_q\} = Y'$ by the choice of q. Thus, we have $s(Y') > C'$, and it follows that $s(Y')/C \geq C'/C \geq (2^{k-2} \cdot s_{\min})/(2^k \cdot s_{\min}) = 1/4$. Therefore, by Lemma 3.1, we have $f(\mathrm{ALG}_C) \geq f(Y')/2 \geq (1 - 1/\sqrt[4]{e})\, f(\mathrm{OPT}_C)/2$. ∎

4. Upper bounds on robustness ratios.

4.1. Randomized policies for KPUC with cancellation. In this subsection, we prove that no randomized policy achieves a robustness ratio better than $(25 + 15 \cdot 2^{1/3} + 9 \cdot 2^{2/3})/71$ (< 0.820) even if the objective function is modular when cancellation is allowed. It is known that the robustness ratio achieved by any deterministic policies for KPUC is at most 1/2 [8], but there was no upper bound on the robustness ratio for randomized policies.

Suppose that there are five items whose sizes are $1, 1, 2^{1/3}, 2^{1/3}, 2^{2/3}$, respectively, and whose weights are equal to their own sizes. Recall that the objective value is defined as the sum of the weights of selected items. Let $\mathrm{OPT}_{C'}$ be an optimal solution when the capacity is $C' \in \mathbb{R}_+$. We provide an upper bound for this instance using Yao's principle [24]. We consider an adversary that submits a probability distribution for the capacity as a mixed strategy. Let C be the random variable which represents the capacity. Then, the robustness ratio for this instance is upper-bounded by

$$\max_{\mathcal{P}} \min_{C' \in \mathbb{R}_+} \frac{f(\mathcal{P}(C'))}{f(\mathrm{OPT}_{C'})} \leq \max_{\mathcal{P}} \mathbb{E}_C \left[\frac{f(\mathcal{P}(C))}{f(\mathrm{OPT}_C)} \right],$$

where the maximum is taken over all randomized policies $\mathcal{P}$.

We assume that the adversary submits $C = 2^{2/3}$ (≈ 1.587) with probability $p_1 \coloneqq (39 - 5 \cdot 2^{1/3} - 3 \cdot 2^{2/3})/71$, $C = 2$ with probability $p_2 \coloneqq (12 - 7 \cdot 2^{1/3} + 10 \cdot 2^{2/3})/71$, and $C = 2^{4/3}$ (≈ 2.520) with probability $p_3 \coloneqq (20 + 12 \cdot 2^{1/3} - 7 \cdot 2^{2/3})/71$. Since $f(\mathrm{OPT}_{2^{2/3}}) = 2^{2/3}$, $f(\mathrm{OPT}_2) = 2$, $f(\mathrm{OPT}_{2^{4/3}}) = 2^{4/3}$, we have

$$\begin{aligned} \max_{\mathcal{P}} \mathbb{E}_C \left[\frac{f(\mathcal{P}(C))}{f(\mathrm{OPT}_C)} \right] &= \max_{\mathcal{P}} \left(p_1 \cdot \frac{f(\mathcal{P}(2^{2/3}))}{f(\mathrm{OPT}_{2^{2/3}})} + p_2 \cdot \frac{f(\mathcal{P}(2))}{f(\mathrm{OPT}_2)} + p_3 \cdot \frac{f(\mathcal{P}(2^{4/3}))}{f(\mathrm{OPT}_{2^{4/3}})} \right) \\ &= \max_{\mathcal{P}} \left(p_1 \cdot \frac{f(\mathcal{P}(2^{2/3}))}{2^{2/3}} + p_2 \cdot \frac{f(\mathcal{P}(2))}{2} + p_3 \cdot \frac{f(\mathcal{P}(2^{4/3}))}{2^{4/3}} \right). \end{aligned}$$

In what follows, we prove that this is at most $(25 + 15 \cdot 2^{1/3} + 9 \cdot 2^{2/3})/71$. Note that the maximum is attained by a deterministic policy. If a deterministic policy $\mathcal{P}$ chooses an item with size 1 first, then we have $f(\mathcal{P}(2^{2/3})) = 1$, $f(\mathcal{P}(2)) = 2$, and $f(\mathcal{P}(2^{4/3}) \leq 1 + 2^{1/3}$. Hence, the robustness ratio for the policy is at most

$$p_1 \cdot \frac{1}{2^{2/3}} + p_2 \cdot \frac{1+1}{2} + p_3 \cdot \frac{1 + 2^{1/3}}{2^{4/3}} = \frac{25 + 15 \cdot 2^{1/3} + 9 \cdot 2^{2/3}}{71}.$$

Similarly, if it selects an item with size $2^{1/3}$ first, the robustness ratio is at most

$$p_1 \cdot \frac{2^{1/3}}{2^{2/3}} + p_2 \cdot \frac{2^{1/3}}{2} + p_3 \cdot \frac{2^{1/3} + 2^{1/3}}{2^{4/3}} = \frac{25 + 15 \cdot 2^{1/3} + 9 \cdot 2^{2/3}}{71}.$$

Finally, if it selects an item with size $2^{2/3}$ first, the robustness ratio is at most

$$p_1 \cdot \frac{2^{2/3}}{2^{2/3}} + p_2 \cdot \frac{2^{2/3}}{2} + p_3 \cdot \frac{2^{2/3}}{2^{4/3}} = \frac{25 + 15 \cdot 2^{1/3} + 9 \cdot 2^{2/3}}{71}.$$

Hence, we obtain the following theorem.

THEOREM 4.1. *Even if cancellation is allowed, no randomized policy has a robustness ratio better than* $(25 + 15 \cdot 2^{1/3} + 9 \cdot 2^{2/3})/71$ (< 0.820) *for* KPUC.

4.2. Randomized policies for KPUC without cancellation. In this subsection, we prove that no randomized policy achieves a constant robustness ratio for KPUC in the setting where cancellation is not allowed.

Let M be a positive integer of at least 2. We assume that the item set I consists of n items $1, \ldots, n$, and the size and the weight of item i is M^i. The objective function f is defined by $f(S) = \sum_{i \in S} M^i$ for any $S \subseteq I$. We fix a policy and show that the robustness ratio of this policy is $O(1/M)$ for this instance if $n = \Omega(M)$.

Since there are n items, at least one item is chosen first by the policy with probability at most $1/n$. Let i denote such an item. Let us consider the case where the capacity is M^i. When the policy does not choose i first (this happens with probability at least $(n-1)/n$), the largest objective value achieved by the solution is $\sum_{i'=1}^{i-1} M^{i'} \leq 2M^{i-1}$. Therefore, the expected objective value of the policy is at most $M^i/n + (n-1)/n \cdot 2M^{i-1}$. On the other hand, the optimal solution for this instance consists of only item i, which attains the objective value M^i. The gap between these values is $O(1/M)$ if $n = \Omega(M)$.

THEOREM 4.2. *If cancellation is not allowed, then no randomized policy achieves a constant robustness ratio for* KPUC.

4.3. Deterministic policies for the unit size case of SMPUC. In this subsection, we consider a special case of SMPUC in which the size of each item is 1, i.e., the cardinality constraint case. We show that no deterministic policy (even one with no computational restrictions) achieves a robustness ratio better than $(1 + \sqrt{5})/4$ (< 0.810), and no randomized policy achieves a robustness ratio better than $(5 + \sqrt{5})/8$ (< 0.905) for this case. We remark that cancellation is not useful in the cardinality constraint case. Also, for the cardinality constraint case of SMPUC, a greedy algorithm achieves $(1 - 1/e)$-robustness (-approximation), and this is known to be the best possible among policies that run in polynomial time.

To present the upper bound on the robustness ratio, let us construct an instance of the problem. Suppose that there are three items a, b, c, each of size 1, and the objective function f is given by

$$f(\emptyset) = 0,\ f(\{a\}) = 4,\ f(\{b\}) = f(\{c\}) = 1 + \sqrt{5},$$
$$f(\{a,b\}) = f(\{a,c\}) = 3 + \sqrt{5},\ f(\{b,c\}) = f(\{a,b,c\}) = 2 + 2\sqrt{5}.$$

Note that the function is monotone submodular and symmetric in b and c. If the policy first packs a, then the robustness ratio for capacity 1 is equal to $f(\{a\})/f(\{a\}) = 1$, and that for capacity 2 is $f(\{a,b\})/f(\{b,c\}) = (3 + \sqrt{5})/(2 + 2\sqrt{5}) = (1 + \sqrt{5})/4$.

Otherwise, i.e., the policy first packs b or c, the robustness ratio for capacity 1 is $f(\{b\})/f(\{a\}) = (1+\sqrt{5})/4$, and that for capacity 2 is at least $f(\{a,b\})/f(\{b,c\}) = (1+\sqrt{5})/4$. Thus, no deterministic policy achieves a robustness ratio better than $(1+\sqrt{5})/4$.

Finally, we prove that no randomized policy achieves a robustness ratio better than $(5+\sqrt{5})/8$. Let us consider a randomized policy for the above instance that first inserts a with probability p. Then the robustness ratio for capacity 1 is

$$\frac{p \cdot f(\{a\}) + (1-p) \cdot f(\{b\})}{f(\{a\})} = \frac{4p + (1-p)(1+\sqrt{5})}{4} = \frac{(3-\sqrt{5})p + (1+\sqrt{5})}{4}.$$

Also, the robustness ratio for capacity 2 is at most

$$\frac{p \cdot f(\{a,b\}) + (1-p) \cdot f(\{b,c\})}{f(\{b,c\})} = \frac{(3+\sqrt{5})p + (2+2\sqrt{5})(1-p)}{(2+2\sqrt{5})} = \frac{4-(3-\sqrt{5})p}{4}.$$

Note that the former value is monotone increasing for p, and the latter is monotone decreasing for p. Thus, the robustness ratio of the policy is at most

$$\min\left\{\frac{(3-\sqrt{5})p + (1+\sqrt{5})}{4}, \frac{4-(3-\sqrt{5})p}{4}\right\} \leq \frac{5+\sqrt{5}}{8},$$

where the inequality holds when $p = 1/2$.

THEOREM 4.3. *For* SMPUC *with only unit-size items, no deterministic policy achieves a robustness ratio better than* $(1+\sqrt{5})/4$ (< 0.810)*, and no randomized policy achieves a robustness ratio better than* $(5+\sqrt{5})/8$ (< 0.905).

5. Approximation algorithms for SMPSC without cancellation. This section considers approximation algorithms for SMPSC without cancellation. We first present a formal definition of the problem in section 5.1 and introduce several concepts and techniques in section 5.2. We then present two approximation algorithms in sections 5.3 and 5.4.

5.1. Setting of problem SMPSC. In SMPSC, the knapsack capacity C is given according to some probability distribution. For ease of notation, we assume that the size of each item and the knapsack capacity C are nonnegative integers throughout this section. Let $T = \sum_{i \in I} s(i)$. For each $t \in [T] := \{0, 1, \ldots, T\}$, we denote by $p(t)$ the probability that the knapsack capacity is t. We assume that the probability is given to an algorithm through an oracle that returns the value of $\sum_{t'=t}^{T} p(t')$ for any query $t \in [T]$. Hence, the input size of a problem instance is $\Theta(n \log T)$, and an algorithm runs in pseudopolynomial time if its running time depends on T polynomially. A solution for SMPSC is a universal policy, i.e., a sequence $\Pi = (\Pi_1, \ldots, \Pi_n)$ of the items in I. When a capacity C is decided, the output $\Pi(C)$ of Π is constructed in the same way as universal policies for SMPUC. The objective of SMPSC is to find a sequence Π that maximizes $\mathbb{E}_C[f(\Pi(C))]$.

5.2. Multilinear extension, continuous greedy, and contention resolution scheme. From any vector $x \in [0,1]^I$, we define a random subset R_x of I so that each $i \in I$ is included in R_x with probability x_i, where the inclusion of i is independent from the inclusion of the other items. For a submodular function $f\colon 2^I \to \mathbb{R}_+$, its *multilinear extension* $F\colon [0,1]^I \to \mathbb{R}_+$ is defined by $F(x) = \mathbb{E}[f(R_x)] = \sum_{X \subseteq I} f(X) \prod_{i \in X} x_i \prod_{i' \in I \setminus X} (1 - x_{i'})$ for all $x \in [0,1]^I$. This function F

satisfies the *smooth monotone submodularity*, that is, $\partial F(x)/\partial x_i \geq 0$ for any $i \in I$ and $\partial^2 F(x)/(\partial x_i \partial x_j) \leq 0$ for any $i, j \in I$. Although it is hard to compute the value of $F(x)$ exactly, it usually can be approximated with arbitrary precision by the random sampling (see, e.g., [5]). In this paper, to make the discussion simple, we assume that F can be computed exactly.

A popular approach for solving the $\mathcal{I}$-constrained submodular maximization problem is to use a continuous relaxation of the problem. Let $P \subseteq [0,1]^I$ be a polytope in which each integer vector in P is the incidence vector of a member of $\mathcal{I}$. Then, $\max_{x \in P} F(x) \geq \max_{X \in \mathcal{I}} f(X)$ holds. In this approach, it is usually assumed that P is *downward-closed* (i.e., if $x, y \in [0,1]^I$ satisfies $y \leq x \in P$, then $y \in P$) and *solvable* (i.e., the maximization problem $\max_{x \in P} \sum_{i \in I} w_i x_i$ can be solved in polynomial time for any $w \in \mathbb{R}_+^I$).

Calinescu et al. [5] gave an algorithm called *continuous greedy* for a continuous maximization problem $\max_{x \in P} F(x)$ over a solvable downward-closed polytope P. They proved that the continuous greedy outputs a vector $x \in P$ such that $F(x) \geq (1 - 1/e - o(1)) \max_{x' \in P} F(x')$. Feldman [10] extended this analysis by observing that the continuous greedy algorithm with stopping time $b \geq 0$ outputs a vector $x \in [0,1]^I$ such that $x/b \in P$ and $F(x) \geq (1 - e^{-b} - o(1)) \max_{X \in \mathcal{I}} f(X)$. (The performance guarantee depending on the stopping time is originally given for the measured continuous greedy algorithm proposed by [11].) It is easy to see that his analysis can be modified to prove a slightly stronger result $F(x) \geq (1 - e^{-b} - o(1)) \max_{x' \in P} F(x')$. In addition, this analysis requires only the smooth monotone submodularity as a property of F.

A fractional vector $x \in P$ can be rounded into an integer vector by a contention resolution scheme. Let $b, c \in [0,1]$. For a vector x, we denote $\mathrm{supp}(x) = \{i \in I \colon x_i > 0\}$. We consider an algorithm that receives $x \in \{z \colon z/b \in P\}$ and $A \subseteq I$ as inputs and returns a random subset $\pi_x(A) \subseteq A \cap \mathrm{supp}(x)$. Such an algorithm is called a (b,c)-*balanced contention resolution scheme* if $\pi_x(A) \in \mathcal{I}$ with probability 1 for all x and A, and $\Pr[i \in \pi_x(R_x) \mid i \in R_x] \geq c$ holds for all x and $i \in \mathrm{supp}(x)$ (recall that R_x is the random subset of I determined from x). It is also called *monotone* if $\Pr[i \in \pi_x(A)] \geq \Pr[i \in \pi_x(A')]$ for any $i \in A \subseteq A' \subseteq I$. If a monotone (b,c)-balanced contention resolution scheme is available, then we can achieve the approximation ratio claimed in the following theorem by applying it to a fractional vector computed by the measured continuous greedy algorithm with stopping time b. This fact is summarized in the following theorem.

THEOREM 5.1 ([11]). *If there exists a monotone (b,c)-balanced contention resolution scheme for $\mathcal{I}$, then the $\mathcal{I}$-constrained submodular maximization problem admits an approximation algorithm of ratio $(1 - e^{-b})c - o(1)$ for any nonnegative monotone submodular function.*

5.3. Pseudopolynomial time $(1/4 - o(1))$-approximation algorithm. We present a pseudopolynomial time $(1/4 - o(1))$-approximation algorithm for SMPSC without cancellation. We reduce the problem to the following problem.

Submodular maximization problem with an interval independent constraint: We are given a set I of items. Each item $i \in I$ is associated with an interval l_i on a line. We are also given a submodular function $f \colon 2^I \to \mathbb{R}_+$. The objective is to find a subset I' of I maximizing $f(I')$ subject to the constraint that no two intervals associated with items in I' intersect, i.e., $l_i \cap l_j = \emptyset$ for all $i, j \in I'$.

Feldman [10] showed that this problem admits a $(1/4 - o(1))$-approximation randomized algorithm for monotone submodular functions.

Let us explain the reduction from SMPSC to this problem. Let I be the set

of items, and let $f\colon 2^I \to \mathbb{R}_+$ be the submodular function given in an instance of SMPSC. Recall that $T = s(I)$ and $[T'] = \{0, 1, \ldots, T'\}$. For each $i \in I$, we make $T - s(i) + 1$ copy items $i_0, \ldots, i_{T-s(i)}$, and i_j is associated with the interval $[j, j + s_i - 1]$ for each $j \in [T - s_i]$. Let I' denote the set of these items. We define a function $f'\colon 2^{I'} \to \mathbb{R}_+$ by

$$f'(U') = \sum_{t=0}^{T} p(t) f(\{i \in I\colon \exists j \in [t - s_i],\ i_j \in U'\})$$

for all $U' \subseteq I'$. It is not difficult to prove the following lemma.

LEMMA 5.2. *The function f' is monotone and submodular.*

Let $U' \subseteq I'$ be a solution for the instance of the problem with an interval independent constraint that consists of the item set I' (with associated intervals) and the submodular function f'. The definition of f' implies that there is no benefit for U' to include more than one copy of an item in I. Thus we can assume that U' includes at most one copy of each item in I. From U', we define an ordering of I as follows. If a copy of $i \in I$ is included in U', then pick item i at the time equal to the index of its copy in U', i.e., $t \in [T]$ such that $i_t \in U'$. Sort the items in nondecreasing order of the times at which they are picked. The other items follow these items, and their order is decided arbitrarily. This sequence achieves the objective value of at least $f'(U')$.

THEOREM 5.3. *Problem* SMPSC *without cancellation admits a pseudopolynomial time randomized $(1/4 - o(1))$-approximation algorithm for monotone submodular functions.*

Feldman [10] also gave a randomized $1/(2e + o(1))$-approximation algorithm for the problem with an interval independent constraint and nonmonotone submodular functions. Thus, if the submodular function is not monotone, then SMPSC admits a pseudopolynomial time randomized $1/(2e + o(1))$-approximation algorithm.

5.4. Polynomial time $((1 - 1/\sqrt[4]{e})/4 - \epsilon)$-approximation algorithm. In this subsection, we present a polynomial time algorithm of approximation ratio $(1 - 1/\sqrt[4]{e})/4 - \epsilon$ for any small constant $\epsilon > 0$. This algorithm is based on the idea of Gupta et al. [14] for the stochastic knapsack problem. We first give a pseudopolynomial time $((1 - 1/\sqrt[4]{e})/2 - \epsilon)$-approximation algorithm, and then we transform it into a polynomial time algorithm.

Our algorithm relies on a continuous relaxation of the problem. The relaxation is formulated based on the idea of using time-indexed variables. We regard the knapsack capacity as a time limit and consider that picking an item i requires processing it for $s(i)$ units of time. In the relaxation, we have a variable $x_{ti} \in [0, 1]$ for each $t \in [T - 1]$ and $i \in I$, and $x_{ti} = 1$ represents that the process of item i starts at time t and continues until time $t + s(i)$. For each $t \in [T]$ and $i \in I$, let $\bar{x}_{ti} = \sum_{t' \in [t - s(i)]} x_{t'i}$ if $t \geq s(i)$, and let $\bar{x}_{ti} = 0$ otherwise. For each $t \in [T]$, let $\bar{x}_t$ be the $|I|$-dimensional vector whose component corresponding to $i \in I$ is $\bar{x}_{ti}$. Let $F\colon [0, 1]^I \to \mathbb{R}_+$ be the multilinear extension of the submodular function f. Then, the relaxation is described

as

$$
\begin{array}{lll}
\text{maximize} & \bar{F}(x) := \sum_{t=1}^{T} p(t) F(\bar{x}_t) & \\
\text{subject to} & \sum_{t \in [T-1]} x_{ti} \le 1 & \forall i \in I, \\
& \sum_{i \in I} \sum_{t' \in [t]} x_{t'i} \min\{s(i), t\} \le 2t & \forall t = 1, \dots, T, \\
& x_{ti} \ge 0 & \forall t \in [T-1], \forall i \in I.
\end{array} \tag{5.1}
$$

Let us see that (5.1) relaxes the problem. It is not difficult to see that the first and third constants are valid. We prove that the second constraint is valid. Suppose that x is an integer solution that corresponds to a sequence of items. Let I' be the set of items picked at time t or earlier in this solution. Notice that $\sum_{i \in I} \sum_{t' \in [t]} x_{t'i} \min\{s(i), t\} = \sum_{i \in I'} \min\{s(i), t\}$ holds. Let j be the item picked last in I'. Then, since the process of all items in $I' \setminus \{j\}$ terminates by time t, we have $\sum_{i \in I' \setminus \{j\}} s(i) \le t$. Therefore, $\sum_{i \in I'} \min\{s(i), t\} = \min\{s(j), t\} + \sum_{i \in I' \setminus \{j\}} s(i) \le 2t$.

Note also that $\bar{F}$ is a smooth monotone submodular function, i.e., $\partial \bar{F}(x)/\partial x_{ti} \ge 0$ for any $t \in [T-1]$ and $i \in I$, and $\partial^2 \bar{F}(x)/(\partial x_{ti} \partial x_{t'i'}) \le 0$ for any $t, t' \in [T-1]$ and $i, i' \in I$. Hence, we can apply the continuous greedy algorithm for solving (5.1). Let x^* be an obtained feasible solution for (5.1). We first present a rounding algorithm for this solution. Since the formulation size of this relaxation is not polynomial, this part does not run in polynomial time. We convert the algorithm into polynomial time later.

Rounding algorithm. The algorithm consists of two rounds. In the first round, each item i chooses an integer t from $[T-1]$ with probability $x^*_{ti}/4$ and chooses no integer with probability $1 - \sum_{t \in [T-1]} x^*_{ti}/4$. An item is discarded if it chooses no integer. Let I_1 be the set of remaining items. For each $i \in I_1$, let t_i denote the integer chosen by i.

Then, the algorithm proceeds to the second round. For each $i \in I_1$, let J_i denote $\{j \in I_1 \colon t_j \le t_i\}$. In the second round, item i is discarded if $s(J_i) \ge t_i$. Let I_2 denote the set of items remaining after the second round. The algorithm outputs a sequence obtained by sorting the items $i \in I_2$ in nondecreasing order of t_i, where ties are broken arbitrarily, and by appending the other items after those in I_2 in an arbitrary order.

For $t \in [T]$, let $I_t = \{i \in I_2 \colon t_i \le t - s(i) - 1\}$. If $i \in I_t$, then i contributes to the objective value of the solution when the knapsack capacity is at least t.

LEMMA 5.4. *For any $t \in [T]$, I_t is the output of a monotone $(1/4, 1/2)$-balanced contention resolution scheme for the maximization problem of f under the knapsack capacity t and the fractional solution $\bar{x}^*_t$. Hence, the sequence output by the algorithm achieves an objective value of at least $\bar{F}(x^*/4)/2$ in expectation.*

Proof. Take arbitrarily $t \in [T]$. We define a random mapping $\pi \colon 2^I \to 2^I$ as follows. Let $I' \subseteq I$. We let each $i \in I'$ independently sample an integer t'_i from $[t - s(i) - 1]$ with probability $x^*_{t'_i i}/\bar{x}^*_{ti}$. Define $J'_i = \{j \in I' \colon t'_j \le t'_i\}$ for each $i \in I'$. Then, $\pi(I')$ is defined as $\{i \in I' \colon s(J'_i) < t'_i\}$.

Let us see that π is a $(1/4, 1/2)$-balanced contention resolution scheme for $\bar{x}^*_t$. For this, we analyze the probability that i is included in $\pi(R_{\bar{x}^*_t/4})$, conditioned that $i \in R_{\bar{x}^*_t/4}$. Recall that $i \in R_{\bar{x}^*_t/4}$ is not included in $\pi(R_{\bar{x}^*_t/4})$ if $s(J'_i) \ge t'_i$. Let j be an arbitrary item other than i, and let $s'(j) = \min\{s(j), t'_i\}$. Notice that $s'(J'_i) \ge t'_i$ holds if $s(J'_i) \ge t'_i$ holds. We give an upper bound on the probability that $s'(J'_i) \ge t'_i$

happens. The item $j \in I \setminus \{i\}$ is included in $R_{\bar{x}^*_t/4}$ with probability $\bar{x}^*_{tj}/4$, and then it is included in J'_i (i.e., j chooses an integer at most t'_i) with probability at most $\sum_{t' \in [t'_i]} x^*_{t'j}/\bar{x}^*_{tj}$. Hence $\mathbb{E}[s'(J'_i)] \leq \sum_{j \in I} \sum_{t' \in [t'_i]} x^*_{t'j} \min\{s(j), t'_i\}/4 \leq t'_i/2$, where the last inequality follows from the second constraint of (5.1). Applying Markov's inequality, we obtain $\Pr[s'(J'_i) \geq t'_i] \leq 1/2$. Therefore, $\Pr[i \in \pi(R_{\bar{x}^*_t/4}) \mid i \in R_{\bar{x}^*_t/4}] \geq 1/2$, which means that π is a $(1/4, 1/2)$-balanced contention resolution scheme for $\bar{x}^*_t$.

Next, we prove that π is monotone. Let $I' \subseteq I'' \subseteq I$. Then, $\sum_{j \in J'_i} s(i')$ is not smaller in the computation of $\pi(I'')$ than in the computation of $\pi(I')$, if each item in I' samples the same integer both in $\pi(I')$ and $\pi(I'')$. This implies $\Pr[i \in \pi(I')] \geq \Pr[i \in \pi(I'')]$ for each $i \in I'$. Thus, π is monotone.

Let us observe that I_t coincides with $\pi(R_{\bar{x}^*_t/4})$. Recall that, in the first round of the algorithm, each item i independently chooses an integer t_i. The probability that $t_i \leq t - s(i) - 1$ is $\sum_{t' \in [t-s(i)-1]} x^*_{ti}/4 = \bar{x}^*_{ti}/4$. Hence, item set $\{i \in I \colon t_i \leq t - s(i) - 1\}$ coincides with $R_{\bar{x}^*_t/4}$. Then, deciding whether or not an item i in this set is discarded in the second round of the algorithm is the same as computing $\pi(R_{\bar{x}^*_t/4})$. Therefore, I_t coincides with $\pi(R_{\bar{x}^*_t/4})$. ∎

By Theorem 5.1 and Lemma 5.4, the approximation ratio of our algorithm is $(1 - 1/\sqrt[4]{e})/2 - o(1)$ if it is combined with the continuous greedy algorithm with stopping time $1/4$.

Lemma 5.4 also implies that the integrality gap of (5.1) is at least $1/8$. On the other hand, there exist some instances indicating that the integrality gap is at most $1/3 + \epsilon$ for any $\epsilon > 0$ even when the objective function is modular. Suppose that the capacity is $T > 1$ with probability 1 and there are three items: two items i and j of size T, and an item k of size 1. The weights of all items are 1, and the objective value is defined as the sum of the weights of chosen items. Clearly a knapsack of capacity T can include at most one of the items, and hence the maximum objective value of integer solutions is 1. On the other hand, a fractional solution defined by $x_{0i} = 1$, $x_{0j} = (T-1)/T$, $x_{T-1,k} = 1$ and setting the other variables be 0 achieves the objective value $3 - 1/T$.

Conversion into a polynomial time algorithm. We transform the pseudopolynomial algorithm into polynomial time. Let $W = f(I)$ and $w = \min_{i \in I} f(\{i\})$. We assume $w > 0$ without loss of generality; if $f(\{i\}) = 0$ for some item $i \in I$, we can safely remove i from I because the submodularity implies $f(S) = f(S \setminus \{i\})$ for any $S \subseteq I$ with $i \in S$. Recall that we assume that the submodular function f is given as an oracle. Indeed, algorithms in this paper can be implemented if we can compute the value of the function (or the value of its multilinear extension). We assume that the oracle is encoded in $\Omega(\log(W/w))$, and hence we say that an algorithm runs in polynomial time if its running time is expressed as a polynomial in $\log(W/w)$.

The idea for the conversion is to use a more compact relaxation, which is obtained by defining variables and constraints for a polynomial number of integers in $[T]$. Let ϵ be a positive constant smaller than 1, and let $\eta = \lfloor \log_{1-\epsilon}(\epsilon w/(W \log T)) \rfloor$. For each $t \in [T-1]$, let $\bar{p}(t)$ denote $\sum_{t'=t+1}^{T} p(t')$. We first define $\{\tau_0, \tau_1, \ldots, \tau_q, \tau_{q+1}\} \subseteq [T]$ such that $q = O(\log T + \log(W/(w\epsilon)))$, $\tau_0 = 0$, $\tau_1 = 1$, $\tau_{q+1} = T$, $\tau_j < \tau_{j+1} \leq 2\tau_j$ holds for any $j = 1, \ldots, q$, and there exists $q_\eta \in \{1, \ldots, q\}$ satisfying the following conditions:

- $\bar{p}(\tau_j) \geq \bar{p}(\tau_{j+1} - 1) \geq (1 - \epsilon)\bar{p}(\tau_j)$ for any $j \in [q_\eta]$, and
- $\bar{p}(\tau_j) < \epsilon w/(W \log T)$ for any $j \in \{q_\eta + 1, \ldots, q\}$.

Such a subset of $[T]$ can be defined as follows. For $j \in \{1, \ldots, \eta + 1\}$, let τ'_j be the

minimum integer $t \in [T-1]$ such that $\bar{p}(t) < (1-\epsilon)^{j-1}$. We assume without loss of generality that $\bar{p}(\min_{i\in I} s(i)) = 1$, which means $\tau'_1 \geq \min_{i\in I} s(i)$. We denote the set of positive integers in $\{\tau'_j : j = 1, \ldots, \eta+1\} \cup \{2^j : j \in [\lceil \log T \rceil - 1]\}$ by $\{\tau_1, \ldots, \tau_q\}$ and sort those integers so that $1 = \tau_1 < \tau_2 < \cdots < \tau_q$. We define q_η so that $\tau_{q_\eta} = \tau'_{\eta+1}$. Then, the obtained subset satisfies the above conditions.

In addition, we define the set of integers in $\{\tau_0, \ldots, \tau_{q+1}\} \cup \{\tau_j - s(i) + 1 : j \in \{1, \ldots, q+1\}, i \in I\}$ as $\{\xi_0, \ldots, \xi_{r+1}\}$, where $0 = \xi_0 < \xi_1 < \cdots < \xi_r < \xi_{r+1} = T$. Notice that $r = O(n \log T + n \log(W/(w\epsilon)))$.

Roughly speaking, we define variables for each ξ_k ($k \in [r]$) and constraints for each τ_j ($j \in [q]$). Specifically, a variable y_{ki} is defined for each $k \in [r]$ and $i \in I$, and y_{ki} replaces variables $x_{\xi_k,i}, \ldots, x_{\xi_{k+1}-1,i}$ in (5.1). For $j = 1, \ldots, q+1$ and $i \in I$, we define an auxiliary variable z_{ji} as $\sum_{\xi_{k+1}-1 \leq \tau_j - s(i)} y_{ki}$ and define z_j as the $|I|$-dimensional vector whose component corresponding to $i \in I$ is z_{ji}. Then, the compact relaxation is described as

$$
\begin{aligned}
\text{maximize} \quad & \sum_{j=0}^{q} \bar{p}(\tau_j)(F(z_{j+1}) - F(z_j)) & \\
\text{subject to} \quad & \sum_{k\in[r]} y_{ki} \leq 1 & \forall i \in I, \\
& \sum_{i\in I} \sum_{\xi_k < \tau_j} y_{ki} \min\{s(i), \tau_j\} \leq 2\tau_j & \forall j \in \{1, \ldots, q\}, \\
& z_{ji} = \sum_{\xi_{k+1}-1 \leq \tau_j - s(i)} y_{ki} & \forall j \in [q], \\
& y_{ki} \geq 0 & \forall i \in I, \forall k \in [r].
\end{aligned} \tag{5.2}
$$

LEMMA 5.5. *The optimal objective value of* (5.2) *is not smaller than that of* (5.1).

Proof. Suppose that x is a feasible solution for (5.1). From x, we define a solution y for (5.2) so that $y_{ki} = \sum_{t=\xi_k}^{\xi_{k+1}-1} x_{ti}$ for each $k \in [r]$ and $i \in I$. We define variables z from y by the third constraints of (5.2). Then, (y, z) is feasible for (5.2). Indeed, it is immediate from the feasibility of x in (5.1) that (y, z) satisfies all but the second of the constraints of (5.2). As for the second constraint, we can observe that $\sum_{\xi_k < \tau_j} y_{ki} = \sum_{t<\tau_j} x_{ti}$ holds for any $i \in I$ and $j \in \{1, \ldots, q\}$ by the definition of y and the fact that τ_j is included in $\{\xi_1, \ldots, \xi_r\}$.

Let us show that the objective value of (y, z) in (5.2) is not smaller than $\bar{F}(x)$. We have

$$
\begin{aligned}
\bar{F}(x) &= \sum_{t=1}^{T} p(t) F(\bar{x}_t) \bar{p}(T-1) F(\bar{x}_T) + \sum_{t=1}^{T-1} (\bar{p}(t-1) - \bar{p}(t)) F(\bar{x}_t) \\
&= \sum_{t=0}^{T-1} \bar{p}(t)(F(\bar{x}_{t+1}) - F(\bar{x}_t)),
\end{aligned}
$$

where $\bar{x}_0$ denotes the zero-vector for convention. The right-hand side can be written

as

$$\sum_{t=0}^{T-1} \bar{p}(t)(F(\bar{x}_{t+1}) - F(\bar{x}_t)) = \sum_{j=0}^{q} \sum_{t=\tau_j}^{\tau_{j+1}-1} \bar{p}(t)(F(\bar{x}_{t+1}) - F(\bar{x}_t))$$
$$\leq \sum_{j=0}^{q} \sum_{t=\tau_j}^{\tau_{j+1}-1} \bar{p}(\tau_j)(F(\bar{x}_{t+1}) - F(\bar{x}_t))$$
$$= \sum_{j=0}^{q} \bar{p}(\tau_j)(F(\bar{x}_{\tau_{j+1}}) - F(\bar{x}_{\tau_j})). \tag{5.3}$$

Recall that $\bar{x}_{\tau_j i} = \sum_{t \leq \tau_j - s(i)} x_{ti}$ for each $j \in [q]$ and $i \in I$. There exists $k' \in [r]$ such that $\tau_j - s(i) + 1 = \xi_{k'}$, and hence $\sum_{t \leq \tau_j - s(i)} x_{ti}$ can be written as $\sum_{\xi_k - 1 \leq \tau_j - s(i)} y_{ki} = z_{ji}$. Thus $\bar{x}_{\tau_j} = z_j$ holds for each $j \in [q]$, implying that (5.3) is equal to the objective value of (y, z). ∎

LEMMA 5.6. *From a feasible solution to* (5.2) *achieving the objective value* θ*, we can construct a feasible solution to* (5.1) *achieving the objective value of at least* $(1-\epsilon)(\theta - \epsilon w)/2$*.*

Proof. Let (y, z) be a feasible solution to (5.2). For each $k \in [r]$, $i \in I$, and $t \in \{\xi_k, \ldots, \xi_{k+1} - 1\}$, we define x_{ti} as $y_{ki}/(\xi_{k+1} - \xi_k)$.

We prove that $x/2$ is feasible for (5.1). It is immediate from the definition of x that x satisfies the first and third constraints of (5.1). We focus on the second constraints. Let $t \in \{1, \ldots, T-1\}$. Suppose that $\tau_j \leq t < \tau_{j+1}$ holds for some $j \in [q]$. Then, for each $i \in I$, we have $\sum_{t' \in [t]} x_{t'i} \leq \sum_{t' < \tau_{j+1}} x_{t'i} = \sum_{\xi_k < \tau_{j+1}} y_{ki}$, where the equality follows from the fact that $\tau_{j+1} \in \{\xi_1, \ldots, \xi_{r+1}\}$. Moreover, $\min\{s(i), t\} \leq \min\{s(i), \tau_{j+1}\}$ also holds. Hence,

$$\sum_{i \in I} \sum_{t' \in [t]} x_{t'i} \min\{s(i), t\} \leq \sum_{i \in I} \sum_{\xi_k < \tau_{j+1}} y_{ki} \min\{s(i), \tau_{j+1}\} \leq 2\tau_{j+1}$$

holds, where the last inequality follows from the second constraint of (5.2) when $j < q$, and from $\tau_{q+1} = T = \sum_{i \in I} s(i)$ and the first constraints of (5.2) when $j = q$. By its definition, $t \geq \tau_j \geq \tau_{j+1}/2$. This implies that $x/2$ is feasible to (5.1).

Let θ be the objective value of (y, z) in (5.2). We show that the objective value $\bar{F}(x/2)$ of $x/2$ in (5.1) is at least $(1-\epsilon)(\theta - \epsilon w)/2$. We observe that

$$\bar{F}\left(\frac{x}{2}\right) = \sum_{t=1}^{T} p(t) F\left(\frac{\bar{x}_t}{2}\right) \geq \frac{1}{2} \sum_{t=1}^{T} p(t) F\left(\bar{x}_t\right) = \frac{1}{2} \sum_{j=0}^{q} \sum_{t=\tau_j}^{\tau_{j+1}-1} \bar{p}(t) \left(F\left(\bar{x}_{t+1}\right) - F\left(\bar{x}_t\right)\right).$$

Since $\bar{p}(t) \geq \bar{p}(\tau_{j+1} - 1)$ for any $t \leq \tau_{j+1} - 1$, the right-hand side of the above inequality satisfies

$$\frac{1}{2} \sum_{j=0}^{q} \sum_{t=\tau_j}^{\tau_{j+1}-1} \bar{p}(t) \left(F\left(\bar{x}_{t+1}\right) - F\left(\bar{x}_t\right)\right) \geq \frac{1}{2} \sum_{j=0}^{q} \bar{p}(\tau_{j+1} - 1) \sum_{t=\tau_j}^{\tau_{j+1}-1} \left(F\left(\bar{x}_{t+1}\right) - F\left(\bar{x}_t\right)\right)$$
$$= \frac{1}{2} \sum_{j=0}^{q} \bar{p}(\tau_{j+1} - 1) \left(F\big(\bar{x}_{\tau_{j+1}}\big) - F\big(\bar{x}_{\tau_j}\big)\right).$$

Recall that $\bar{x}_{\tau_j} = z_j$ and $\bar{p}(\tau_{j+1} - 1) \geq (1-\epsilon)\bar{p}(\tau_j)$ hold for any $j \in [q_\eta]$. Therefore,

$$\frac{1}{2}\sum_{j=0}^{q} \bar{p}(\tau_{j+1}-1)\left(F\left(\bar{x}_{\tau_{j+1}}\right) - F\left(\bar{x}_{\tau_j}\right)\right) \geq \frac{1-\epsilon}{2}\sum_{j=0}^{q_\eta} \bar{p}(\tau_j)\left(F\left(z_{j+1}\right) - F\left(z_j\right)\right).$$

On the other hand, θ can be written as

$$\begin{aligned}\theta &= \sum_{j=0}^{q} \bar{p}(\tau_j)(F(z_{j+1}) - F(z_j)) \\ &= \sum_{j=0}^{q_\eta} \bar{p}(\tau_j)(F(z_{j+1}) - F(z_j)) + \sum_{j=q_\eta+1}^{q} \bar{p}(\tau_j)(F(z_{j+1}) - F(z_j)).\end{aligned}$$

Recall that $\bar{p}(\tau_j) \leq \epsilon w/(W \log T)$ if $j \geq q_\eta + 1$. Moreover, we have $q - q_\eta \leq \log T$, and hence,

$$\sum_{j=q_\eta+1}^{q} \bar{p}(\tau_j)(F(z_{j+1}) - F(z_j)) \leq \sum_{j=q_\eta+1}^{q} \bar{p}(\tau_j) W \leq \epsilon w.$$

Combining all these discussions, we have $\bar{F}(x/2) \geq (1-\epsilon)(\theta - \epsilon w)/2$. ∎

We now wrap up our algorithm. Our algorithm first applies the continuous greedy algorithm with stopping time $1/4$ to compute a solution y such that $4y$ is feasible for (5.2) and the objective value of y in (5.2) is $1-1/\sqrt[4]{e}$ times that of any feasible solution, particularly the optimal value h of (5.2). From y, we compute a solution x for (5.1) by Lemma 5.6. We see that $4x$ is feasible for (5.1), and the objective value of x in (5.1) is at least $(1-\epsilon)((1-1/\sqrt[4]{e})h - \epsilon w)/2$. Since we are assuming $\bar{p}(\min_{i\in I} s(i)) \geq 1$, picking the item of the smallest size at time 0 achieves objective value w. This means $h \geq w$, and hence the objective value of x is at least $(1-\epsilon)(1-\epsilon-1/\sqrt[4]{e})h/2$. Then, applying the rounding algorithm to $4x$, we obtain a sequence of objective value $(1-\epsilon)(1-1/\sqrt[4]{e}-\epsilon)h/4$.

To make this algorithm run in polynomial time, we do not explicitly write down x. In the rounding algorithm, values of x are used for deciding t_i for each $i \in I$ in the first round of the rounding algorithm. This is possible without writing down x as follows. Notice that x_{ti} takes the same value for any $t \in [\tau_j, \tau_{j+1})$ by the construction of x. Hence each i chooses t_i as follows. First, i chooses $k \in [q]$ with probability y_{ki} and is discarded with probability $1 - \sum_{k\in[r]} y_{ki}$. Then, i chooses t_i from $[\tau_k, \tau_{k+1})$ uniformly at random. This algorithm runs in polynomial time with respect to $1/\epsilon$ and the input size of the instance. We give a pseudocode of the algorithm in Algorithm 5.

With the conversion given above, we obtain the following theorem.

THEOREM 5.7. *For any constant $\epsilon \in (0,1)$, there exists a randomized approximation algorithm of approximation ratio $(1-\epsilon)(1-\epsilon-1/\sqrt[4]{e})/4 \approx 0.055$ for* SMPSC*, which runs in polynomial time with respect to $1/\epsilon$ and the input size of the instance.*

6. Conclusion. We considered the maximization problem of a nonnegative monotone submodular function under an unknown or a stochastic knapsack constraint. We presented adaptive policies that achieve constant robustness ratios for an unknown knapsack constraint when cancellation is allowed. For the case where cancellation is not allowed, we presented approximation algorithms that achieve constant approximation ratios for a stochastic knapsack constraint.

There still remain many interesting directions for further study. We mention two of them. First, even for KPUC with cancellation, there is still a gap between the best

Algorithm 5: Randomized algorithm of approximation ratio $(1-\epsilon)(1-\epsilon-1/\sqrt[4]{e})/4$.

for $\forall j \in \{1,\dots,\eta+1\}$ **do**
  compute $\tau'_j = \arg\min\{t \in [T-1] \colon \bar{p}(t) < (1-\epsilon)^{j-1}\}$ by the binary search;
compute $\tau_0,\dots,\tau_{q+1}$, q_η, and $\xi_0,\dots,\xi_{r+1}$;
$y \leftarrow$ output of the continuous greedy with stopping time $1/4$ applied to (5.2);
$I' \leftarrow \emptyset$;
for $i \in I$ **do**
  choose a number k from $[q]$ with probability y_{ki} or $I' \leftarrow I' \cup \{i\}$ with probability $1 - \sum_{k\in[q]} y_{ki}$;
  if $i \notin I'$ **then** choose an integer t_i from $[\tau_k, \tau_{k+1})$ uniformly at random;
$\Pi' \leftarrow$ sequence obtained by sorting the items $i \in I \setminus I'$ in nondecreasing order of t_i;
$\Pi \leftarrow (\Pi'_1)$, $l \leftarrow 2$;
for $i = 2,\dots,|\Pi'|$ **do**
  if $\sum_{j\in[i-1]} s(\Pi'_j) < t_{\Pi'_i}$ **then** $\Pi_l \leftarrow \Pi'_i$, $l \leftarrow l+1$;
  else $I' \leftarrow I' \cup \{\Pi'_i\}$;
append the items in I' to the suffix of Π arbitrarily, and **return** Π;

known upper and lower bounds on the robustness ratio if we consider randomized policies; the best known lower bound is $1/2$ achieved by the deterministic policy of Disser et al. [8], and we give an upper bound $(25+15\cdot 2^{1/3}+9\cdot 2^{2/3})/71$ (<0.820) in subsection 4.1. Hence it is an interesting direction to investigate whether there exists a randomized policy achieving a robustness ratio better than $1/2$.

Another interesting direction is to investigate an upper bound on the robustness ratio of SMPUC with cancellation. The best known upper bound on the robustness ratio achieved by the deterministic policies is $1/2$, which is given by an instance of KPUC. Although this is tight for deterministic policies to KPUC even if they are restricted to universal policies, it may be possible to give a smaller upper bound if we consider submodular objective functions. It is interesting to investigate whether an upper bound smaller than $1/2$ is achievable for SMPUC.

REFERENCES

[1] M. Adamczyk, M. Sviridenko, and J. Ward, *Submodular stochastic probing on matroids*, Math. Oper. Res., 41 (2016), pp. 1022–1038.

[2] A. Asadpour and H. Nazerzadeh, *Maximizing stochastic monotone submodular functions*, Management Sci., 62 (2016), pp. 2374–2391.

[3] A. Asadpour, H. Nazerzadeh, and A. Saberi, *Stochastic submodular maximization*, in Proceedings of the 4th International Workshop on Internet and Network Economics (WINE 2008), Shanghai, China, 2008, pp. 477–489.

[4] A. Bernstein, Y. Disser, and M. Gross, *General Bounds for Incremental Maximization*, preprint, https://arxiv.org/abs/1705.10253, 2017.

[5] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, *Maximizing a monotone submodular function subject to a matroid constraint*, SIAM J. Comput., 40 (2011), pp. 1740–1766, https://doi.org/10.1137/080733991.

[6] M. Dabney, *A PTAS for the Uncertain Capacity Knapsack Problem*, Master's thesis, Clemson University, 2010, http://tigerprints.clemson.edu/all_theses/982.

[7] B. C. Dean, M. X. Goemans, and J. Vondrák, *Approximating the stochastic knapsack problem: The benefit of adaptivity*, Math. Oper. Res., 33 (2008), pp. 945–964.

[8] Y. Disser, M. Klimm, N. Megow, and S. Stiller, *Packing a knapsack of unknown capacity*, SIAM J. Discrete Math., 31 (2017), pp. 1477–1497, https://doi.org/10.1137/16M1070049.

[9] L. Epstein, A. Levin, A. Marchetti-Spaccamela, N. Megow, J. Mestre, M. Skutella, and L. Stougie, *Universal sequencing on an unreliable machine*, SIAM J. Comput., 41 (2012), pp. 565–586, https://doi.org/10.1137/110844210.

[10] M. Feldman, *Maximization Problems with Submodular Objective Functions*, Ph.D. thesis, Technion–Israel Institute of Technology, 2013.

[11] M. Feldman, J. Naor, and R. Schwartz, *A unified continuous greedy algorithm for submodular maximization*, in Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS 2011), Palm Springs, CA, 2011, pp. 570–579.

[12] M. Feldman, O. Svensson, and R. Zenklusen, *Online contention resolution schemes*, in Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2016), Arlington, VA, 2016, pp. 1014–1033, https://doi.org/10.1137/1.9781611974331.ch72.

[13] D. Golovin and A. Krause, *Adaptive submodularity: Theory and applications in active learning and stochastic optimization*, J. Artificial Intelligence Res., 42 (2011), pp. 427–486.

[14] A. Gupta, R. Krishnaswamy, M. Molinaro, and R. Ravi, *Approximation algorithms for correlated knapsacks and non-martingale bandits*, in IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS 2011), Palm Springs, CA, 2011, pp. 827–836.

[15] A. Gupta, V. Nagarajan, and S. Singla, *Adaptivity gaps for stochastic probing: Submodular and XOS functions*, in Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017), Barcelona, Spain, 2017, pp. 1688–1702, https://doi.org/10.1137/1.9781611974782.111.

[16] W. Höhn and T. Jacobs, *On the performance of Smith's rule in single-machine scheduling with nonlinear cost*, ACM Trans. Algorithms, 11 (2015), 25.

[17] N. Kakimura, K. Makino, and K. Seimi, *Computing knapsack solutions with cardinality robustness*, Japan J. Indust. Appl. Math., 29 (2012), pp. 469–483.

[18] Y. Kobayashi and K. Takazawa, *Randomized strategies for cardinality robustness in the knapsack problem*, Theoret. Comput. Sci., 699 (2017), pp. 53–62.

[19] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance, *Cost-effective outbreak detection in networks*, in Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, CA, 2007, pp. 420–429.

[20] W. Ma, *Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms: Extended abstract*, in Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014), Portland, OR, 2014, pp. 1154–1163, https://doi.org/10.1137/1.9781611973402.85.

[21] N. Megow and J. Mestre, *Instance-sensitive robustness guarantees for sequencing with unknown packing and covering constraints*, in Innovations in Theoretical Computer Science (ITCS '13), Berkeley, CA, 2013, pp. 495–504.

[22] N. Megow and J. Verschae, *Dual techniques for scheduling on a machine with varying speed*, SIAM J. Discrete Math., 32 (2018), pp. 1541–1571, https://doi.org/10.1137/16M105589X.

[23] M. Sviridenko, *A note on maximizing a submodular set function subject to a knapsack constraint*, Oper. Res. Lett., 32 (2004), pp. 41–43.

[24] A. C. Yao, *Probabilistic computations: Toward a unified measure of complexity (extended abstract)*, in Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS 1977), Providence, RI, 1977, pp. 222–227.