Online Removable Knapsack Problem for Integer-Sized Unweighted Items**

Hiroshi FUJIWARA†, Kanaho HANJI‡, Nonmembers, and Hiroaki YAMAMOTO†, Member

SUMMARY In the online removable knapsack problem, a sequence of items, each labeled with its value and its size, is given one by one. At each arrival of an item, a player has to decide whether to put it into a knapsack or to discard it. The player is also allowed to discard some of the items that are already in the knapsack. The objective is to maximize the total value of the knapsack. Iwama and Taketomi gave an optimal algorithm for the case where the value of each item is equal to its size. In this paper we consider a case with an additional constraint that the capacity of the knapsack is a positive integer \( N \) and that the sizes of items are all integral. For each positive integer \( N \), we design an algorithm and prove its optimality. It is revealed that the competitive ratio is not monotonic with respect to \( N \).

**key words:** online algorithm, knapsack problem, competitive analysis.

1. Introduction

In the knapsack problem, a player receives a set of items, each with a value and a size, and packs some of the items into a knapsack so that the total value is maximized and the total size does not exceed the knapsack capacity [1]. Iwama and Taketomi first proposed an online version called the online removable knapsack problem [2], which has the following additional settings: (i) The player receives items one by one. Each time an item arrives, the player has to decide whether to put it into the knapsack or to discard it. The player does not know when the sequence of items ends in advance. (ii) When an item arrives, the player can discard some of the items that are already in the knapsack.

Throughout this paper, we focus on deterministic algorithms that given a particular sequence of items, always behave the same way. The competitive ratio is often used as a performance measure of algorithms for online problems [3]. For an algorithm \( ALG \), let \( ALG(\sigma) \) be the total value of items in the knapsack that is obtained by running \( ALG \) for a sequence of items \( \sigma \). Let \( OPT(\sigma) \) be the maximum value of items that are in \( \sigma \) and can be contained in the knapsack.

The competitive ratio of \( ALG \) is defined as

\[ R_{ALG} = \sup_{\sigma} \frac{OPT(\sigma)}{ALG(\sigma)}. \]

Intuitively, \( R_{ALG} \) tells how badly \( ALG \) may pack in terms of value, compared with an ideal knapsack that is packed with referring to the entire sequence. Thus, a smaller competitive ratio implies a better performance.

Based on this, the competitive ratio of the problem indicates the intrinsic difficulty of the problem, defined as

\[ R = \inf_{ALG} R_{ALG}. \]

If \( R \) is small, then the problem is “easy” in the sense that the problem can have a good algorithm.

The paper [2] studied the online removable knapsack problem for arbitrary-sized unweighted items, where an unweighted item is such that its value is equal to its size. The size of each item is a positive real number at most the knapsack capacity. The authors identified the value of \( R \) as Theorem 1 below. Hereinafter, the constant \( \Phi \) equals \( \sqrt{5}+1 \approx 0.618 \), which is called the golden ratio conjugate [4] and is the reciprocal of the golden ratio \( \varphi = \frac{\sqrt{5}+1}{2} \approx 1.618 \).

**Theorem 1.** ([2]) For the online removable knapsack problem for arbitrary-sized unweighted items, it holds that

\[ R = \frac{1}{\Phi}. \]

Moreover, an algorithm \( ALG \) with competitive ratio \( R_{ALG} = \frac{1}{\Phi} \) exists and it is optimal.

In this paper, we impose an additional setting that the capacity of the knapsack is a positive integer \( N \) and that the size of each item is also an integer between 1 and \( N \). In a sense, we are going to tackle a narrowed problem that deals with a finite set of sizes.

It should be noted that calculations in computers are mostly done over a finite set of rational numbers, which is isomorphic to the set of integers between 1 and \( N \) with some \( N \). A tuned algorithm based on the value of \( N \) may offer a better performance there.

One may think that the algorithm of Theorem 1 with some rounding technique will just work and therefore our problem is trivial. However, as we will illustrate in Section 4.2, a straightforward rounding fails to obtain an optimal performance. Our study warns that designing an optimal algorithm for a discretized problem sometimes requires...
a somewhat careful analysis.

1.1 Our Contribution

Below is our main theorem.

**Theorem 2.** For the online removable knapsack problem for integer-sized unweighted items, it holds that

\[
R = \max \left\{ \frac{N}{[N\Phi] + 1}, \min \left\{ \frac{N}{[N\Phi]}, \frac{N}{N - [N\Phi] + 1} \right\} \right\},
\]

where \( N \) is the capacity of the knapsack.

That is to say, we have an algorithm with competitive ratio being the above value and it is optimal. Theorem 2 follows from two lemmas which appear in Sections 3 and 4: Lemma 1 on a lower bound and Lemma 2 on an upper bound. The lower bound is derived in almost the same way as in the paper [2]. To establish the upper bound, we modify the algorithm of the paper [2] by regarding the classification of item sizes as a parameter. Our main work is to construct a parameter that provides a tight bound for each \( N \).

See Table 1 and Figure 1 for the value of \( R \) for each \( N \). It is observed that \( R \) roughly increases as \( N \) grows, and then the problem becomes more “difficult”. This would be intuitive. However, \( R \) does not increase monotonically as \( N \) grows. In fact, \( R = \frac{1}{3} \) for \( N = 4 \) but \( R = \frac{5}{8} < \frac{1}{2} \) for \( N = 5 \).

From Theorem 1 we obtain that \( R \leq \frac{6}{5} \) for every \( N \). Besides, Theorem 2 together with Lemma 7, which will be shown in Section 4.4, implies the corollary below.

**Corollary 1.** For the online removable knapsack problem for integer-sized unweighted items, \( \lim_{N \to \infty} R = \frac{1}{3} \) holds, where \( N \) is the capacity of the knapsack.

1.2 Related Works

The knapsack problem is one of fundamental combinatorial optimization problems [1]. The task is to choose a subset of items so that the total value is maximized and the total size does not exceed the knapsack capacity. To consider an online version of optimization problems is a fascinating interest of researchers [3], [5], [6].

We already said that Iwama and Taketomi first proposed the online removable knapsack problem and proved Theorem 1 [2]. In the rest of this section, the size of an item is always assumed to be arbitrary. Han, Kawase, and Makino studied randomized algorithms for the online removable knapsack problem for unweighted items [7]. The authors proved a lower bound of \( \frac{2}{3} \) and an upper bound of \( \frac{10}{7} \) on the competitive ratio of the problem. In their other paper [8], they considered an extended version of the online removable knapsack problem for unweighted items in which the player has to pay some cost when discarding an item.

The weighted case that the value of an item is independent of its size has been studied as well. Han et al. studied the online removable knapsack problem in which the value of an item is a convex function of its size [9]. Han, Chen, and Makino analyzed the case of a concave function [10]. In addition, it was proved that if each item can have arbitrary value, then the competitive ratio of the problem is unbounded [2].

2. Preliminaries

Let \( \mathbb{Z} \) be the set of all integers. The operator \( \times \) always represents the multiplication of two scalars. We have already defined the constant \( \Phi := \sqrt{\Phi} - 1 \). In the later analysis, we will often use the equality \( \Phi^2 + \Phi - 1 = 0 \) and the inequalities 0.6180 < \( \Phi < 0.6181 \) < 0.6666 < \( \frac{5}{3} \).

2.1 Problem Statement

Our problem is the online removable knapsack problem for integer-sized unweighted items, stated as follows. A player receives a sequence of items one by one. When item \( u \) arrives, the player has to decide (i) whether to put \( u \) into the knapsack or to discard \( u \), and (ii) a subset of items in the knapsack to be discarded, so that the total size of items in the knapsack does not exceed the capacity of the knapsack.

We denote the size of item \( u \) by \( |u| \).

In our problem, for every item, its value is equal to its size. We refer to such an item as an unweighted item. Thus, the value of item \( u \) is also \( |u| \). We do not define any notation for the value of an item.

An input, which is a sequence of items, is displayed as a sequence of the sizes (= values) of items, such as \( \sigma = (|u_1|, |u_2|, |u_3|, \ldots) \). The set of items in the knapsack is denoted by \( B \), and the total value (= total size) of items in the knapsack is denoted by \( |B| \).

Another important setting in our problem is that for every item, its size is a positive integer. The capacity of the knapsack is parameterized by a positive integer \( N \). Then, we can assume that \( 1 \leq |u| \leq N \) for every item \( u \).

2.2 Evaluation of Algorithms

Let \( ALG \) be a deterministic online algorithm of the player. That is to say, \( ALG \) receives items one by one and is not informed of the next item before processing the current item. Besides, \( ALG \) must make every decision with a probability of one. We denote by \( ALG(\sigma) \) the total value (= total size) of items in the knapsack that is obtained by running \( ALG \) for an input \( \sigma \). Define \( OPT(\sigma) \) to be the maximum value (= maximum size) of items that are in \( \sigma \) and can be contained in the knapsack.

The competitive ratio of an algorithm is commonly used as a performance measure [3]. The competitive ratio of \( ALG \) is defined as

\[
R_{ALG} = \sup_{\sigma} \frac{OPT(\sigma)}{ALG(\sigma)},
\]

which is understood as a worst-case evaluation compared
with the total value (= total size) of an ideal knapsack that is packed with referring the entire sequence. Thus, a smaller competitive ratio means that the online algorithm performs better.

Based on this, the competitive ratio of the problem serves as a measure of the intrinsic difficulty of the problem, which is defined as the competitive ratio of an optimal deterministic online algorithm:

\[ R = \inf_{ALG} R_{ALG}. \]

This indicates how good algorithm can be designed for the problem. From this viewpoint, we can explain that the smaller \( R \) is, the “easier” the problem is.

The main target of our research is to identify the value of \( R \). Since our problem is parameterized by the capacity of the knapsack \( N \), so is the value of \( R \).

3. Lower Bound

Lemma 1 gives a lower bound on the competitive ratio of our problem. The idea of the proof is due to the paper [2], while the proof in [2] uses two input sequences, we use four input sequences in our proof.

**Lemma 1.** For each positive integer \( N \), it holds that

\[ R \geq \max \left\{ \frac{N}{\lceil N\Phi \rceil + 1}, \min \left\{ \frac{N}{\lceil N\Phi \rceil}, \frac{N}{N - \lceil N\Phi \rceil + 1} \right\} \right\}. \]

**Proof.** (I) For \( N \geq 3 \), the lemma is proved by the two inequalities obtained from (I-i) and (I-ii) below.

(I-i) Consider the following two input sequences

\[ \sigma_1 = ([N\Phi] + 1, N - \lceil N\Phi \rceil) \]

and \( \sigma_2 = ([N\Phi] + 1, N - \lceil N\Phi \rceil, [N\Phi]). \]

The knapsack of capacity \( N \) cannot contain the two items in \( \sigma_1 \) at the same time. Then, the set of all deterministic online algorithms can be partitioned by behavior for \( \sigma_1 \) into the following three sets: the whole set of algorithms that packs the item of size \( \lceil N\Phi \rceil + 1 \), denoted by \( A_1 \); the whole set of algorithms that packs the item of size \( N - \lceil N\Phi \rceil \), denoted by \( A_2 \); and the whole set of algorithms that packs nothing, denoted by \( A_3 \).

If \( ALG \in A_1 \), then \( ALG(\sigma_2) \leq \lceil N\Phi \rceil + 1 \). This is because the behavior for \( \sigma_1 \) and that for the first two items in \( \sigma_2 \) is the same. On the other hand, \( OPT(\sigma_1) = N \).

If \( ALG \in A_2 \), then \( ALG(\sigma_1) = N - \lceil N\Phi \rceil \). On the other hand, \( OPT(\sigma_1) = \lceil N\Phi \rceil + 1 \).

If \( ALG \in A_3 \), then \( ALG(\sigma_1) = 0 \), which implies that the competitive ratio \( \sup_{\sigma} OPT(\sigma) \) is unbounded.

From these observations, we obtain:

\[ R = \inf_{ALG} R_{ALG} \]

\[ = \inf_{ALG} \sup_{\sigma} OPT(\sigma) \]

\[ = \min \left\{ \inf_{ALG \in A_1} \sup_{\sigma} OPT(\sigma), \inf_{ALG \in A_2} \sup_{\sigma} OPT(\sigma) \right\} \]

\[ \geq \min \left\{ \inf_{ALG \in A_1} OPT(\sigma_2), \inf_{ALG \in A_2} OPT(\sigma_1) \right\} \]

\[ \geq \min \left\{ \frac{N}{\lceil N\Phi \rceil + 1}, \frac{N}{N - \lceil N\Phi \rceil + 1} \right\} \]

\[ = \frac{N}{\lceil N\Phi \rceil + 1}. \]

The reason why the latter operand of the minimum operator vanishes is that: The equality

\[ \frac{\lceil N\Phi \rceil + 1}{N - \lceil N\Phi \rceil} \frac{N}{\lceil N\Phi \rceil + 1} \]

\[ = \frac{\lceil N\Phi \rceil^2 + (N + 2)\lceil N\Phi \rceil - N^2 + 1}{(\lceil N\Phi \rceil + 1)(N - \lceil N\Phi \rceil + 1)} \] (1)
holds, the denominator of (1) is positive since
\[
N - \lfloor N \Phi \rfloor > N - (N \Phi + 1) \\
= N(1 - \Phi) - 1 \\
= 3(1 - 0.6181) - 1 \\
= 0.1457 > 0,
\]
and the numerator of (1) is also positive since
\[
\lfloor N \Phi \rfloor^2 + (N + 2)\lfloor N \Phi \rfloor - N^2 + 1 \\
= (N \Phi)^2 + (N + 2)N \Phi - N^2 + 1 \\
= N^2(\Phi^2 + \Phi - 1) + 2N \Phi + 1 \\
= 2N \Phi + 1 > 0.
\]

(I-i) Consider the following two input sequences
\[
\sigma_3 = ([N \Phi], N - \lfloor N \Phi \rfloor + 1)
\]
and \(\sigma_4 = ([N \Phi], N - \lfloor N \Phi \rfloor + 1, \lfloor N \Phi \rfloor - 1)\).

The knapsack of capacity \(N\) cannot contain the two items in \(\sigma_3\) at the same time. Then, similarly to (I-i), the set of all deterministic online algorithms can be partitioned by behavior for \(\sigma_3\) into the following three sets: the whole set of algorithms that packs the item of size \(\lfloor N \Phi \rfloor\), denoted by \(A_4\); the whole set of algorithms that packs the item of size \(N - \lfloor N \Phi \rfloor + 1\), denoted by \(A_5\); and the whole set of algorithms that packs nothing, denoted by \(A_6\).

Observing as the same as (I-i), we have:
\[
R = \min \left\{ \inf_{\text{ALG} \in A_4} \sup_{\sigma} \frac{\text{OPT}(\sigma)}{\text{ALG}(\sigma)}, \inf_{\text{ALG} \in A_5} \sup_{\sigma} \frac{\text{OPT}(\sigma)}{\text{ALG}(\sigma)} \right\} \\
\geq \min \left\{ \inf_{\text{ALG} \in A_4} \sup_{\sigma} \frac{\text{OPT}(\sigma_4)}{\text{ALG}(\sigma_4)}, \inf_{\text{ALG} \in A_5} \sup_{\sigma} \frac{\text{OPT}(\sigma_5)}{\text{ALG}(\sigma_5)} \right\} \\
\geq \min \left\{ \frac{N}{\lfloor N \Phi \rfloor}, \frac{N}{N - \lfloor N \Phi \rfloor + 1} \right\}.
\]

From (I-i) and (I-ii), we conclude that \(R \geq \max \left\{ \frac{N}{\lfloor N \Phi \rfloor}, \min \left\{ \frac{N}{\lfloor N \Phi \rfloor}, \frac{N}{N - \lfloor N \Phi \rfloor + 1} \right\} \right\}\) holds from the definition of the competitive ratio. It is easy to see that \(\max \left\{ \frac{N}{\lfloor N \Phi \rfloor}, \min \left\{ \frac{N}{\lfloor N \Phi \rfloor}, \frac{N}{N - \lfloor N \Phi \rfloor + 1} \right\} \right\} = 1\) for \(N = 1\) and \(N = 2\).

Note. This kind of lemma is often proved in such a way that the adversary gives a “bad” input for the algorithm, referring to its behavior, and the worst ratio is evaluated. See the paper [2] for example. Unlike this way, we here give a proof based on the classification of all algorithms.

4. Upper Bound

We provide a deterministic online algorithm whose competitive ratio is at most the lower bound value of Lemma 1, for each \(N\), by slightly modifying the deterministic online algorithm of the Iwama and Taketomi’s paper [2]. This leads us to that the bound of Lemma 1 is tight and the modified algorithm is optimal. Formally, we give the lemma below, which follows from two lemmas which we will prove later: Lemma 4 for \(N \leq 4\) and Lemma 13 for \(N \geq 5\).

Lemma 2. For each positive integer \(N\), it holds that
\[
R \leq \min \left\{ \frac{N}{\lfloor N \Phi \rfloor + 1}, \min \left\{ \frac{N}{\lfloor N \Phi \rfloor}, \frac{\lfloor N \Phi \rfloor}{N - \lfloor N \Phi \rfloor + 1} \right\} \right\}.
\]

4.1 Iwama and Taketomi’s Algorithm Revisited

For the online removable knapsack problem, the deterministic online algorithm of the Iwama and Taketomi’s paper [2] runs for arbitrary-sized unweighted items, and achieves the competitive ratio of \(\frac{1}{2}\) in Theorem 1. During the execution, the algorithm classifies each given item into four classes \(S\), \(M\), \(L\), and \(X\) by size, and decides what to do for it.

We adapt the algorithm of the paper [2] to our problem, that is to say, the problem in which the capacity of the knapsack is a positive integer \(N\) and the size of each item is an integer between 1 and \(N\). The modified algorithm is given as Algorithm IT (see Algorithm 1).

Algorithm IT takes a tuple of four classes \((S, M, L, X)\) as a parameter. Each class is either an interval of integers that represent item sizes, or the empty set, denoted by \(\emptyset\). For example, if class \(S\) is not empty, we represent \(S\) as \([S_{\text{min}}, S_{\text{max}}]\), where \(S_{\text{min}}\) and \(S_{\text{max}}\) stand for the minimum and maximum values of the interval, respectively. We regard an interval of \([a, b]\) with \(a > b\) as the empty set, though this is abuse of notation. An item whose size is in class \(S\), for example, will be referred to as an \(S\) item.

We denote \(S < M\), for example, if \(S < m\) holds for all \(s \in S\) and \(m \in M\). Note that if \(S\) is the empty set, \(S < M\) holds for any \(M\).

The next lemma gives a sufficient condition for the competitive ratio of Algorithm IT to be at most a target, which can be originally found in the proof of Theorem 1 in [2]. The intuition can be explained as follows. On each “foreach” iteration, Algorithm IT gets item \(u\) and processes it along any of the marks (a), (b), . . . , (f-ii) in the comments. In any case, the condition of the lemma ensures that \(\frac{\text{OPT}(\sigma)}{\text{ALG}(\sigma)}\) is at most the target if \(u\) is the last item of input \(\sigma\).

Lemma 3 ([2]). Suppose that a tuple of classes \((S, M, L, X)\) and a real number \(c\) are such that \(M\), \(L\), and \(X\) are non-empty, \(S < M < L < X\) holds, and the following inequalities (2), (3), (4), and (5) are satisfied. Then, \(R_{IT} \leq c\) holds.

\[
M_{\text{max}} + L_{\text{max}} \leq N, \tag{2}
\]
\[
X_{\text{min}} \geq \frac{N}{c}, \tag{3}
\]
\[
M_{\text{min}} + L_{\text{min}} \geq X_{\text{min}}, \tag{4}
\]
\[
and \ L_{\text{min}} \geq \frac{L_{\text{max}}}{c}. \tag{5}
\]
Algorithm 1: IT, the algorithm of [2] with modification for integer-sized items. The claims in the comments are valid when \((S, M, L, X)\) satisfies the condition of Lemma 3.

Input: A sequence of items \(\sigma\)

Output: A knapsack \(B\) of capacity \(N\)

Parameter: A tuple of classes \((S, M, L, X)\)

\begin{verbatim}
1. \(B \leftarrow 0;\)
2. foreach item \(u \in \sigma\) do
   1. if \(|B| \in X\) then
      discard \(u;\) // (a)
   2. else if \(|u| + |B| \leq N\) then
      \(B \leftarrow B \cup \{u\};\) // (b)
   3. else \(// u \in L, or u \in X\)
      \(B \leftarrow \{u\};\) // (c)
   4. if \(B\) includes only \(S\) items and \(M\) items then
      while \(|u| + |B| > N\) do
         \(B \leftarrow B \setminus \{\text{some single item in } B\}\)
      end
   5. else \(// B\) must contain a single \(L\) item and possibly \(S\) items
         \(l \leftarrow \text{the single } L\text{ item in } B;\)
      if \(|u| + |l| \leq N\) then
         \(B \leftarrow B \cup \{u\};\) // (e)
      else if \(|u| < |l|\) then
         \(B \leftarrow \{B \setminus \{l\}\} \cup \{u\};\) // (f-i)
      else
         discard \(u;\) // (f-ii)
      end
   end
end
\end{verbatim}

4.2 A Naive Extension by Rounding Fails

All we have to do is to find a “good” tuple of classes \((S, M, L, X)\) for Algorithm IT. For items of arbitrary size in the interval \((0,1]\) and a knapsack of capacity 1, the algorithm of [2] originally uses

\((S, M, L, X) = ((0,2\Phi - 1],
(2\Phi - 1, 1 - \Phi],
(1 - \Phi, \Phi],
(\Phi, 1)).\)

By multiplying the bounds by \(N\) and rounding them, we can have a naive extension to our problem:

\((S, M, L, X) = ([1, \lceil N(2\Phi - 1) \rceil - 1],
[\lceil N(2\Phi - 1) \rceil, \lceil N(1 - \Phi) \rceil - 1],
[\lceil N(1 - \Phi) \rceil, \lceil N\Phi \rceil - 1],
\lceil N\Phi \rceil, N)).\)

It may seem that if we employ this as a parameter for Algorithm IT, then the algorithm works optimally and achieves a tight bound. However, this attempt fails.

Indeed, consider the case of \(N = 40\) and an input sequence of \(\sigma = (25,20,20)\) for example. The tuple of classes by the naive extension is \((S, M, L, X) = ([1, 9], [10, 15], [16, 24], [25, 40]).\) Run Algorithm IT using this. For \(\sigma\), the algorithm iterates as: (b), (a), and (a). Then, we obtain \(IT(\sigma) = 25.\) Obviously, \(OPT(\sigma) = 20 + 20 = 40.\) This implies that \(R_{IT} \geq \frac{20}{25} = 1.6.\) On the other hand, the lower bound of Lemma 1 for \(N = 40\) is \(\frac{25}{16} = 1.5625,\) which means that a tight bound cannot be established in this way.

The above observation tells us that providing a tight bound is a bit more involved. In Sections 4.3 and 4.4, we are going to construct a tuple of classes that depends on the lower bound value of Lemma 1 for each \(N.\) We will find that for \(N = 40\), we should apply \((S, M, L, X) = ([1, 9], [10, 15], [16, 25], [26, 40])\) instead.

4.3 Upper Bound For \(N \leq 4\)

For each of \(N \leq 4,\) we provide a tuple of classes for Algorithm IT individually. Note that the competitive ratios in Lemma 4 coincide with the lower bound values of Lemma 1, which are calculated as 1, 1, 1, and \(\frac{3}{2}\) for \(N = 1, 2, 3,\) and 4, respectively.

Lemma 4. (I) For \(N = 1,\) set \((S, M, L, X) = (0, 0, 0, 0, 1, 1)).\) Then, \(R_{IT} = 1\) holds. (II) For \(N = 2,\) set \((S, M, L, X) = (0, 0, 1, 1, 2, 2).\) Then, \(R_{IT} = 1\) holds. (III) For \(N = 3,\) set \((S, M, L, X) = (0, 1, 1, 2, 2, 3, 3)).\) Then, \(R_{IT} = 1\) holds. (IV) For \(N = 4,\) set \((S, M, L, X) = (0, 1, 1, 2, 2, 3, 4)).\) Then, \(R_{IT} \leq \frac{5}{4}\) holds.

Proof. (I) For \(N = 1,\) given the tuple of classes, only either (a) or (b) occurs in the last iteration of Algorithm IT, since any item in \(X.\) Trivially, \(\frac{OPT(\sigma)}{TT(\sigma)} = \frac{1}{1} = 1\) holds. (II) For \(N = 2,\) given the tuple of classes, only either (a), (b), or (c) occurs at every iteration of Algorithm IT. This fact is confirmed as follows. Assume that (d) or the later part is executed. Then, \(|u| = 1\) must hold. Look back to the beginning of the “foreach” block. If \(|B| \geq 2,\) then (a) occurs. Otherwise, item \(u\) can be put into the knapsack and (b) occurs. Anyway, the result contradicts the assumption.

See the behavior of Algorithm IT for \(\sigma.\) (i) When \(\sigma\) consists of a single item of size 1. Then, (b) is executed and \(\frac{OPT(\sigma)}{TT(\sigma)} = \frac{1}{1} = 1\) holds. (ii) Otherwise. The total size of \(\sigma\) is at least 2. As long as (a), (b), or (c) is repeated, the total size of items in the knapsack does not decrease. Thus, \(\frac{OPT(\sigma)}{TT(\sigma)} \leq \frac{2}{3}\) holds.

(III) and (IV) are shown by Lemma 3. For both cases, it is clear that \(M, L,\) and \(X\) are non-empty and that \(S < M < L < X\) holds. For \(N = 3,\) take \(c = 1.\) Then, the inequalities hold true as follows: \(M_{max} + L_{max} - N = 1 + 2 - 3 = 0,\)
\(X_{min} - \frac{2}{c} = 3 - 3 = 0, M_{min} + L_{min} - X_{min} = 1 + 2 - 3 = 0,\)
and \(L_{min} - L_{max} = 2 - 2 = 0.\) For \(N = 4,\) take \(c = \frac{4}{3}\).
Then, we have \(M_{max} + L_{max} - N = 1 + 2 - 4 = -1 < 0,\)
\(X_{min} - \frac{N}{c} = 3 - \frac{4}{3} = 0, M_{min} + L_{min} - X_{min} = 1 + 2 - 3 = 0,\)
\(L_{min} - L_{max} = 2 - 2 = 0.\) For \(N = 4,\) take \(c = \frac{4}{3} = 1.33\).
and \( L_{\text{min}} - \frac{L_{\text{max}}}{e} = 2 - \frac{2}{3} = \frac{4}{3} > 0. \)

\section*{4.4 Upper Bound for \( N \geq 5 \)}

We construct a tuple of classes for Algorithm IT that satisfies the condition of Lemma 3 for each \( N \geq 5 \). To this end, we define the reciprocal of the lower bound value as \( \alpha_N \). More precisely, we define as:

\[
\alpha_N := \min \left\{ \frac{[N\Phi] + 1}{N}, \max \left\{ \frac{[N\Phi]}{N}, \frac{N - [N\Phi] + 1}{[N\Phi]} \right\} \right\}.
\]

We further define a partition of the whole set of integers \( \geq 5 \) into the following three sets:

\[
E_1 := \{ N \in \mathbb{Z} \mid \frac{[N\Phi] + 1}{N} \leq \frac{N - [N\Phi] + 1}{[N\Phi]}, N \geq 5 \}
= \{ 8, 16, 21, 24, 29, 37, 42, \ldots \},
\]

\[
E_2 := \{ N \in \mathbb{Z} \mid \frac{[N\Phi]}{N} < \frac{N - [N\Phi] + 1}{N}, N \geq 5 \}
= \{ 6, 11, 14, 19, 22, 27, 32, \ldots \},
\]

\[
E_3 := \{ N \in \mathbb{Z} \mid \frac{N - [N\Phi] + 1}{[N\Phi]} \leq \frac{[N\Phi]}{N}, N \geq 5 \}
= \{ 5, 7, 9, 10, 12, 13, 15, \ldots \}.
\]

By using these, \( \alpha_N \) is rewritten as below:

\[
\alpha_N = \begin{cases} 
\frac{[N\Phi] + 1}{N}, & \text{if } N \in E_1; \\
\frac{N - [N\Phi] + 1}{[N\Phi]}, & \text{if } N \in E_2; \\
\frac{[N\Phi]}{N}, & \text{if } N \in E_3.
\end{cases}
\]

We begin with some fundamental lemmas on \( \alpha_N \).

\textbf{Lemma 5.} For \( N \in E_1 \), \([N\alpha_N] = [N\Phi] + 1 = N\alpha_N \) holds and \( N\alpha_N \) is integral. For \( N \in E_2 \), \([N\alpha_N] = [N\Phi] + 1 > N\alpha_N \) holds and \( N\alpha_N \) is not integral. For \( N \in E_3 \), \([N\alpha_N] = [N\Phi] = N\alpha_N \) holds and \( N\alpha_N \) is integral.

\textit{Proof.} (I) For \( N \in E_1 \), we know \( \alpha_N = \frac{[N\Phi] + 1}{N} \). Thus, \( N\alpha_N = [N\Phi] + 1 \), which implies that \( N\alpha_N \) is integral. As a result, \([N\alpha_N] = N\alpha_N \) holds. (II) For \( N \in E_2 \), it holds that \( \frac{[N\Phi]}{N} < \alpha_N < \frac{[N\Phi] + 1}{N} \) by definition of \( E_2 \). Thus, \( [N\Phi] < N\alpha_N < [N\Phi] + 1 \), which states that \( N\alpha_N \) cannot be an integer. We then obtain \([N\alpha_N] = [N\Phi] + 1 \). (III) For \( N \in E_3 \), we know \( \alpha_N = \frac{[N\Phi]}{N} \). Thus, \( N\alpha_N = [N\Phi] \), which implies that \( N\alpha_N \) is integral. Consequently, \([N\alpha_N] = N\alpha_N \) holds.

\textbf{Lemma 6.} For each integer \( N \geq 5 \), \( \Phi < \alpha_N < 1 \) holds.

\textit{Proof.} (A) Proof of \( \Phi < \alpha_N \): For \( N \in E_1 \), we derive as:

\[
\alpha_N = \frac{[N\Phi] + 1}{N} > \frac{N\Phi + 1}{N} > \frac{N\Phi}{N} = \Phi.
\]

For \( N \in E_2 \), it holds that \( \alpha_N = \frac{N - [N\Phi] + 1}{N} > \frac{[N\Phi]}{N} > \frac{N\Phi}{N} = \Phi \) by definition of \( E_2 \). For \( N \in E_3 \), we have \( \alpha_N = \frac{[N\Phi]}{N} > \frac{N\Phi}{N} = \Phi \).

(B) Proof of \( \alpha_N < 1 \): By definition of \( \alpha_N \), it holds that \( \alpha_N \leq \frac{[N\Phi] + 1}{N} \). Substituting 5 for \( N \), we obtain \( \frac{[N\Phi] + 1}{5} \leq \frac{3}{5} \) for all \( N \geq 5 \). For \( N \geq 6 \), we evaluate as:

\[
\frac{[N\Phi] + 1}{N} < \frac{N\Phi + 2}{N} < \frac{N\Phi + 2}{N} < \frac{N\Phi + 2}{N} < 0.9515 < 1.
\]

\textbf{Lemma 7.} \( \lim_{N \to \infty} \alpha_N = \Phi \).

\textit{Proof.} By definition of \( \alpha_N \), we derive that \( \alpha_N \leq \frac{[N\Phi] + 1}{N} \) for all \( N \geq 5 \). We compute \( \lim_{N \to \infty} \frac{N\Phi + 2}{N} = \Phi \). On the other hand, by Lemma 6, \( \Phi < \alpha_N \) holds for all \( N \geq 5 \). Hence, by the squeeze theorem, we conclude that \( \lim_{N \to \infty} \alpha_N = \Phi \).

\textbf{Lemma 8.} For each integer \( N \geq 5 \), it holds that

\[
2N - 3[N\alpha_N] + 3 \geq 0.
\]

The equality sign holds only for \( N = 6 \).

\textit{Proof.} (I) For \( N \in E_1 \cup E_2 \) \( = \{ 6, 8, 11, 14, 16, 19, 21, 22, \ldots \} \), by Lemma 5, we have

\[
2N - 3[N\alpha_N] + 3 = 2N - 3[N\Phi].
\]

By simple substitution, we obtain: For \( N = 6, 2N - 3[N\Phi] = 0 \), which is the only case when the equality holds. For \( N = 8, 11, \) and \( 14, 2N - 3[N\Phi] = 1 > 0 \). Also, for \( N = 16 \) and \( 19, 2N - 3[N\Phi] = 2 > 0 \). Besides, for \( N \geq 21 \), we show that

\[
2N - 3[N\Phi] \geq 2N - 3(N\Phi + 1) + 3 \geq 3(N\Phi + 1) + 3 \geq 3\left( \frac{2}{3} - \Phi \right) > 0.
\]

(II) For \( N \in E_3 \), applying Lemma 5, we derive that

\[
2N - 3[N\alpha_N] + 3 = 2N - 3[N\Phi] + 3 > 2N - 3(N\Phi + 1) + 3 \geq 3(N\Phi + 1) + 3 \geq 3\left( \frac{2}{3} - \Phi \right) > 0.
\]

\textbf{Lemma 9.} For each integer \( N \geq 5 \), it holds that

\[
2[N\alpha_N] - N - 3 \geq 0.
\]

The equality sign holds only for \( N = 5, 7, \) and \( 9 \).

\textit{Proof.} (I) For \( N \in E_1 \cup E_2 \), applying Lemma 5, we show that

\[
2[N\alpha_N] - N - 3 = 2([N\Phi] + 1) - N - 3 > 2(N\Phi + 1) - N - 3 = N(2\Phi - 1) - 1 \geq 5 \times (2 \times 0.6180 - 1) - 1.
\]
For other \( N \), before checking \( \overline{S} < \overline{M} \), we should confirm that \( \overline{S} \) is non-empty. Noting the condition of the equality sign in Lemma 9 and the fact that \( |N\alpha_N| \) is integral, we have \( 2|N\alpha_N| - N - 4 \geq 0 \). Hence, \( 1 \leq 2|N\alpha_N| - N - 3 \). Therefore, \( \overline{S} \) is non-empty.

Then, we can declare \( M_{\text{min}} = \overline{S}_{\text{max}} + 1 \), which implies that \( \overline{S} < \overline{M} \). \( \square \)

Lemma 11 is a helper lemma for showing \((\overline{S}, \overline{M}, \overline{L}, \overline{X})\) to satisfy the condition of Lemma 3.

**Lemma 11.** For each integer \( N \geq 5 \), it holds that

\[
N - (1 + \alpha_N)|N\alpha_N| + \alpha_N + 2 \geq 0.
\]

**Proof.** (I) For \( N \in E_1 \), the definition of \( E_1 \) and Lemma 5 imply that \( \alpha_N = \frac{[N\Phi] + 1}{N} \leq \frac{N - [N\Phi] + 1}{N} = \frac{N - N\alpha_N + 2}{N\alpha_N - 1} \). By Lemma 6, we have \( N\alpha_N - 1 > 5\Phi - 1 > 5 \times 0.618 - 1 = 2.090 > 0 \). From these inequalities, we get

\[
(N\alpha_N - 1)\alpha_N \leq N - N\alpha_N + 2. \tag{6}
\]

Using Lemma 5 and (6), we derive that

\[
N - (1 + \alpha_N)|N\alpha_N| + \alpha_N + 2 = N - N\alpha_N - (N\alpha_N - 1)\alpha_N + 2 \\
\geq N - N\alpha_N - (N - N\alpha_N + 2) + 2 = 0.
\]

(II) For \( N \in E_2 \), by Lemma 5, we have

\[
N - (1 + \alpha_N)|N\alpha_N| + \alpha_N + 2 = N - (1 + \alpha_N)(|N\Phi| + 1) + \alpha_N + 2 \\
= N - [N\Phi] - \alpha_N[N\Phi] + 1 \\
= N - [N\Phi] - (N - [N\Phi] + 1) + 1 = 0.
\]

(III) For \( N \in E_3 \), by Lemma 5, we derive that

\[
N - (1 + \alpha_N)|N\alpha_N| + \alpha_N + 2 = N - (1 + \frac{[N\Phi]}{N})|N\Phi| + \frac{[N\Phi]}{N} + 2 \\
= N - \frac{[N\Phi]^2}{N} - \frac{N - 1}{N} \times [N\Phi] + 2 \\
> N - \frac{(N\Phi + 1)^2}{N} - \frac{N - 1}{N} \times (N\Phi + 1) + 2 \\
= N(-\Phi^2 - \Phi + 1) - \Phi + 1 \\
= -\Phi + 1 \\
> 0.3819 > 0.
\]

\( \square \)

**Lemma 12.** For each integer \( N \geq 5 \), \((\overline{S}, \overline{M}, \overline{L}, \overline{X})\) and \( \overline{c} = \frac{1}{\alpha_N} \) satisfy the condition of Lemma 3.

**Proof.** Lemma 10 guarantees that \( \overline{M}, \overline{L}, \) and \( \overline{X} \) are non-empty, and \( \overline{S} < \overline{M} < \overline{L} < \overline{X} \). What remains is to check the inequalities in the condition of Lemma 3. It is easy to see
inequalities (2) and (3) hold for $(\bar{S}, \bar{M}, \bar{L}, \bar{X})$ and $\bar{c} = \frac{1}{\alpha_N}$:
\[
N - \bar{M}_{\text{max}} - \bar{L}_{\text{max}} = N - (N - \lfloor N\alpha_N \rfloor + 1) - (\lfloor N\alpha_N \rfloor - 1) = 0
\]
and
\[
\bar{X}_{\text{min}} - \frac{N}{\bar{c}} = \lfloor N\alpha_N \rfloor - N\alpha_N \geq 0.
\]
By Lemma 11, we have
\[
\bar{L}_{\text{min}} - \frac{\bar{L}_{\text{max}}}{\bar{c}} = (N - \lfloor N\alpha_N \rfloor + 2) - \alpha_N (\lfloor N\alpha_N \rfloor - 1) = N - (1 + \alpha_N)\lfloor N\alpha_N \rfloor + \alpha_N + 2 \geq 0,
\]
which is inequality (5). The confirmation of inequality (4) is done depending on the value of $N$. For $N = 5, 7,$ and 9, applying Lemma 9, we have
\[
\bar{M}_{\text{min}} + \bar{L}_{\text{min}} - \bar{X}_{\text{min}} = 1 + (N - \lfloor N\alpha_N \rfloor + 2) - \lfloor N\alpha_N \rfloor = N - 2\lfloor N\alpha_N \rfloor + 3 = 0.
\]
For other $N$,
\[
\bar{M}_{\text{min}} + \bar{L}_{\text{min}} - \bar{X}_{\text{min}} = (2\lfloor N\alpha_N \rfloor - N - 2) + (N - \lfloor N\alpha_N \rfloor + 2) - \lfloor N\alpha_N \rfloor = 0.
\]
Lemmas 3 and 12 immediately establish an upper bound.

Lemma 13. For each integer $N \geq 5$, set $(S, M, L, X) = (\bar{S}, \bar{M}, \bar{L}, \bar{X})$. Then, $R_{IT} \leq \frac{1}{\alpha_N}$ holds.

5. Concluding Remarks

Our result reveals that the competitive ratio of our problem does not increase monotonically as the capacity of the knapsack grows. This anomaly may be somehow related to the validity of the competitive ratio.

In Sections 4.3 and 4.4, the tuple of classes that achieves a tight bound was given in an individual form for each $N \leq 9$. We wonder if the tuple can be represented in a unified form for every positive integer $N$.

A natural extension of our work is to consider the case of integer-sized weighted items, in which the size of each item is an integer but its value is not always equal to the size.

References