MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

Computing Knapsack Solutions with Cardinality Robustness

Naonori KAKIMURA, Kazuhisa MAKINO,
and Kento SEIMI

METR 2011–31 September 2011

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.keisu.t.u-tokyo.ac.jp/research/techrep/index.html

Computing Knapsack Solutions with Cardinality Robustness*

Naonori Kakimura[†‡] Kazuhisa Makino[†‡] Kento Seimi[†]

September 2011

Abstract

In this paper, we study the robustness over the cardinality variation for the knapsack problem. For the knapsack problem and a positive number $\alpha \leq 1$, we say that a feasible solution is *α-robust* if, for any positive integer k, it includes an α-approximation of the maximum k-knapsack solution, where a k-knapsack solution is a feasible solution that consists of at most k items.

In this paper, we show that, for any $\varepsilon > 0$, the problem of deciding whether the knapsack problem admits a $(\nu + \varepsilon)$-robust solution is weakly NP-hard, where ν denotes the rank quotient of the corresponding knapsack system. Since the knapsack problem always admits a ν-robust knapsack solution [7], this result provides a sharp border for the complexity of the robust knapsack problem. On the positive side, we show that a max-robust knapsack solution can be computed in pseudo-polynomial time, and present a fully polynomial time approximation scheme (FPTAS) for computing a max-robust knapsack solution.

1 Introduction

The classical *knapsack problem* (KP) is defined as follows: We are given a set of *items* $E = \{1, \ldots, n\}$ with *price* p_i and *weight* w_i for items $i \in E$, and the capacity $C \in \mathbb{Z}_+$. The aim is to find a subset X of E that maximizes the total price of X subject to the knapsack constraint, i.e., the total weight of X is bounded by C. We denote an instance of the knapsack problem by a quadruple $I = (E, p, w, C)$.

The knapsack problem is one of the most well-studied combinatorial optimization problems. It is known that the knapsack problem is weakly NP-hard [6], but pseudo-polynomially solvable through dynamic programming [2, 14]. In addition, KP admits a *fully polynomial time approximation scheme* (*FPTAS*), i.e., for any $\varepsilon > 0$, a $(1 - \varepsilon)$-approximate solution can be computed in time polynomial in the input length and $1/\varepsilon$ [9, 13, 16, 18]. See e.g., [14, 20] for surveys on the knapsack problems.

A natural extension of KP is the *k-item knapsack problem* (*k-KP*) [1], which is a KP in which we are additionally given a cardinality constraint that bounds the number of selected items to be at most k. That is, given a knapsack instance $I = (E, p, w, C)$ and a positive integer k, the k-KP is the following problem:

$$\begin{array}{lll} k\text{-KP}: & \text{maximize} & p(X) \\ & \text{subject to} & w(X) \leq C, \\ & & |X| \leq k, \end{array}$$

*A preliminary version appears in Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011) [12].

[†]Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan. {kakimura, makino, kento_seimi}@mist.i.u-tokyo.ac.jp

[‡]This work was partially supported by Grant-in-Aid for Scientific Research and by Global COE Program "The research and training center for new development in mathematics" from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

where for a vector $u \in \mathbb{R}^E$, we define $u(X) = \sum_{i \in X} u_i$ for $X \subseteq E$. A subset X with $w(X) \leq C$ is called a *knapsack solution*, and a *k-knapsack solution* if the size is at most k in addition. If k is sufficiently large, e.g., $k = n$, then k-KP coincides with KP. Also, k-KP can be seen as a special case of the two-dimensional knapsack problem. The k-item knapsack problem is also NP-hard, but the classical dynamic programming, PTAS, and FPTAS for KP are all adapted to k-KP [1].

For k-KP, imagine the case where we do not know an integer k in advance. In such an uncertain situation, it is natural to keep a knapsack solution which has a good approximation for *all* k-KPs. Such solution is called *robust*, where we will give more precise definition as below. For a knapsack instance I, let O_k denote an optimal k-knapsack solution, and for a knapsack solution $X = \{e_1, \ldots, e_{|X|}\}$ with $p_{e_1} \geq \cdots \geq p_{e_{|X|}}$, define

$$p_{\leq k}(X) = \sum_{i \leq k} p_{e_i}, \quad k = 1, 2, \ldots, |E|.$$

For a real number α with $0 < \alpha \leq 1$, we say that a knapsack solution X is *α-robust* if $p_{\leq k}(X) \geq \alpha \cdot p(O_k)$ for any size k. We also say that a knapsack instance I is *α-robust* if it has an α-robust knapsack solution. The main purpose of this paper is to find the maximum α so that a given knapsack instance I is α-robust. A knapsack solution with the maximum α is called *max-robust*.

The robustness was first introduced by Hassin and Rubinstein [7] for the *maximum weight independent problem*, which is the problem of maximizing a linear function over an independence system. Here, an *independence system* is a family $\mathcal{F}$ of subsets in E with $\emptyset \in \mathcal{F}$ and the *hereditary property*, that is, if $X \subseteq Y \in \mathcal{F}$ then $X \in \mathcal{F}$, and it includes as a special case a variety of combinatorial objects in graphs and hypergraphs such as matchings, stable sets, and matroids. The α-robustness for independence systems is a natural generalization of the greedy property of matroids, since the greedy algorithm for the maximum weight independent problem finds a 1-robust solution if $\mathcal{F}$ is a matroid [3, 21]. For a general independence system $\mathcal{F}$, Hassin and Rubinstein [7] proved that a greedy solution is $\nu(\mathcal{F})$-robust, where $\nu(\mathcal{F})$ is the *rank quotient* defined in Section 2. Moreover, they showed that the maximum matching problem, i.e., when $\mathcal{F}$ arises from the family of matchings in a graph, admits a $1/\sqrt{2}$-robust solution, and that $1/\sqrt{2}$-robustness is the best possible for the maximum matching problem. Fujita, Kobayashi, and Makino [4] extended their matching result to the matroid intersection problem. Recently, Kakimura and Makino [11] showed the existence of a highly robust solution for general independence systems, which includes the results of both matchings and matroid intersections. The robustness was also studied for several combinatorial optimization problems such as coloring and subgraph problems [5, 8]. Another concept similar to the robustness, called the *incremental problems*, has been investigated for covering problems in connection with online algorithms [17, 19].

Given an instance I of the knapsack problem, the family of knapsack solutions forms an independence system, called the *knapsack system*. By Hassin and Rubinstein [7], we can compute a $\nu(I)$-robust solution in polynomial time, where $\nu(I)$ denotes the rank quotient of the knapsack system that corresponds to I. It is, however, *not* known whether we can efficiently compute a knapsack solution with maximum robustness, or even, with *better* robustness.

In this paper, we show that for any constant $\varepsilon > 0$, it is weakly NP-hard to decide whether a given instance I of the knapsack problem has a $(\nu(I) + \varepsilon)$-robust solution. By [7], this gives us a sharp border for the complexity of the robust knapsack problem.

We then design two kinds of pseudo-polynomial time algorithms for finding a max-robust knapsack solution. The first one is a simple extension of dynamic programming algorithms for the knapsack problem, and requires $O(n^2CU)$ time, where $U = \sum_i p_i$. We note that the time complexity depends on the knapsack capacity C, and hence it seems difficult to design an FPTAS

based on this algorithm. The second one is based on a procedure to find an α-robust solution for a given α. This procedure, together with a binary search for α, leads to an $O(n^2 U \log U)$ algorithm for finding a max-robust knapsack solution. In addition, the second algorithm leads to an FPTAS by using a cost rounding technique. That is, for a knapsack instance I and $\varepsilon > 0$, it returns a $(1-\varepsilon)\alpha(I)$-robust solution in time polynomial in the input length and $1/\varepsilon$, where $\alpha(I)$ denotes the maximum robustness for I. We note that the max-robust knapsack problem has complexity properties similar to the classical knapsack problem, where it is *not* generally true, for example, the max-robust matching problem is strongly NP-hard [4], while the matching problem is polynomially solvable. We can also say that our result for FPTAS is a first positive result for the max-robust problems, except for matroids.

This paper is organized as follows. In Section 2, we present the NP-hardness result for the robust knapsack problem. Sections 3 and 4 describe two kinds of pseudo-polynomial time algorithms for finding a max-robust knapsack solution, respectively, and in Section 5, we design an FPTAS based on the algorithm in Section 4.

2 NP-hardness for the Max-Robust Knapsack Problem

In this section, we prove the NP-hardness for finding max-robust knapsack solutions. We first provide some notations. Let $\mathcal{F}$ be an independence system. For $J \subseteq E$, let $\rho(J)$ and $\gamma(J)$ be the minimum and maximum sizes of maximal independent sets in $\mathcal{F}$ contained in J, respectively. Define the *rank quotient* $\nu(\mathcal{F})$ to be

$$\nu(\mathcal{F}) = \min_{J \subseteq E} \frac{\rho(J)}{\gamma(J)}.$$

For a knapsack instance $I = (E, p, w, C)$, let $\nu(I)$ denote the rank quotient of the knapsack system $\mathcal{F}$ corresponding to I, i.e., $\mathcal{F} = \{X \subseteq E \mid w(X) \leq C\}$. Jenkyns [10] and Korte and Hausmann [15] showed the greedy algorithm finds a $\nu(\mathcal{F})$-approximate solution for the maximum weight independent problem. Hassin and Rubinstein [7] proved a greedy solution is in fact $\nu(\mathcal{F})$-robust.

Theorem 2.1 (Hassin and Rubinstein [7])**.** *Every knapsack problem admits a ν-robust solution, where ν is the rank quotient.*

It, however, turns out to be NP-hard to decide whether or not a given knapsack instance has better than a greedy solution.

Theorem 2.2. *Let $\varepsilon < 1$ be a fixed positive constant. Problem ε-RobustKnapsack defined as below is NP-hard.*

Problem: ε-RobustKnapsack

Instance: *A knapsack instance $I = (E, p, w, C)$.*

Question: *Is there a knapsack solution X whose robustness is $\nu(I) + \varepsilon$?*

The rest of this section is devoted to prove Theorem 2.2.

We will show that the well-known NP-complete problem Partition [6] can be reduced to the ε-robust knapsack problem. The problem Partition is as follows: Given n nonnegative integers $a_1 \geq \cdots \geq a_n$, find $X \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in X} a_i = A/2$, where $A = \sum_{i=1}^{n} a_i$. The following problem is also NP-complete, which easily follows from Partition.

Problem: δ-PARTITION

Instance: n nonnegative integers $a_1 \geq \cdots \geq a_n$. Let $A = \sum_{i=1}^{n} a_i$.

Question: Is there any X such that $\sum_{i \in X} a_i = \delta A$?

Lemma 2.3. *For a fixed positive constant $\delta < 1$, Problem δ-PARTITION is NP-complete.*

Proof. It is obvious that this problem is in NP. We will show that Problem PARTITION can be reduced to this problem. Let $a_1 \geq \cdots \geq a_n$ with $A = \sum_{i=1}^{n} a_i$ be an instance I of PARTITION. We may assume that $\delta < 1/2$ and that n is sufficiently large so that $2(1-2\delta)n > 1$ and $n\delta > 1$.

We construct an instance J of δ-PARTITION as $b_1, \ldots, b_n, b_{n+1}, b_{n+2}$, where

$$\begin{aligned} b_i &= a_i \, (i = 1, \ldots, n), \\ b_{n+1} &= \delta n A - \frac{1}{2} A, \\ b_{n+2} &= (1-\delta) n A - \frac{1}{2} A. \end{aligned}$$

Then we will show that J has a solution if and only if so does I.

First assume that J has a solution $X \subseteq \{1, \ldots, n, n+1, n+2\}$. Then it holds that $\sum_{i \in X} b_i = \delta \sum_{i=1}^{n+2} b_i = \delta n A$. We observe that $n+2 \notin X$ and $n+1 \in X$, since $b_{n+2} > \delta n A$ and $\sum_{i=1}^{n} b_i = A < \delta n A$. Thus the total weight of $X \setminus \{n+1\}$ is $A/2$, which is a solution of the instance I. Conversely, if I has a solution Y, then $Y \cup \{n+1\}$ is a solution of J. Thus the statement holds. □

We slightly modify δ-PARTITION as follows.

Problem: δ-PARTITION$'$

Instance: n nonnegative integers $a_1 \geq \cdots \geq a_n$. Let $A = \sum_{i=1}^{n} a_i$.

Question: Is there any X such that $\sum_{i \in X} a_i = \delta A$ and $|X| = \delta n$?

Lemma 2.4. *For a fixed positive constant $\delta < 1$, Problem δ-PARTITION$'$ is NP-complete.*

Proof. Let J be an instance of δ-PARTITION, denoted by $a_1 \geq \cdots \geq a_n$. We add $(1/\delta - 1)n$ items with zero weights. Then this instance has a solution of δ-PARTITION$'$ if and only if the original instance has a solution of δ-PARTITION. □

Define $\delta = (1-\varepsilon)/2$. Let $a_1 \geq \cdots \geq a_n$ with $A = \sum_{i=1}^{n} a_i$ be an instance I of δ-PARTITION$'$. We can take a sufficiently large n so that

$$n \geq \frac{2}{\varepsilon}, \quad \varepsilon \, (\varepsilon + \delta) \geq \frac{1}{n-1}, \text{ and } \frac{n-1}{n+1} \geq \varepsilon + \delta,$$

e.g.,

$$n \geq \max \left\{ \frac{2}{\varepsilon} + 1, \frac{4}{1-\varepsilon} \right\}. \tag{1}$$

Define $b_i = a_i + n a_1$ for $i = 1, \ldots, n$ and $B = \sum b_i = A + n^2 a_1$, and the obtained instance $b_1 \geq \cdots \geq b_n$ of δ-PARTITION$'$ is denoted by I'. Then I has a solution if and only if so does I'. We observe that $n b_n \leq B < (n+1) b_n$, $(n-1) b_1 < B \leq n b_1$, and hence $b_1 < \frac{n+1}{n-1} b_n$ holds.

We construct an instance $J = (E, p, w, C)$ of ε-RobustKnapsack as follows: Let $E = \{0, 1, \ldots, n\}$, and define

$$\begin{aligned} p_0 &= \varepsilon B,\ w_0 = (1-\delta)B, \\ p_i &= w_i = b_i\,(i = 1, \ldots, n), \\ C &= B,\ \text{and } \alpha = \delta + \varepsilon. \end{aligned}$$

Then $p_0 \geq B/(\alpha(n-1)) > \alpha^{-1}p_1$ holds by (1) and $(n-1)b_1 < B$. Hence we have $p_0 > \alpha^{-1}p_1 > p_1 \geq \cdots \geq p_n$ and $w_0 \geq w_1 \geq \cdots \geq w_n$.

We first note that $\nu(J) = \delta$ holds. Indeed, a maximum maximal solution is $\{1, \ldots, n\}$, and a minimum maximal solution is $\{0, 1, \ldots, \delta n - 1\}$ because $\{0, 1, \ldots, \delta n\}$ is not feasible by $w_0 + \sum_{i=1}^{\delta n} w_i > (1-\delta)B + \delta B = B$.

We will claim that the instance J has an α-robust solution if and only if I' has a solution of δ-Partition$'$, which implies Theorem 2.2.

First assume that the knapsack instance J has an α-robust solution X. Since $p_0 > \alpha^{-1}p_1$, the solution X has to contain the item 0. Let $Y = X \setminus \{0\}$. Since X is α-robust and $\{1, \ldots, n\}$ is an optimal solution for this knapsack problem, we have

$$p_0 + \sum_{i \in Y} b_i \geq \alpha \sum_{i=1}^{n} b_i = \alpha B.$$

Since the knapsack constraint says that $w_0 + \sum_{i \in Y} b_i \leq B$, we have

$$B - w_0 = \delta B \geq \sum_{i \in Y} b_i \geq \alpha B - \varepsilon B = \delta B.$$

Therefore, we obtain $\sum_{i \in Y} b_i = \delta B$.

Moreover, $|Y| = \delta n$ holds. Indeed, if $|Y| > \delta n$ then $\sum_{i \in Y} b_i \geq (\delta n + 1)na_1 > \delta B$, and if $|Y| < \delta n$ then $\sum_{i \in Y} b_i \leq (\delta n - 1)b_1 \leq \delta n^2 a_1 < \delta B$, both of which are contradictions. Thus Y is a solution of the δ-Partition$'$ instance I'.

Conversely, assume that we have a solution Y of the δ-Partition$'$ instance I'. Then $X = Y \cup \{0\}$ is a feasible solution of the knapsack instance J. We will show that X is α-robust.

Let k be a positive integer and O_k be an optimal k-knapsack solution for J. Note that it holds

$$p(O_k) \leq \begin{cases} p_0 + \min\{\delta B, (k-1)b_1\} & \text{if } kb_1 \leq p_0 + \delta B \\ kb_1 & \text{if } p_0 + \delta B < kb_1 \text{ and } k \leq n-1 \\ B & \text{if } k = n. \end{cases}$$

Moreover, we have

$$p_{\leq k}(Y) \geq p_0 + \min\{\delta B, (k-1)b_n\}.$$

Note that if $\delta B \geq (k-1)b_n$, that is, $k \leq 1 + \delta\frac{B}{b_n}$, then $kb_1 < p_0 + \delta B$ holds. Indeed,

$$kb_1 \leq b_1 + \delta\frac{Bb_1}{b_n} < \left(\frac{1}{n-1} + \delta\frac{n+1}{n-1}\right)B = \left(\frac{2\delta+1}{n-1} + \delta\right)B \leq p_0 + \delta B,$$

by $\delta = (1-\varepsilon)/2$ and $\varepsilon n \geq 2$.

First assume that $k \leq 1 + \delta\frac{B}{b_n}$. Since $b_n \geq \alpha b_1$ by (1),

$$p_{\leq k}(Y) \geq p_0 + (k-1)b_n \geq \alpha(p_0 + (k-1)b_1) \geq \alpha p(O_k).$$

Next assume that $1 + \delta\frac{B}{b_n} < k$ and $kb_1 \leq p_0 + \delta B$. Then $p_{\leq k}(Y) = p(O_k) = p_0 + \delta B$, and hence $p_{\leq k}(Y) \geq \alpha p(O_k)$. Finally assume that $kb_1 > p_0 + \delta B$. Then, by $B > (n-1)b_1$, we obtain

$$p_{\leq k}(X) = p_0 + \delta B = \alpha B \geq \alpha p(O_k).$$

Thus X is α-robust.

3 Simple Pseudo-Polynomial Time Algorithm

In this section we propose a pseudo-polynomial time algorithm for finding a max-robust knapsack solution. Given a knapsack instance $I = (E, p, w, C)$, let O_k be an optimal solution of the k-knapsack problem with respect to I. We may assume that p_i's are sorted in nonincreasing order, and $p_i \leq C$ for any $i \in E$.

For $X \subseteq E$ with $X \neq \emptyset$, define $\alpha(X)$ and $\beta(X)$ as follows:

$$\alpha(X) = \min_{1 \leq k \leq n} \frac{p_{\leq k}(X)}{p(O_k)} \quad \text{and} \quad \beta(X) = \min_{1 \leq k \leq |X|} \frac{p_{\leq k}(X)}{p(O_k)}.$$

Note that $\alpha(X)$ means that X is $\alpha(X)$-robust, but not $(\alpha(X) + \varepsilon)$-robust for any $\varepsilon > 0$. The function β is related to the robustness as follows.

Proposition 3.1. *A knapsack solution X $(\neq \emptyset)$ satisfies $\alpha(X) = \min\left\{\beta(X), \dfrac{p(X)}{p(O_n)}\right\}$.*

Proof. By the definition of α we obtain

$$\alpha(X) = \min\left\{\beta(X), \min_{|X|+1 \leq k \leq n} \frac{p(X)}{p(O_k)}\right\} = \min\left\{\beta(X), \frac{p(X)}{p(O_n)}\right\},$$

where the last equation holds because of $p(O_{|X|+1}) \leq \cdots \leq p(O_n)$. □

For $1 \leq i, \ell \leq n$, $0 \leq P \leq U$, and $0 \leq W \leq C$, define

$$A(i, \ell, P, W) = \max\{\beta(X) \mid X \in \mathcal{S}(i, \ell, P, W)\},$$

where

$$\mathcal{S}(i, \ell, P, W) = \{X \mid X \subseteq \{1, \ldots, i\},\ |X| = \ell,\ p(X) = P,\ w(X) \leq W\}.$$

Then $A(i, \ell, P, W)$ is represented recursively as follows.

Lemma 3.2. *We can express $A(i, \ell, P, W)$ to be as follows:*

(i) *For $i = 1$,*

$$A(1, \ell, P, W) = \begin{cases} 1 & \text{if } \ell = 1, P = p_1, \text{ and } W \geq w_1, \\ -\infty & \text{otherwise.} \end{cases}$$

(ii) *For $2 \leq i \leq n$ and $\ell = 1$,*

$$A(i, 1, P, W) = \begin{cases} \max\left\{A(i-1, 1, P, W), \dfrac{p_i}{p_1}\right\} & \text{if } P = p_i \text{ and } W \geq w_i, \\ A(i-1, 1, P, W) & \text{otherwise.} \end{cases}$$

(iii) *For $2 \leq i, \ell \leq n$, if $P \geq p_i$ and $W \geq w_i$ then*

$$A(i, \ell, P, W) = \max\left\{\begin{array}{l} A(i-1, \ell, P, W), \\ \min\left\{A(i-1, \ell-1, P-p_i, W-w_i), \dfrac{P}{p(O_\ell)}\right\} \end{array}\right\},$$

and otherwise, $A(i, \ell, P, W) = A(i-1, \ell, P, W)$.

Proof. Since (i) and (ii) easily follow from the definition of A, we only consider (iii). Let $2 \leq i, \ell \leq n$. If $P < p_i$ or $W < w_i$ then $A(i, \ell, P, W) = A(i-1, \ell, P, W)$ by $\mathcal{S}(i, \ell, P, W) = \mathcal{S}(i-1, \ell, P, W)$. Thus assume that $P \geq p_i$ and $W \geq w_i$, and we may suppose that $\mathcal{S}(i, \ell, P, W) \neq \emptyset$. Let $X \in \mathcal{S}(i, \ell, P, W)$ be a set with $\beta(X) = A(i, \ell, P, W)$. If $i \notin X$, then $\beta(X) = A(i-1, \ell, P, W)$, and hence $A(i, \ell, P, W) = A(i-1, \ell, P, W)$ holds. If $i \in X$, then we have

$$\beta(X) = \min\left\{\beta(X \setminus \{i\}), \frac{P}{p(O_\ell)}\right\}$$

by $p_1 \geq \cdots \geq p_n$, which implies that $\beta(X) \leq \min\{A(i-1, \ell-1, P-p_i, W-w_i), P/p(O_\ell)\}$, and the equality holds if $X = Y \cup \{i\}$ with $\beta(Y) = A(i-1, \ell-1, P-p_i, W-w_i)$. Thus the statement holds. □

With the recursive equations in Lemma 3.2, we can design a dynamic programming algorithm to obtain $A(i, j, P, W)$ for all (i, j, P, W)'s. This can be done in $\mathrm{O}(n^2CU)$ time, where $U = \sum_i p_i$. Let me remark that this dynamic programming needs the optimal value of k-KP for all sizes k, which can be obtained in $\mathrm{O}(n^2U)$ time by standard dynamic programming [1]. It follows from Proposition 3.1 that the maximum robustness $\alpha(I)$ is equal to

$$\alpha(I) = \max_{\ell, P} \min\left\{A(n, \ell, P, C), \frac{P}{p(O_n)}\right\}. \tag{2}$$

Thus one can obtain $\alpha(I)$ from (2) in pseudo-polynomial time.

Let me notice that a max-robust knapsack solution can also be obtained by the above dynamic programming. Indeed, in each step of the above recursion, $A(i, \ell, P, W)$ is obtained from $A(i-1, \ell, P, W)$ or $A(i-1, \ell-1, P-p_i, W-w_i)$ with a new item i. Hence, associated with $A(i, \ell, P, W)$ we store a pointer to the previous entry and a pointer to the item i if we add i. Then, by traversing these pointers from $A(n, \ell, P, C)$ with $\alpha(I) = \min\{A(n, \ell, P, C), \frac{P}{p(O_n)}\}$, one can obtain an $\alpha(I)$-robust knapsack solution in linear time.

Theorem 3.3. *A dynamic programming algorithm based on Lemma 3.2 finds a max-robust knapsack solution in* $\mathrm{O}(n^2CU)$ *time, where* $n = |E|$ *and* $U = \sum_i p_i$.

4 Improved Pseudo-Polynomial Time Algorithm

In this section, we present another pseudo-polynomial time algorithm, whose time complexity does not depend on the item weights. We first develop a procedure EXIST so that, for a given knapsack instance $I = (E, p, w, C)$ and a given α, it decides whether I is α-robust or not and returns an α-robust solution if exists. Using this procedure, we design an algorithm for finding a max-robust solution with the aid of a binary search for α. Recall that O_k denotes a maximum k-knapsack solution for a positive integer k, and that p_i's are sorted in nonincreasing order.

For $1 \leq i, \ell \leq n$ and $1 \leq P \leq U$, let

$$\mathcal{T}(i, \ell, P) = \{X \mid X \subseteq \{1, \ldots, i\},\ |X| = \ell,\ p(X) = P,\ \beta(X) \geq \alpha\}.$$

We first observe the following proposition.

Proposition 4.1. *Let* $2 \leq i, \ell \leq n$. *If* $P < \alpha P(O_\ell)$ *then* $\mathcal{T}(i, \ell, P) = \emptyset$, *and, otherwise, i.e., if* $P \geq \alpha P(O_\ell)$, *it holds that*

$$\mathcal{T}(i, \ell, P) = \begin{cases} \mathcal{T}(i-1, \ell, P) \cup \{Z \cup \{i\} \mid Z \in \mathcal{T}(i-1, \ell-1, P-p_i)\} & \text{if } P \geq p_i, \\ \mathcal{T}(i-1, \ell, P) & \text{if } P < p_i, \end{cases}$$

Proof. If $P < \alpha P(O_\ell)$ then $\mathcal{T}(i,\ell,P) = \emptyset$ by the definition of $\mathcal{T}$. Suppose that $P \geq \alpha p(O_\ell)$. If $P < p_i$ then $i \notin X$ for any $X \in \mathcal{T}(i,\ell,P)$, and hence $\mathcal{T}(i,\ell,P) = \mathcal{T}(i-1,\ell,P)$ holds. Next assume that $P \geq p_i$. For $X \in \mathcal{T}(i,\ell,P)$, if $i \notin X$ then $X \in \mathcal{T}(i-1,\ell,P)$, and otherwise, we have

$$\beta(X) = \min\left\{\beta(X \setminus \{i\}), \frac{P}{P(O_\ell)}\right\},$$

which follows from that p_i's are nonincreasing order. Since $P \geq \alpha p(O_\ell)$, this implies that $\beta(X) \geq \alpha$ if and only if $X \setminus \{i\} \in \mathcal{T}(i-1,\ell-1,P-p_i)$. Thus the statement holds. □

For $1 \leq i,\ell \leq n$ and $0 \leq P \leq U$, let us define

$$B(i,\ell,P) = \min\{w(X) \mid X \in \mathcal{T}(i,\ell,P)\}.$$

We show the following recursive equation by Proposition 4.1.

Lemma 4.2. *We can express $B(i,\ell,P)$ as follows:*

(i) *For $i = 1$,*

$$B(1,\ell,P) = \begin{cases} w_1 & \text{if } \ell = 1 \text{ and } P = p_1 \\ \infty & \text{otherwise.} \end{cases}$$

(ii) *For $2 \leq i \leq n$ and $\ell = 1$,*

$$B(i,1,P) = \begin{cases} \min\{B(i-1,1,P), w_i\} & \text{if } p_i \geq \alpha p_1 \text{ and } p_i = P \\ B(i-1,1,P) & \text{otherwise.} \end{cases}$$

(iii) *For $2 \leq i,\ell \leq n$, if $P < \alpha P(O_\ell)$ then $B(i,\ell,P) = \infty$, and otherwise, i.e., if $P \geq \alpha P(O_\ell)$,*

$$B(i,\ell,P) = \begin{cases} \min\{B(i-1,\ell,P), B(i-1,\ell-1,P-p_i) + w_i\} & \text{if } P \geq p_i \\ B(i-1,\ell,P) & \text{if } P < p_i \end{cases}$$

Proof. By the definition of B, (i) and (ii) hold. Let $2 \leq i,\ell \leq n$. By Proposition 4.1, it is not difficult to see the cases where $P < \alpha P(O_\ell)$ and where $P \geq \alpha p(O_\ell)$ and $P < p_i$. If $P \geq \alpha p(O_\ell)$ and $P \geq p_i$, Proposition 4.1 implies that $B(i,\ell,P)$ is the minimum of $\min\{w(Y) \mid Y \in \mathcal{T}(i-1,\ell,P)\}$ and $\min\{w(Z \cup \{i\}) \mid Z \in \mathcal{T}(i-1,\ell-1,P-p_i)\}$. The former is equal to $B(i-1,\ell,P)$, while the latter is $B(i-1,\ell-1,P-p_i) + w_i$. Thus the lemma holds. □

Procedure EXIST can be designed as follows. In the first step, we compute the optimal values of k-KP for all $k = 1,\ldots,n$ in $\mathrm{O}(n^2U)$ time [1], where $U = \sum_i p_i$. With the recursive equation in Lemma 4.2, we can design a dynamic programming algorithm to obtain $B(i,\ell,P)$ for all (i,ℓ,P)'s. By Proposition 3.1, there exists an α-robust solution X with $|X| = \ell$ and $p(X) = P$ if and only if $B(n,\ell,P) \leq C$ and $P \geq \alpha p(O_n)$. Thus we can decide whether there exists an α-robust solution or not from all $B(i,j,P)$'s. Moreover, an α-robust knapsack solution can be found in a similar way to the algorithm of Theorem 3.3.

Theorem 4.3. *Procedure* EXIST *decides whether there exists an α-robust knapsack solution or not and returns one if exists. Its time complexity is $\mathrm{O}(n^2U)$, where $n = |E|$ and $U = \sum_i p_i$.*

Using this procedure, we can find a max-robust knapsack solution with the aid of a binary search.

Theorem 4.4. *We can find a max-robust knapsack solution in $\mathrm{O}(n^2U \log U)$ time, where $n = |E|$ and $U = \sum_i p_i$.*

Proof. The algorithm is described as follows. Let $I = (E, p, w, C)$ be a given knapsack instance. First initialize $\alpha = 1/2$. We repeat the following for $r = 1, 2, \ldots$: Apply Procedure EXIST to find an α-robust knapsack solution for I. If such a solution exists, set $\alpha = \alpha + 2^{-r-1}$, and otherwise set $\alpha = \alpha - 2^{-r-1}$.

We claim that the number of the repetition is at most $\lceil 2 \log_2 U \rceil$. Indeed, for two knapsack solutions X, Y with $\alpha(X) > \alpha(Y)$, the difference is

$$\alpha(X) - \alpha(Y) = \min_k \frac{p_{\leq k}(X)}{p(O_k)} - \min_k \frac{p_{\leq k}(Y)}{p(O_k)} > \frac{1}{U^2}.$$

Hence it suffices to repeat $\lceil 2 \log_2 U \rceil$ times to distinguish any two knapsack solutions with different robustness. Since EXIST requires $\mathrm{O}(n^2 U)$ time by Theorem 4.3, the total time complexity is $\mathrm{O}(n^2 U \log U)$. □

5 Fully Polynomial Time Approximation Scheme

In this section, we propose an FPTAS for the max-robust knapsack problem. This scheme is obtained from the second pseudo-polynomial time algorithm presented in Section 4 by rounding prices.

We first describe the outline of the scheme, denoted by Approx, as below.

Approximation scheme for the max-robust knapsack problem: Approx

Input: A knapsack instance $I = (E, p, w, C)$ and $0 < \varepsilon < 1$.

Step 1: (Rounding) Define $p'_i = \left\lfloor \frac{p_i}{N} \right\rfloor$ for $i = 1, \ldots, n$, where

$$N = \frac{p_1 \delta}{n} \quad \text{and} \quad \delta = 1 - \sqrt{1 - \varepsilon}. \tag{3}$$

Let $I' = (E, p', w, C)$ be the rounded instance.

Step 2: (Pseudo-polynomial time algorithm) Execute the algorithm in Theorem 4.4 for I', and return a knapsack solution X which is max-robust with respect to I'.

The main purpose of this section is to show Approx is an FPTAS.

Theorem 5.1. *Scheme* Approx *is an FPTAS for finding a max-robust knapsack solution. That is, it returns a $(1-\varepsilon)\alpha(I)$-robust knapsack solution in time polynomial in the input length and $1/\varepsilon$.*

To prove Theorem 5.1, we first discuss the time complexity.

Lemma 5.2. *Scheme* Approx *runs in $\mathrm{O}(n^4 \frac{1}{\varepsilon} \log(\frac{n}{\varepsilon}))$ time, where n is the number of items.*

Proof. For $i = 1, \ldots, n$, it holds

$$p'_i \leq \frac{p_i}{N} = \frac{p_i}{p_1} \frac{n}{\delta} \leq \frac{n}{\delta},$$

where the last inequality follows from $p_1 \geq p_i$ for all i's. Note that $1/\delta \leq 2/\varepsilon$, and hence $p'_i \leq 2n/\varepsilon$ holds. Therefore, it follows from Theorem 4.4 that Step 2 requires $\mathrm{O}(n^4 \frac{1}{\varepsilon} \log(\frac{n}{\varepsilon}))$, and thus so is the total time complexity. □

We next estimate the approximation factor of the output. For that purpose, we show the following lemma. For the rounded instance I', let O'_k be an optimal k-knapsack solution with respect to I'. For a knapsack solution X with $X \neq \emptyset$, we denote by $\alpha'(X)$ the robustness of X with respect to the rounded instance I', that is,

$$\alpha'(X) = \min_{1 \leq k \leq n} \frac{p'_{\leq k}(X)}{p'(O'_k)}.$$

Lemma 5.3. *The following statements hold.*

(a) *Let Y be a max-robust solution of I'. Then $\alpha(Y) \geq \sqrt{1-\varepsilon} \cdot \alpha'(Y)$.*

(b) *Let Z be a max-robust solution of I. Then $\alpha'(Z) \geq \sqrt{1-\varepsilon} \cdot \alpha(Z)$.*

We first provide the following observations to prove Lemma 5.3.

Lemma 5.4. *For any $X \subseteq E$, it holds that*

$$p(X) \geq Np'(X) \geq p(X) - N|X|.$$

Proof. This easily follows from the definition of p'_i's. □

Lemma 5.5. *For $k = 1, \ldots, n$, it holds that*

$$\frac{p(O_k)}{k} \geq \frac{p(O_n)}{n}.$$

Proof. If $|O_n| \leq k$ then $p(O_k) = p(O_n)$, and hence the statement clearly holds. If $|O_n| > k$ then

$$p(O_k) \geq p_{\leq k}(O_n) \geq \frac{k}{|O_n|} p(O_n) \geq \frac{k}{n} p(O_n),$$

and thus the statement holds. □

We are now ready to show Lemma 5.3.

Proof of Lemma 5.3. (a) Let $k \geq 1$. Since Y is $\alpha'(Y)$-robust with respect to I', it holds that $p'_{\leq k}(Y) \geq \alpha'(Y) \cdot p'(O'_k)$. Hence we have by Lemma 5.4,

$$p_{\leq k}(Y) \geq Np'_{\leq k}(Y) \geq N \cdot \alpha'(Y) p'(O'_k) \geq \alpha'(Y)(p(O_k) - Nn),$$

where the last inequality follows from $p'(O'_k) \geq p'(O_k)$. Therefore, (3) implies that

$$p_{\leq k}(Y) \geq \alpha'(Y) \left(1 - \frac{p_1}{p(O_k)} \delta\right) p(O_k) \geq \sqrt{1-\varepsilon} \cdot \alpha'(Y) p(O_k),$$

which follows from $p_1 \leq p(O_k)$. This means that Y is $\sqrt{1-\varepsilon} \cdot \alpha'(Y)$-robust with respect to I.

(b) Let $k \geq 1$. Since Z is $\alpha(Z)$-robust with respect to I, by Lemma 5.4, we have

$$Np'_{\leq k}(Z) \geq p_{\leq k}(Z) - Nk \geq \alpha(Z) p(O_k) - Nk = \left(1 - \frac{Nk}{\alpha(Z) p(O_k)}\right) \alpha(Z) p(O_k).$$

By $p(O_k) \geq p(O'_k)$ and $p(O'_k) \geq Np'(O'_k)$, we obtain

$$Np'_{\leq k}(Z) \geq \left(1 - \frac{Nk}{\alpha(Z) p(O_k)}\right) \alpha(Z) Np'(O'_k). \tag{4}$$

By Lemma 5.5 and (3), it holds

$$\frac{Nk}{\alpha(Z)p(O_k)} = \frac{k}{p(O_k)}\frac{p(O_n)}{n}\frac{p_1}{\alpha(Z)p(O_n)}\delta \leq \frac{p_1}{\alpha(Z)p(O_n)}\delta. \tag{5}$$

Moreover, since $\{1\}$ is a knapsack solution, the maximum robustness of Z implies

$$\frac{p_1}{p(O_n)} = \min_k \frac{p_1}{p(O_k)} \leq \alpha(Z). \tag{6}$$

Therefore, by (4), together with (5) and (6), we have

$$p'_{\leq k}(Z) \geq (1-\delta)\,\alpha(Z)p'(O'_k) \geq \sqrt{1-\varepsilon}\cdot\alpha(Z)p'(O'_k),$$

which means that Z is $\sqrt{1-\varepsilon}\cdot\alpha(Z)$-robust with respect to I'. □

Lemma 5.3 implies the following lemma, which, together with Lemma 5.2, completes the proof of Theorem 5.1.

Lemma 5.6. *Scheme* Approx *returns a* $(1-\varepsilon)\alpha(I)$*-robust knapsack solution.*

Proof. Scheme Approx returns a max-robust solution X with respect to the rounded instance I'. Let Z be a max-robust solution with respect to I. Then Lemma 5.3 (b) implies that $\sqrt{1-\varepsilon}\cdot\alpha(Z) \leq \alpha'(Z) \leq \alpha'(X)$ by the maximum robustness of X. Therefore, it follows from Lemma 5.3 (a) that

$$\alpha(X) \geq \sqrt{1-\varepsilon}\cdot\alpha'(X) \geq (1-\varepsilon)\alpha(Z).$$

Thus X is $(1-\varepsilon)\alpha(I)$-robust. □

References

[1] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger, Approximation algorithms for knapsack problems with cardinality constraints, *European Journal of Operational Research*, 123, pp. 333–345, 2000.

[2] G. B. Dantzig, Discrete-variable extremum problems, *Operations Research*, 5 Issue 2, pp. 266–277, 1957.

[3] J. Edmonds, Matroids and the greedy algorithm, *Mathematical Programming*, 1, pp. 127–136, 1971.

[4] R. Fujita, Y. Kobayashi, and K. Makino, Robust matchings and matroid intersections, Lecture Notes in Computer Science 6347 (ESA 2010), Springer-Verlag, pp. 123–134, 2010.

[5] T. Fukunaga, M. Halldórsson, and H. Nagamochi, Robust cost colorings, *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*, pp. 1204–1212, 2008.

[6] M. R. Garey and D. E. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[7] R. Hassin and S. Rubinstein, Robust matchings, *SIAM Journal on Discrete Mathematics*, 15, pp. 530–537, 2002.

[8] R. Hassin and D. Segev, Robust subgraphs for trees and paths, *ACM Transaction on Algorithms*, 2, pp. 263–281, 2006.

[9] O. H. Ibarra and C. E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, *Journal of the ACM*, 22, No. 4, pp. 463–468, 1975.

[10] T. A. Jenkyns, The efficacy of the "greedy" algorithm, *Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory and Computing*, pp. 341–350, 1976.

[11] N. Kakimura and K. Makino, Robust independence systems, Lecture Notes in Computer Science 6755 (ICALP 2011), pp. 367–378, 2011. See also Mathematical Engineering Technical Reports METR 2011-14, University of Tokyo, 2011.

[12] N. Kakimura, K. Makino, and K. Seimi, Computing Knapsack Solutions with Cardinality Robustness, *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011)*, Lecture Notes in Computer Science, to appear, 2011.

[13] H. Kellerer and U. Pferschy, A new fully polynomial approximation scheme for the knapsack problem, *Journal of Combinatorial Optimization*, 3, pp. 59–71, 1999.

[14] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*, Springer-Verlag, Berlin, 2004.

[15] B. Korte and D. Hausmann, An analysis of the greedy heuristic for independence systems, *Annals of Discrete Mathematics*, 2, pp. 65–74, 1978.

[16] E.G. Lawler, Fast approximation algorithms for knapsack problems, *Mathematics of Operations Research*, 4, pp. 339–356, 1979.

[17] G. Lin, C. Nagarajan, R. Rajarama, and D. Williamson, A general approach for incremental approximation and hierarchical clustering, *SIAM Journal on Computing*, 39, pp. 3633–3669, 2010.

[18] M.J. Magazine and O. Oguz, A fully polynomial approximation algorithm for the $0-1$ knapsack problem, *European Journal of Operational Research*, 8, pp. 270–273, 1981.

[19] R. R. Mettu and C. G. Plaxton, The online median problem, *SIAM Journal on Computing*, 32, pp. 816–832, 2003.

[20] D. Pisinger and P. Toth, Knapsack Problems, in Handbook of Combinatorial Optimization, Kluwer, Norwell, pp. 1–89, 1998.

[21] R. Rado, Note on independence relations, *Proceedings of the London Mathematical Society*, 7, pp. 300–320, 1957.