Stealing from the Dragon's Hoard

Online Unbounded Knapsack With Removal

Matthias Gehnen
Department of Computer Science, RWTH Aachen University, Germany

Moritz Stocker[1]
Department of Computer Science, ETH Zurich, Switzerland

Abstract

We introduce the Online Unbounded Knapsack Problem with Removal, a variation of the well-known Online Knapsack Problem. Items, each with a weight and value, arrive online and an algorithm must decide on whether or not to pack them into a knapsack with a fixed weight limit. An item may be packed an arbitrary number of times and items may be removed from the knapsack at any time without cost. The goal is to maximize the total value of items packed, while respecting a weight limit. We show that this is one of the very few natural online knapsack variants that allow for competitive deterministic algorithms in the general setting, by providing an algorithm with competitivity 1.6911. We complement this with a lower bound of 1.5877.

We also analyze the proportional setting, where the weight and value of any single item agree, and show that deterministic algorithms can be exactly 3/2-competitive. Lastly, we give lower and upper bounds of 6/5 and 4/3 on the competitivity of randomized algorithms in this setting.

2012 ACM Subject Classification Theory of computation → Adversary models

Keywords and phrases online problems, online knapsack, unbounded knapsack, removal

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.43

1 Introduction

Imagine, if you will, a hobbit creeping through a dragon's lair. Having found the dragon, it now swiftly makes its escape. Surrounding it on its way out are piles of coins, heaps of rubies, mountains of diamonds, each containing far more than the hobbit could possibly carry. Grinning, the hobbit opens its trusty knapsack and fills it to the brim with silver coins. Around a corner, it almost stumbles over a number of large gold bars. It empties its knapsack of silver and packs two of these gold bars instead, which completely fill its knapsack. Around the next bend, gleaming in the light of the hobbit's candle, lies an enormous jewel. Greedily, the hobbit removes the two gold bars and packs the jewel instead. It silently curses, wishing it had kept the silver coins, some of which would have nicely filled up the remaining space in the knapsack. But it dares not go back, deeper into the lair, towards the dormant dragon. And thus, the hobbit shoulders its knapsack and presses on...

In the Online Knapsack Problem, an algorithm receives an unknown number of items one by one, each with a weight and a value. It must decide whether or not to pack these items into a knapsack, without any knowledge about future items that may be offered. Its goal is to maximize the combined value of the items it packs without exceeding the knapsack's weight capacity, often normalized to 1.

[1] Corresponding Author

43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).
Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thắng
Article No. 43; pp. 43:1–43:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Ever since the problem was introduced by Marchetti-Spaccamela and Vercellis [14], it has offered fertile ground for analysis. Though the original problem is not competitive, even for the Proportional Knapsack variant where the weight and value of any single item are equal, there are countless variants of the problem and aspects under which to consider them that prove interesting – and for which well-performing algorithms exist. A recent survey detailing many of these results has been compiled by Böckenhauer et al. [4].

One of the simplest competitive variants is the Online Knapsack Problem with Removal. In this model, items may be removed from the knapsack at any time without cost. It was first studied for the proportional case by Iwama and Taketomi [10], who showed that the best competitive ratio that can be achieved by a deterministic algorithm is exactly the golden ratio $\varphi \approx 1.618$. Han et al. [7] considered randomized algorithms and provided an algorithm with competitivity $10/7$. They also gave a lower bound of $5/4$ on the competitive ratio of any randomized algorithm, which has recently been improved by Hächler [6] to approximately 1.2695. The general case, where items have arbitrary weights and values remains non-competitive for deterministic algorithms even with removal, as shown by Iwama and Zhang [11]. However, Han et al. [7] showed that a competitivity of 2 can in fact be achieved by randomized algorithms, which they complemented by a lower bound of $1 + 1/\mathrm{e}$.

More recently, Böckenhauer et al. [3] considered the Online Unbounded Knapsack Problem. This can be seen as the "industrial" setting of the knapsack problem, where each item is available in unlimited quantities. In this model, rather than only being able to decide on whether to pack an item or not, an algorithm may choose to pack it any number of times. The classical model, where each item can only be packed once or not at all is correspondingly sometimes referred to as the 0-1 Knapsack Problem. More correctly, the Unbounded Knapsack Problem should not be viewed as a model that provides additional resources to an algorithm, but as one that limits the possible *instances* considered. An instance of the Unbounded Knapsack Problem is equivalent to one of the 0-1 Knapsack Problem where any item of weight w is guaranteed to consecutively appear at least $\lfloor 1/w \rfloor$ times. Böckenhauer et al. showed that deterministic algorithms can be exactly 2-competitive in the proportional case, and gave upper and lower bounds of $1 + \ln(2) \approx 1.6931$ and 1.7353, respectively, for randomized algorithms. They also showed that in the general setting, no finite competitive ratio is possible even for randomized algorithms.

Our Contributions

In this paper, we combine the Online Unbounded Knapsack Problem and the Online Knapsack Problem with Removal to the Online Unbounded Knapsack Problem with Removal. This can be interpreted as an application of removal, being one of the most natural variants of the knapsack problem, to the industrial setting of the Unbounded Knapsack, resulting in a situation where an algorithm is not penalized for initially being forced to pack worthless items.

More whimsically, it can be described by a hobbit stealing from the riches of a dragon's hoard. The online nature of the problem is captured by the fact that the hobbit's first priority is to move away from the dragon, while it tries to maximize the value of items it takes along. It is natural to assume that the hobbit would be able to remove items from its knapsack. Given that a dragon's hoard is commonly associated with untold riches in vast quantities, the unbounded setting is also a reasonable choice.

In this model, we show that the best possible competitive ratio for deterministic algorithms in the proportional variant is exactly $3/2$ (Lemmas 2 and 3). For randomized algorithms, we offer a simple lower bound of $6/5$ on the competitive ratio (Lemma 4) and a slightly

Table 1 Results contained in this paper (left column), as compared to previous results for ONLINE UNBOUNDED KNAPSACK and ONLINE KNAPSACK WITH REMOVAL. These results cover deterministic and randomized algorithms for the proportional case, as well as deterministic algorithms for the general case. Randomized algorithms for the general problem are not studied separately in this paper. The lower bound follows from the proportional case, while the upper bound follows from the deterministic case.

	UNBOUNDED WITH REMOVAL		UNBOUNDED [3]		WITH REMOVAL	
	lower bound	upper bound	l.b.	u.b.	l.b.	u.b.
prop. det.	3/2 (Lemma 3)	3/2 (Lemma 2)	2		1.618 [10]	
prop. rand.	6/5 (Lemma 4)	4/3 (Thm. 5)	1.6931	1.7353	1.2695 [6]	10/7 [7]
general det.	1.5877 (Thm. 11)	1.6911 (Thm. 9)	∞		∞ [11]	
general rand.	6/5 (Lemma 4)	1.6911 (Thm. 9)	∞		1.3679 [7]	2 [7]

more complex randomized algorithm with competitivity 4/3 (Theorem 5). These results are summarized in Table 1. Since this model is obviously no harder than the ONLINE UNBOUNDED KNAPSACK PROBLEM (lending additional capabilities to the algorithm) and the ONLINE KNAPSACK PROBLEM WITH REMOVAL (essentially limiting the set of possible instances), it is not surprising that all our bounds are lower than the best-known bounds in those models.

Our main results concern the GENERAL KNAPSACK PROBLEM, however. Interestingly, the ONLINE UNBOUNDED KNAPSACK PROBLEM WITH REMOVAL is one of the few natural knapsack variants where deterministic algorithms can be competitive without further assumptions.[2] Neither the ONLINE KNAPSACK PROBLEM WITH REMOVAL alone [11] nor the ONLINE UNBOUNDED KNAPSACK PROBLEM (without removal) [3] allow for this behavior. However, we show that in the combined model of the ONLINE UNBOUNDED KNAPSACK PROBLEM WITH REMOVAL, a simple algorithm can achieve a competitivity of 1.6911 (Theorem 9). We complement this with a lower bound of 1.5877 (Theorem 11).

Preliminaries

An instance of the ONLINE UNBOUNDED KNAPSACK PROBLEM WITH REMOVAL consists of a sequence of items $I = (x_1, \ldots, x_n)$. An item is given as $x = (w, v)$, where $w = w(x)$ is its *weight* and $v = v(x)$ is its *value*; we define the item's *density* as $\rho(x) = v(x)/w(x)$.

A solution is given as a multiset S of items, such that $x \in I$ for any $x \in S$, satisfying

$$\sum_{x \in S} w(x) \leq 1\,.$$

The goal is to maximize the *gain*

$$v(S) = \sum_{x \in S} v(x)\,.$$

We denote by $\textsc{Opt}(I)$ an optimal solution to I, that is, a solution that maximizes $v(\textsc{Opt}(I))$.

[2] The few other problems that allow for this include ONLINE KNAPSACK WITH REMOVAL with resource augmentation [11] and resource buffer [8], as well as ONLINE MULTIPLE KNAPSACK WITH REMOVAL [5], all of which tinker with the size of the knapsack and are arguably less simple than the variant we consider.

An online algorithm ALG maintains a solution (or "knapsack") S. When it receives an item x, it may add (or "pack") any number of copies of x to S, then remove an arbitrary subset of items from S, such that S is again a valid solution. We denote by $\textsc{Alg}(I)$ the solution of ALG on the instance I.

A deterministic algorithm ALG is said to be (strictly) c-competitive for some constant $c \geq 1$ if $c \cdot v(\textsc{Alg}(I)) \geq v(\textsc{Opt}(I))$ for any instance I. Equivalently, a randomized algorithm ALG is said to be (strictly) c-competitive in expectation if $c \cdot \mathbb{E}[v(\textsc{Alg}(I))] \geq v(\textsc{Opt}(I))$ for any instance I. In either case, the *(strict) competitive ratio* or *competitivity* of ALG is defined as

$$\inf_{c\geq 1}\{c \mid \textsc{Alg} \text{ is (strictly) } c\text{-competitive (in expectation)}\}\,.$$

There is also a non-strict version of competitivity: an algorithm is said to be (non-strictly) c-competitive if there exists a constant $\alpha \geq 0$ such that $c \cdot v(\textsc{Alg}(I)) + \alpha \geq v(\textsc{Opt}(I))$ for any instance I. Throughout this paper, we only consider the strict competitive ratio and reserve the term *competitive ratio* for this version. It should be noted, however, that the lower bound for general item values of Theorem 11 also holds for the non-strict version, which can easily be seen by scaling the values of all items. For any upper bound, proving a strict competitive ratio obviously implies a corresponding non-strict ratio.

For an item $x = (w, v)$, the maximal number of copies of x that fit into the knapsack at the same time is exactly $\lfloor 1/w(x)\rfloor$. We define the *cumulative value* $v^*(x) = v(x) \cdot \lfloor 1/w(x)\rfloor$ as the combined value of these copies, that is, the maximal gain that can be achieved using only copies of x. Similarly, we define the *cumulative weight* of x as $w(x) \cdot \lfloor 1/w(x)\rfloor$. We denote by $\mathrm{fill}(x)$ the corresponding multiset consisting of $\lfloor 1/w(x)\rfloor$ copies of x.

We finally state a simple result that will prove useful throughout the paper:

▶ **Lemma 1.** *If* $0 < x \leq 1/n$*, for* $n \in \mathbb{N}$ *and* $n \geq 1$*, then*

$$x \cdot \lfloor 1/x\rfloor = \frac{\lfloor 1/x\rfloor}{1/x} \geq \frac{n}{n+1}\,.$$

Proof. Let $n' = \lfloor 1/x\rfloor$, so $1/x \leq n' + 1$. Since $1/x \geq n$, clearly $n' \geq n$, and therefore

$$\frac{\lfloor 1/x\rfloor}{1/x} = \frac{n'}{1/x} \geq \frac{n'}{n'+1} \geq \frac{n}{n+1}\,.$$ ◀

2 The Proportional Knapsack Problem

In this section, we consider the PROPORTIONAL KNAPSACK PROBLEM, where the weight and value of any item are equal. For convenience, we identify an item x with its weight and value, that is, we say that $x = w(x) = v(x)$.

2.1 Deterministic Algorithms

We start by considering a very simple algorithm:

The algorithm SIMPLE keeps the heaviest item in the instance, as long as there are no items of weight less than or equal to $1/2$. If there is an item of weight smaller than or equal to $1/2$ in the instance, SIMPLE removes any previous item in the knapsack, packs as many copies of this small item as possible and ignores all further items.

The algorithm is shown in pseudocode as Algorithm 1.

Algorithm 1 The deterministic algorithm Simple for the proportional knapsack problem.

```
K ← ∅
for x ∈ I do
    if x ≤ 1/2 then
        K ← fill(x)
        return K
    else if K = ∅ then
        K ← {x}
    else if K = {x′} and x > x′ then
        K ← {x}
return K
```

▶ **Lemma 2.** *The competitive ratio of* Simple *is at most* $3/2$*.*

Proof. Consider any instance I. If there are no items of weight less than or equal to $1/2$ in I, no two items in the instance fit in the knapsack together. Since Simple keeps the heaviest item, it is optimal.

Otherwise, let x be the first item in I of weight $x \leq 1/2$. By definition, Simple achieves a gain of $x \cdot \lfloor 1/x \rfloor$, which is at least $2/3$ by Lemma 1. Since the optimal solution has value at most 1, Simple therefore has a competitive ratio of at most $3/2$. ◀

While, as its name indicates, Simple uses a straightforward strategy, we can show that no deterministic online algorithm can improve on its competitive ratio.

▶ **Lemma 3.** *No deterministic algorithm for the* Proportional Online Unbounded Knapsack Problem with Removal *can have a competitive ratio of less than* $3/2$*.*

Proof. Let Alg be any online algorithm and let $\varepsilon > 0$ be sufficiently small. Consider the two instances

$$I_1 = (1/3 + 2\varepsilon, 2/3 - \varepsilon, 1/3 + \varepsilon)\,,$$
$$I_2 = (1/3 + 2\varepsilon, 2/3 - \varepsilon, 2/3 - 2\varepsilon)\,.$$

Since I_1 and I_2 agree on the first two items, Alg must act identically on that prefix. If it packs the second item $2/3 - \varepsilon$, it cannot keep any copies of $1/3 + 2\varepsilon$. In this case, it has a gain of at most $2/3 - \varepsilon$ on I_2, while by packing one copy of $1/3 + 2\varepsilon$ and the last item, a gain of 1 would have been possible.

If, on the other hand, it does not pack the item $2/3 - \varepsilon$, it cannot pack more than two copies of $1/3 + 2\varepsilon$ and $1/3 + \varepsilon$ in total. Therefore, its gain on I_1 is at most $2/3 + 4\varepsilon$, while again, a gain of 1 is possible by packing the second item and one copy of the last item. In either case, its competitive ratio is at least $1/(2/3 + 4\varepsilon)$. The statement of the lemma follows as $\varepsilon \to 0$. ◀

2.2 Randomized Algorithms

We now consider randomized algorithms for the Proportional Online Unbounded Knapsack Problem with Removal. We start with a simple lower bound using Yao's principle [19]:

▶ **Fact** (Yao's Principle). *Consider any online maximization problem, where $v(S)$ denotes the value of a solution S. Let $\mathcal{I}$ be a set of instances, and let* $\mathrm{Prob}_{\mathcal{I}}$ *denote any probability distribution on $\mathcal{I}$, with $\mathbb{E}_{\mathcal{I}}$ being the corresponding expectation. If, for any deterministic algorithm* Alg*, it holds that*

$$c \cdot \mathbb{E}_{\mathcal{I}}[v(\textsc{Alg})] \leq \mathbb{E}_{\mathcal{I}}[v(\textsc{Opt})]\,,$$

for some constant $c \geq 1$, then no randomized algorithm for the problem can have a competitive ratio of less than c.

A more detailed explanation of Yao's principle can be found, for example, in the textbook by Komm [12].

▶ **Lemma 4.** *No randomized algorithm for the* Proportional Online Unbounded Knapsack Problem with Removal *can have a competitive ratio of less than* $6/5$.

Proof. The proof uses the same instances as in Lemma 3, that is,

$$I_1 = (1/3 + 2\varepsilon, 2/3 - \varepsilon, 1/3 + \varepsilon)\,,$$
$$I_2 = (1/3 + 2\varepsilon, 2/3 - \varepsilon, 2/3 - 2\varepsilon)\,,$$

for some sufficiently small $\varepsilon > 0$. Let Alg be any deterministic algorithm that receives I_1 or I_2 each with probability $1/2$. Then, as in the proof of Lemma 3, if Alg packs the item $2/3 - \varepsilon$, its gain is at most $2/3 - \varepsilon$ on I_2 and at most 1 on I_1. If it does not pack the item $2/3 - \varepsilon$, its gain is at most $2/3 + 4\varepsilon$ on I_1 and at most 1 on I_2. In either case, its expected gain is at most $1/2 \cdot (2/3 + 4\varepsilon + 1) = 5/6 + 2\varepsilon$, while the optimal solution is always 1. Therefore, by Yao's principle, no randomized algorithm can have a competitive ratio of less than $(5/6 + 2\varepsilon)^{-1}$. The statement of the lemma follows as $\varepsilon \to 0$. ◀

We now present a randomized algorithm for the Proportional Online Unbounded Knapsack Problem with Removal that uses a single random bit and has a competitive ratio of at most $4/3$.

The algorithm RandChoice divides items into four different weight categories. *Good* items $G = [0, 1/3] \cup [3/8, 1/2] \cup [3/4, 1]$, *small* items $S =]1/3, 3/8[$, *medium* items $M =]1/2, 5/8[$, and *large* items $L = [5/8, 3/4[$. It chooses uniformly at random between two deterministic strategies A_1 and A_2. In both strategies, if it encounters an item in G, it removes any items in the knapsack, packs the new item as often as possible and ignores all further items. Otherwise the corresponding strategies are as follows:

- $\boldsymbol{A_1}$**:** RandChoice initially keeps copies of a single item, in the following order of priority:
  1. Two copies of the smallest item in S encountered so far,
  2. The largest item in M encountered so far (if none in S are present),
  3. The smallest item in L encountered so far (if none in $S \cup M$ are present).

  To do so, it removes any previous items in the knapsack if necessary.

  If at any point it encounters an item from S that fits with an item in $M \cup L$ that is already in the knapsack or vice versa, it packs one copy of each. From then on, it replaces the item in S with any smaller item from S if possible, and the item from $M \cup L$ with a larger item from $M \cup L$ if it can do so without removing the item from S.

- A_2: RandChoice initially keeps copies of a single item, in the following order of priority:
  1. The smallest item in L encountered so far,
  2. Two copies of the smallest item in S encountered so far (if none in L are present),
  3. The largest item in M encountered so far (if none in $S \cup L$ are present).

  To do so, it removes any previous items in the knapsack if necessary.

  If at any point it encounters an item from S that fits with an item in L (but not M) that is already in the knapsack or vice versa, it packs one copy of both and ignores all further items in $S \cup M \cup L$.

The algorithm is shown in pseudocode as Algorithm 2.

▶ **Theorem 5.** RandChoice *achieves a competitive ratio of at most* $4/3$.

Proof. Let I be any non-empty instance and let $\text{Opt} := \text{Opt}(I)$ be a fixed optimal solution of value $v(\text{Opt})$. We first consider what happens when there is an item from G in the instance. Let $g \in G$ be the first such item. In that case, RandChoice achieves a gain of $g \cdot \lfloor 1/g \rfloor$. If $g \in [0, 1/3]$, this is at least $3/4$ by Lemma 1. If $g \in [3/8, 1/2]$, we know that $g \cdot \lfloor 1/g \rfloor = 2g \geq 3/4$. If $g \in [3/4, 1]$, RandChoice also achieves a gain of at least $3/4$. In any case, its competitive ratio is at most $4/3$. We can therefore assume that I only contains items in $S \cup M \cup L$. Additionally, if I contains only items from M, then RandChoice is clearly optimal since both strategies keep the largest such item in this case. We therefore also assume that I contains an item from $S \cup L$.

If $v(\text{Opt}) \leq 3/4$, then RandChoice is easily seen to be 4/3-competitive: Strategy A_1 always achieves a gain of at least $1/2$ and, since the instance contains an item from $S \cup L$, Strategy A_2 achieves a gain of at least $5/8$ (any item from S has a cumulative value of at least $2/3 > 5/8$). The expected gain of RandChoice is therefore at least $1/2 \cdot (1/2 + 5/8) = 9/16$, and its competitive ratio in expectation is at most $(3/4)/(9/16) = 4/3$. We can therefore assume that $v(\text{Opt}) > 3/4$, which means that Opt contains one copy of an item $s \in S$ and one copy of an item $b \in M \cup L$. We will now distinguish three different cases:

- **Case 1 ($b \in M$):** Since there is an item from $S \cup L$ in the instance, as above, Strategy A_2 will achieve a gain of at least $5/8 \geq 5/8 \cdot v(\text{Opt})$. Since any item from S fits with any item from M, and since it prioritizes items from $S \cup M$ over items from L, Strategy A_1 will pack both an item from S and an item from $M \cup L$. More specifically, since it first prioritizes larger items from M and then replaces the item from $M \cup L$ with larger items if possible, it packs an item from S and an item of size at least b, leading to a gain of at least $1/3 + b$. On the other hand, $v(\text{Opt}) \leq 3/8 + b$. Therefore, Strategy A_1 achieves a gain of at least $(1/3 + b)/(3/8 + b) \cdot v(\text{Opt})$. Since $(1/3 + b)/(3/8 + b)$ increases with b, the gain of Strategy A_1 is at least $(1/3 + 1/2)/(3/8 + 1/2) \cdot v(\text{Opt}) = 20/21 \cdot v(\text{Opt})$. Combining Strategies A_1 and A_2, the algorithm RandChoice has an expected gain of $1/2 \cdot (5/8 + 20/21) \cdot v(\text{Opt}) = 265/336 \cdot v(\text{Opt}) > 3/4 \cdot v(\text{Opt})$.
- **Case 2 ($b \in L$ and s arrives before b):** As in the previous case, Strategy A_2 achieves a gain of at least $5/8$. By definition of Strategy A_1, it will have packed an item from S of size at most s when b arrives, possibly together with an item from $M \cup L$. This means that it could pack b together with that item, and since it replaces the item from $M \cup L$ with a larger item if possible, it has a gain of at least $1/3 + b \geq 1/3 + 5/8 = 23/24$. This means that RandChoice has an expected gain of at least $1/2 \cdot (5/8 + 23/24) = 19/24 > 3/4$.

- **Case 3 ($b \in L$ and b arrives before s):** Since the instance contains an item from S, Strategy A_1 achieves a gain of at least $2/3$. Strategy A_2 on the other hand is always able to pack both an item from S and an item from L: if it still has only one item packed when s arrives, that item must be an item from L of size at most b. Therefore, Strategy A_2 achieves a gain of at least $1/3 + 5/8 = 23/24$ and RandChoice achieves an expected gain of at least $1/2 \cdot (2/3 + 23/24) = 13/16 > 3/4$.

In any case, RandChoice has an expected gain of at least $3/4 \cdot v(\text{Opt})$ and thus a competitive ratio in expectation of at most $4/3$. ◀

It may seem that the competitive ratio of RandChoice can easily be improved by not choosing the two strategies uniformly. The competitive ratio of either strategy is however bounded from above by $4/3$ by the assumptions made before the case distinction and especially by the way they deal with items from G. For example, if the first item Simple receives has weight $3/4$, it will pack it and ignore all future items, while the optimal solution might well be 1.

Though it may be possible to improve the algorithm, this would thus require a more fine-grained treatment of the items in G.

3 The General Knapsack Problem

We now consider the general case of the problem, where the value of an item may be unrelated to its weight. We first show that there are in fact competitive deterministic algorithms for the General Online Unbounded Knapsack Problem with Removal, which may come as a surprise, given that most variants of the General Online Knapsack Problem remain non-competitive [11], many of them even for randomized algorithms [2,3]. To this end, consider an algorithm Focus.

The algorithm Focus only keeps copies of a single item of maximal cumulative value. If an item of higher cumulative value arrives, it removes all items in the knapsack and packs as many copies of this new item as possible.

The algorithm is shown in pseudocode as Algorithm 3. Note that if Focus is used in an instance of the Proportional Online Knapsack Problem, its gain is at least that of the algorithm Simple from Section 2.1, which also only keeps copies of a single item. Therefore, Focus also achieves the best possible competitive ratio of $3/2$ in the proportional case.

To analyze the competitivity of Focus in the general case, we define a series of numbers a_n for $n \geq 1$ recursively by

$$a_1 = 2, \quad a_n = 1 + \prod_{i=1}^{n-1} a_i \quad \text{for } n \geq 2\,. \tag{1}$$

The following properties of this sequence[3] can be easily verified by induction:

▶ **Lemma 6.** *For any $n \geq 1$,*

(i) $a_{n+1} = a_n \cdot (a_n - 1) + 1\,,$

(ii) $\dfrac{1}{a_n - 1} = 1 - \displaystyle\sum_{j=1}^{n-1} \frac{1}{a_j}\,.$

⌟

[3] The sequence, whose first elements are $2, 3, 7, 43, 1807, \ldots$, is known as *Sylvester's sequence* (sequence A000058 in the OEIS [15]) after James Joseph Sylvester, who studied a family of unit fractions related to it [18].

Algorithm 2 The randomized algorithm RandChoice for the proportional knapsack problem.

```
G ← [0, 1/3] ∪ [3/8, 1/2] ∪ [3/4, 1]
S ← ]1/3, 3/8[
M ← ]1/2, 5/8[
L ← [5/8, 3/4[
K ← ∅
b ← rand({0, 1})                    ▷ Uniformly random choice between two strategies
if b = 1 then                       ▷ Deterministic strategy A_1
    for x ∈ I do
        if x ∈ G then
            K ← fill(x)
            return K
        else if K = ∅ then
            K ← fill(x)
        else if K = fill(x′) for x′ ∈ S ∪ M ∪ L then
            if x′ ∈ M ∪ L and x ∈ S or vice versa, and x + x′ ≤ 1 then
                K ← {x, x′}
            else if x ∈ S and x < x′ then
                K ← {x, x}
            else if x ∈ M and x′ ∈ M and x > x′ then
                K ← {x}
            else if x < x′ then
                K ← {x}
        else
            K = {s, b} with s ∈ S and b ∈ M ∪ L
            if x < s then
                K ← {x, b}
            else if x > b and x + s ≤ 1 then
                K ← {s, x}
else                                ▷ Deterministic strategy A_2
    for x ∈ I do
        if x ∈ G then
            K ← fill(x)
            return K
        else if K = ∅ then
            K ← fill(x)
        else if K = fill(x′) for x′ ∈ S ∪ M ∪ L then
            if x′ ∈ L and x ∈ S or vice versa, and x + x′ ≤ 1 then
                K ← {x, x′}
            else if x ∈ L and x < x′ then
                K ← {x}
            else if x ∈ S and x′ ∈ M ∪ S and x < x′ then
                K ← {x, x}
            else if x ∈ M and x′ ∈ M and x > x′ then
                K ← {x}
return K
```

Algorithm 3 The deterministic algorithm FOCUS for the general knapsack problem.

$K \leftarrow \emptyset$
for $x \in I$ **do**
  if $K = \emptyset$ **then**
    $K \leftarrow \text{fill}(x)$
  else if $K = \text{fill}(x')$ and $v^*(x) > v^*(x')$ **then**
    $K \leftarrow \text{fill}(x)$
return K

Now let $N \geq 1$. We further define

$$S_N = \sum_{n=1}^{N} (a_n - 1)^{-1}, \quad \text{with } S_0 = 0\,,$$
$$T_N = \frac{a_N}{(a_N - 1)^2} + S_{N-1}\,.$$

Lastly, we define $S_\infty = \sum_{n=1}^{\infty}(a_n - 1)^{-1}$; clearly, $S_N \to S_\infty$ as $N \to \infty$. Since the numbers a_n grow indefinitely, we also know that $a_N/(a_N - 1)^2 \to 0$, and thus also $T_N \to S_\infty$ as $N \to \infty$. Furthermore, the numbers T_N satisfy the following two properties:

▶ **Lemma 7.** *For any $N \geq 1$ and $1 \leq k \leq N$,*

(i) $\dfrac{(a_N - 1)^2}{a_N} \cdot \big(1 - S_{N-1}/T_N\big) = T_N^{-1}\,.$

(ii) $\dfrac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot \big(1 - S_{k-1}/T_N\big) > T_N^{-1}\,.$

Proof. Property (i) follows directly from the definition of T_N, since

$$T_N - S_{N-1} = \frac{a_N}{(a_N - 1)^2} \implies 1 - S_{N-1}/T_N = \frac{a_N}{(a_N - 1)^2} \cdot T_N^{-1}\,.$$

We will prove (ii) by induction on k, starting with $k = N$. We first convince ourselves that for $k = N$,

$$\frac{a_N \cdot (a_N - 1)}{a_N + 1} \cdot (1 - S_{N-1}/T_N) > \frac{(a_N - 1)^2}{a_N} \cdot (1 - S_{N-1}/T_N) = T_N^{-1}$$

by (i). Now let $1 \leq k < N$ and assume by induction that (ii) holds for $k + 1$. Then

$$\begin{aligned}
&\frac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot \big(1 - S_{k-1}/T_N\big) - T_N^{-1} \\
&= \frac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot \left(1 - S_{k-1}/T_N - \frac{a_k + 1}{a_k \cdot (a_k - 1)} \cdot T_N^{-1}\right) \\
&= \frac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot \left(1 - S_{k-1}/T_N - \frac{1}{a_k - 1} \cdot T_N^{-1} - \frac{1}{a_k \cdot (a_k - 1)} \cdot T_N^{-1}\right) \\
&= \frac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot \left(1 - \left(S_{k-1} + \frac{1}{a_k - 1}\right) \cdot T_N^{-1} - \frac{1}{a_k \cdot (a_k - 1)} \cdot T_N^{-1}\right) \\
&= \frac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot \left(1 - S_k/T_N - \frac{1}{a_k \cdot (a_k - 1)} \cdot T_N^{-1}\right) \\
&= \frac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot \big(1 - S_k/T_N\big) - \frac{1}{a_k + 1} \cdot T_N^{-1}
\end{aligned}$$

$$
\begin{aligned}
&> \frac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot \frac{a_{k+1} + 1}{a_{k+1} \cdot (a_{k+1} - 1)} \cdot T_N^{-1} - \frac{1}{a_k + 1} \cdot T_N^{-1} \quad \text{(by induction)} \\
&= T_N^{-1} \cdot \left(\frac{a_{k+1} - 1}{a_k + 1} \cdot \frac{a_{k+1} + 1}{a_{k+1} \cdot (a_{k+1} - 1)} - \frac{1}{a_k + 1} \right) \quad \text{(by Lemma 6(i))} \\
&= T_N^{-1} \cdot \left(\frac{a_{k+1} + 1}{(a_k + 1) \cdot a_{k+1}} - \frac{1}{a_k + 1} \right) \\
&= \frac{T_N^{-1}}{(a_k + 1) \cdot a_{k+1}} > 0 \,,
\end{aligned}
$$

which means that

$$
\frac{a_k \cdot (a_k - 1)}{a_k + 1} \cdot (1 - S_{k-1}/T_N) > T_N^{-1} \,.
$$

By induction, property (ii) holds for all $1 \leq k \leq N$. ◀

Before we state and prove the competitivity of Focus, we give one final preparatory lemma.

▶ **Lemma 8.** *Let A be any multiset of items in an instance with $\sum_{x \in A} w(x) \leq 1$. Let $v(A) = \sum_{x \in A} v(x)$ and let $x_d = (w_d, v_d)$ be an item of maximal density in A. Then* Focus *will achieve a gain of at least $v(A) \cdot \frac{\lfloor 1/w_d \rfloor}{1/w_d}$ on that instance.*

Proof. We know that

$$
v(A) = \sum_{x \in A} v(x) = \sum_{x \in A} \frac{v(x)}{w(x)} \cdot w(x) \leq \sum_{x \in A} \frac{v_d}{w_d} \cdot w(x) \leq \frac{v_d}{w_d} \,.
$$

The cumulative value of x_d satisfies

$$
v^*(x_d) = v_d \cdot \lfloor 1/w_d \rfloor \geq v(A) \cdot \frac{\lfloor 1/w_d \rfloor}{1/w_d} \,.
$$

Since Focus keeps an item of maximal cumulative value, it will definitely achieve such a gain. ◀

▶ **Theorem 9.** Focus *achieves a competitive ratio of at most $S_\infty < 1.69104$.*

Proof. Let I be any instance and let Opt $:=$ Opt(I) be an optimal solution on I. We first start with an intuitive idea of the proof. Ideally, an optimal solution would fill all of its space in the knapsack optimally with the densest possible items. Lemma 8 means that the higher the cumulative weight of the densest item in Opt, the better the competitive ratio of Focus. The smallest cumulative weight that any item can have is at least $1/2$, so we assume this as a worst-case scenario. By definition, the gain of Focus is at least the value of that item. Then we can consider the second-densest item in Opt. Since it must have weight smaller than $1/2$, it can be packed at least twice and thus its value can be at most $1/2$ the gain of Focus. The smallest cumulative weight such an item can have is at least $2/3$, corresponding to a weight of at least $1/3$, which we again assume as a worst-case scenario. We then continue to the third-densest item, which must have a weight of at most $1 - 1/2 - 1/3 = 1/6$, and so its value can be at most $1/6$ the gain of Focus. We can continue this process arbitrarily and see that the value of Opt is at most $1 + 1/2 + 1/6 + \cdots = S_\infty$ times the gain of Focus.

More formally, we normalize the values of all items such that $v(\text{Opt}) = 1$. Let $N \geq 1$. We show that the algorithm has a gain of at least T_N^{-1} and thus a competitive ratio of at most T_N on I. The statement of the theorem then follows since $T_N \to S_\infty$ as $N \to \infty$. We first assume that this is not the case, that is, that the algorithm has a gain of strictly less than T_N^{-1}. From this, we inductively construct a series of items $y_1 = (w_1, v_1), \dots, y_N = (w_N, v_N)$ with the following properties:

- The optimal solution OPT contains exactly one copy of each item y_k.
- The weight of y_k is bounded by $1/a_k < w_k \le 1/(a_k - 1)$ for each $1 \le k \le N$.
- The value of y_k is bounded by $v_k < (T_N \cdot (a_k - 1))^{-1}$ for each $1 \le k \le N$.

We first define $y_1 = (w_1, v_1)$ as an item of maximal density in OPT. If $w_1 \le 1/2$, then by Lemmas 1 and 8, FOCUS achieves a gain of at least $2/3$. Recall that $a_1 = 2$ and $S_0 = 0$, so the gain of FOCUS is at least

$$2/3 = \frac{a_1 \cdot (a_1 - 1)}{a_1 + 1} \cdot (1 - S_0/T_N) > T_N^{-1}$$

by Lemma 7, which we assumed was not the case. This means that $w_1 > 1/2 = 1/a_1$, which also means that the optimal solution can contain only one copy of y_1. If $v_1 \ge T_N^{-1}$, the algorithm achieves a gain of at least T_N^{-1}, so $v_1 < T_N^{-1} = (T_N \cdot (a_1 - 1))^{-1}$, which means that y_1 satisfies the properties listed above.

Now assume by induction that we have defined $y_1, \dots, y_{k-1}$ satisfying these conditions for $k \le N$. Since these items are all part of the optimal solution, the other items in OPT must have a combined value of at least $1 - \sum_{j=1}^{k-1} v_j > 1 - \sum_{j=1}^{k-1} (T_N \cdot (a_j - 1))^{-1} = 1 - S_{k-1}/T_N$. They must also have a combined weight of at most $1 - \sum_{j=1}^{k-1} w_j < 1 - \sum_{j=1}^{k-1} 1/a_j = 1/(a_k - 1)$ by Lemma 6(ii). This means that it would be possible to pack the combination of all items in $\text{OPT} \setminus \{y_1, \dots, y_{k-1}\}$ a total of $a_k - 1$ times for a gain of at least $(a_k - 1) \cdot (1 - S_{k-1}/T_N)$. Now let $y_k = (v_k, w_k)$ be an item of maximal density among these items, so clearly also $w_k < 1/(a_k - 1)$. By Lemma 8, FOCUS achieves a gain of at least $\frac{\lfloor 1/w_k \rfloor}{1/w_k} \cdot (a_k - 1) \cdot (1 - S_{k-1}/T_N)$. If $w_k \le 1/a_k$, then by Lemma 1, its gain is at least $\frac{a_k}{a_k + 1} \cdot (a_k - 1) \cdot (1 - S_{k-1}/T_N) > T_N^{-1}$ by Lemma 7, which we assumed was not the case. Therefore, $w_k > 1/a_k$, which also means that the optimal solution can contain only one copy of y_k, since it would otherwise contain items of combined weight at least

$$\left(\sum_{j=1}^{k-1} w_j \right) + 2 \cdot w_k > \left(\sum_{j=1}^{k-1} \frac{1}{a_j} \right) + 2 \cdot \frac{1}{a_k} \ge \left(\sum_{j=1}^{k-1} \frac{1}{a_j} \right) + \frac{1}{a_k - 1} = 1 \,,$$

where the last equality follows from Lemma 6(ii).

Since $w_k \le 1/(a_k - 1)$, the item y_k can be packed at least $a_k - 1$ times by itself. Therefore, if $v_k \ge (T_N \cdot (a_k - 1))^{-1}$, then FOCUS achieves a gain of at least T_N^{-1}, which we assumed was not the case, so $v_k < (T_N \cdot (a_k - 1))^{-1}$. Therefore, y_k satisfies all the properties above.

This means that we can construct $y_1, \dots, y_N$ satisfying these properties. Further note that in the case $k = N$ we showed that FOCUS achieves a gain of at least $\frac{\lfloor 1/w_N \rfloor}{1/w_N} \cdot (a_N - 1) \cdot (1 - S_{N-1}/T_N)$. By Lemma 1, this is at least

$$\frac{a_N - 1}{a_N} \cdot (a_N - 1) \cdot (1 - S_{N-1}/T_N) \,,$$

which is equal to T_N^{-1} by Lemma 7(i), once again contradicting our assumption. We are forced to conclude that this assumption was wrong. The algorithm FOCUS will therefore always achieve a gain of at least T_N^{-1} and thus a competitive ratio of at most T_N. ◀

We know show that this bound is tight, that is, FOCUS does not achieve a better competitive ratio.

▶ **Lemma 10.** *The competitive ratio of* Focus *is at least* S_∞.

Proof. Consider the instance I_N consisting of items $x_i = (w_i, v_i)$ for $1 \leq i \leq N$ with $w_i = 1/a_i + \varepsilon$ for some $0 < \varepsilon < (N \cdot (a_{N+1} - 1))^{-1}$ and $v_i = \frac{1}{a_i - 1}$. The item x_i can be packed $(a_i - 1)$ times, so its cumulative value is given by $(a_i - 1) \cdot v_i = 1$. The gain of Focus will therefore always be 1 on I_N. On the other hand, there is a solution consisting of a single copy of each of the N items, since $(\sum_{i=1}^{N} 1/a_i) + N \cdot \varepsilon = 1 - 1/(a_{N+1} - 1) + N \cdot \varepsilon < 1$ by Lemma 6(ii). This solution has a value of $S_N = \sum_{i=1}^{N} 1/(a_i - 1)$, so Focus has a competitive ratio of at least S_N on I_N. The statement of the lemma follows as $N \to \infty$. ◀

The lower bound of Lemma 10 only holds for the specific, simple algorithm Focus. We now give a lower bound for the general case that improves upon the one implied by the proportional setting (Lemma 3). The proof once again makes use of the Sylvester numbers a_n defined in Equation (1).

▶ **Theorem 11.** *Let $N \geq 3$. No deterministic algorithm for the* General Unbounded Knapsack Problem with Removal *can have a better competitive ratio than c_N, where $c_N, v_1, \ldots, v_N$ solve the following system of equations:*

$$v_N = 1$$

$$c_N = \frac{v_{i-1} + \frac{v_i}{a_i - 1}}{v_i} \quad \textit{for } 3 \leq i \leq N \tag{2}$$

$$c_N = \frac{v_2 + v_1/2}{v_1} \tag{3}$$

$$c_N = \frac{v_2 + v_1/2 + \sum_{i=3}^{N} \frac{v_i}{a_i - 1}}{v_2} \tag{4}$$

$$c_N \geq 1$$

In particular, no deterministic algorithm can have a competitive ratio better than $c_5 > 1.5877$.

Proof. Let Alg be any deterministic algorithm and let $\varepsilon > 0$ be sufficiently small. Assume by contradiction that the competitive ratio of Alg is strictly less than c_N. The following process is illustrated in Figure 1.

For $i = N, \ldots, 3$, the algorithm receives first an item $x_i = (1/a_i + 2\varepsilon, v_i/(a_i - 1))$ and then an item $y_i = (1 - 1/a_i - \varepsilon, v_{i-1})$.

Given that Alg holds only a single copy of y_{i+1} when x_i arrives (or no items in the case of $i = N$), we can assume without loss of generality that it removes any items in the knapsack and packs $a_i - 1$ copies of x_i, since that configuration of items is strictly preferable: it achieves the same gain with a smaller weight, and y_{i+1} cannot be subdivided. It cannot pack both the item y_{i+1} and a copy of x_i, since $(1 - 1/a_{i+1} - \varepsilon) + (1/a_i + 2\varepsilon) > 1$.

We now claim that whenever an item y_i arrives, Alg must remove all copies of x_i from the knapsack and pack a single copy of y_i. Otherwise, consider the first item y_i that Alg does not pack. In that case, Alg now receives an item $x'_i = (1/a_i + \varepsilon, v_i/(a_i - 1))$. A solution consisting of y_i and a single copy of x'_i would achieve a gain of $v_{i-1} + v_i/(a_i - 1)$. However, since Alg can only pack a combined total of $a_i - 1$ copies of x_i and x'_i, it has a gain of at most v_i. Therefore, it has a competitive ratio of at least $(v_i + v_{i+1}/(a_i - 1))/v_i = c_N$ by Equation (2).

This process continues until Alg has packed a single copy of the item $y_3 = (1 - 1/a_3 - \varepsilon, v_2) = (6/7 - \varepsilon, v_2)$. It then receives an item $x_2 = (1/3 + 2\varepsilon, v_1/2)$. Note that we cannot assume as before that Alg packs this item, since the values of these two items are unrelated a priori. However, if it does not pack the item x_2, ALG receives an item $z = (1/2 + \varepsilon, v_2)$. The best gain it can then achieve is v_2. Packing a single copy of each of $z, x_2, x_3, \ldots, x_N$ is possible, since $\sum_{i=1}^{N} 1/a_i < 1$ by Lemma 6(ii). This leads to an optimal gain of at least $v_2 + v_1/2 + \sum_{i=1}^{N} v_i/(a_i - 1)$, and thus to a competitive ratio of c_N by Equation (4).

If it packs the item x_2, ALG now receives an item $y_2 = (2/3 - \varepsilon, v_2)$. If it does not pack y_2, ALG receives an item $x_2' = (1/3 + \varepsilon, v_1)$. The best gain it can achieve is v_1, while packing a single copy each of x_2' and y_2 is possible, leading to a competitive ratio of at least $(v_2 + v_1/2)/v_1 = c_N$ by Equation (3). Finally, if it does pack y_2, ALG again receives an item $z = (1/2 + \varepsilon, v_2)$. As before, the best gain it can achieve is v_2, while packing a single copy each of $z, x_2, \ldots, x_N$ is possible, leading to a competitive ratio of c_N by Equation (4).

To solve the system of equations, note that Equation (2) implies that $v_{i-1} = (c_N - 1/(a_i - 1)) \cdot v_i$ for $3 \leq i \leq N$, and Equation (3) implies that $v_1 = v_2/(c_N - 1/2)$. With the additional condition of $v_N = 1$, this means that

$$v_i = \prod_{j=i+1}^{N} \left(c_N - \frac{1}{a_j - 1} \right) \quad \text{for } i \geq 2 \tag{5}$$

$$v_1 = (c_N - 1/2)^{-1} \cdot v_2 \tag{6}$$

Equation (4) then becomes

$$c_N \cdot v_2 = v_2 + \frac{v_1}{2} + \sum_{i=3}^{N} \frac{v_i}{a_i - 1} \iff (c_N - 1) \cdot v_2 \overset{(6)}{=} \frac{(c_N - 1/2)^{-1} \cdot v_2}{2} + \sum_{i=3}^{N} \frac{v_i}{a_i - 1}$$

$$\iff (c_N - 1) \cdot (c_N - 1/2) \cdot v_2 = \frac{v_2}{2} + (c_N - 1/2) \cdot \sum_{i=3}^{N} \frac{v_i}{a_i - 1} .$$

Using the expressions for v_i from Equation (5) and the fact that $1 = 1/(a_1 - 1)$ and $1/2 = 1/(a_2 - 1)$, this becomes

$$\prod_{i=1}^{N} \left(c_N - \frac{1}{a_i - 1} \right) = \frac{1}{2} \cdot \prod_{i=3}^{N} \left(c_N - \frac{1}{a_i - 1} \right) + (c_N - 1/2) \cdot \sum_{i=3}^{N} \frac{1}{a_i - 1} \prod_{j=i+1}^{N} \left(c_N - \frac{1}{a_j - 1} \right) . \tag{7}$$

This is a polynomial equation of degree N, which can be solved numerically for fixed values of N. In the special case of $N = 5$, we find that the largest positive root of Equation (7) is $c_5 > 1.5877$. ◀

4 Conclusion

We have introduced and studied the ONLINE UNBOUNDED KNAPSACK PROBLEM WITH REMOVAL. Apart from showing that the competitive ratio that can be achieved by deterministic algorithms in the proportional setting is exactly 3/2, we also gave lower and upper bounds of 6/5 and 4/3, respectively, for randomized algorithms.

We further showed that deterministic algorithms can be competitive in the general setting and provided upper and lower bounds of approximately 1.5877 and 1.6913 on their competitive ratio. Given the simple nature of the algorithm FOCUS, it does not seem unreasonable that an algorithm with a better competitive ratio exists. It might be worth considering randomized algorithms in the general setting, improving on the implied bounds shown in Table 1.

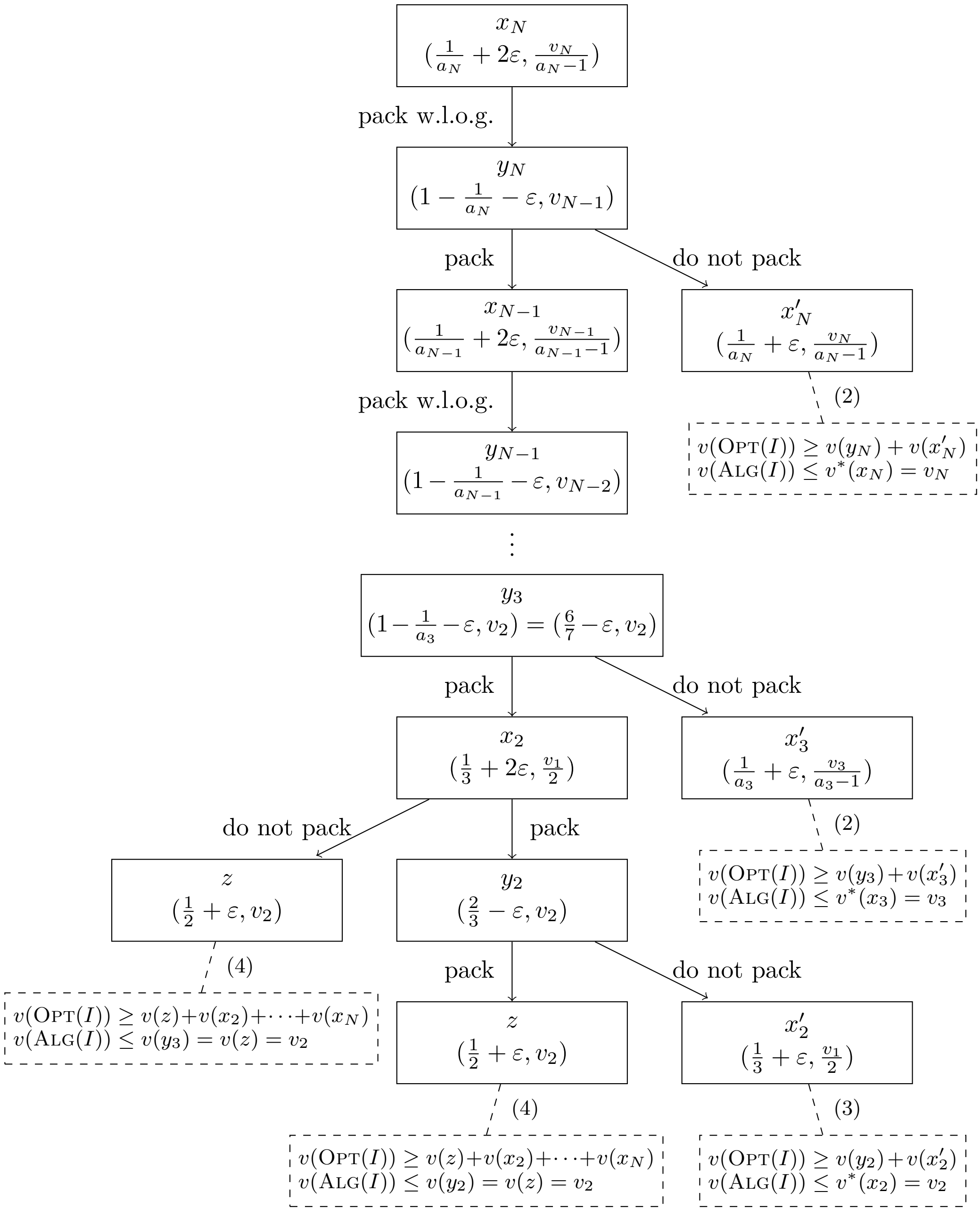

Figure 1 Illustration of the process used in the proof of Theorem 11, indicating which of the conditions (2), (3), or (4) leads to the desired competitive ratio of c_N.

A Possible Connection to Bin-Packing

Both the upper bound of Theorem 9 and the lower bound of Theorem 11 depend on the Sylvester numbers $(a_n)_{n=1}^{\infty}$ defined in Equation (1). The upper bound, which is explicitly given as $S_\infty = \sum_{n=1}^{\infty}(a_n - 1)^{-1}$, is of particular interest, since this is not the first time that S_∞ has appeared as the competitive ratio of an online algorithm for a packing problem: it was previously shown to be the (asymptotic) competitive ratio of the Harmonic algorithm for Bin-Packing [13].

Any speculation that the problems of Bin-Packing and Online Unbounded Knapsack with Removal might have the same competitivity is moot, since the best known algorithm for Bin-Packing due to Balogh et al. [1] has competitivity at most 1.5783, which beats the lower bound of 1.5877 we proved in Theorem 11. However, the Harmonic algorithm is part of a larger family, the Super-Harmonic framework introduced by Seiden [17]. The best-known lower and upper bounds for algorithms in this framework are 1.5833 [16] and 1.5884 [9], respectively, which tantalizingly frame our lower bound. It is therefore at least theoretically possible that the best competitive ratios of an algorithm in the Super-Harmonic framework and an algorithm for Online Unbounded Knapsack with Removal agree.

If there is some connection between the two problems, further research might provide deeper insight into the competitive analysis of both problems. Potentially, this could lead to reducing the gap between the upper and lower bounds of the Super-Harmonic framework, or even closing it. However, this is of course mere speculation:

(a) There is no reason to assume that the true competitive ratio of the Online Unbounded Knapsack Problem with Removal is closer to the lower bound of 1.5877 than the upper bound of 1.6911.

(b) Even if this were the case, and the competitive ratios of the two problems are close together, there is a priori no reason for them to agree.

(c) Even if the ratios happen to agree, this might not necessarily be due to some deeper connection between the two problems.

Even so, we find this numerical coincidence between the two online packing problems to be worth noting. The possibility of such a connection remains an open problem. It could be answered in the negative by even a comparatively small improvement of the lower bound of Theorem 11.

References

1 János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A New and Improved Algorithm for Online Bin Packing. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2018.5.

2 Hans-Joachim Böckenhauer, Dennis Komm, Richard Královič, and Peter Rossmanith. The online knapsack problem: Advice and randomization. *Theoretical Computer Science*, 527:61–72, 2014. doi:10.1016/j.tcs.2014.01.027.

3 Hans-Joachim Böckenhauer, Matthias Gehnen, Juraj Hromkovič, Ralf Klasing, Dennis Komm, Henri Lotze, Daniel Mock, Peter Rossmanith, and Moritz Stocker. Online unbounded knapsack. *Theory of Computing Systems*, 69(1):1–25, 2025. doi:10.1007/s00224-025-10215-0.

4 Hans-Joachim Böckenhauer, Juraj Hromkovič, Dennis Komm, Peter Rossmanith, and Moritz Stocker. A survey of online knapsack problems. *Discrete Applied Mathematics*, 378:492–507, 2026. doi:10.1016/j.dam.2025.08.011.

5 Marek Cygan, Lukasz Jez, and Jirí Sgall. Online knapsack revisited. *Theory of Computing Systems*, 58(1):153–190, 2016. doi:10.1007/s00224-014-9566-4.

6 Valentin Hächler. Untrusted predictions for knapsack problem variants. Bachelor's Thesis, ETH Zurich, 2025. doi:10.3929/ethz-c-000782659.

7 Xin Han, Yasushi Kawase, and Kazuhisa Makino. Randomized algorithms for online knapsack problems. *Theoretical Computer Science*, 562:395–405, 2015. doi:10.1016/j.tcs.2014.10.017.

8 Xin Han, Yasushi Kawase, Kazuhisa Makino, and Haruki Yokomaku. Online knapsack problems with a resource buffer. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation (ISAAC 2019)*, volume 149 of *LIPIcs*, pages 28:1–28:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.28.

9 Sandy Heydrich and Rob van Stee. Beating the harmonic lower bound for online bin packing. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.41.

10 Kazuo Iwama and Shiro Taketomi. Removable online knapsack problems. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan J. Eidenbenz, and Ricardo Conejo, editors, *29th International Colloquium on Automata, Languages and Programming (ICALP 2002)*, volume 2380 of *Lecture Notes in Computer Science*, pages 293–305. Springer, 2002. doi:10.1007/3-540-45465-9_26.

11 Kazuo Iwama and Guochuan Zhang. Online knapsack with resource augmentation. *Information Processing Letters*, 110(22):1016–1020, 2010. doi:10.1016/j.ipl.2010.08.013.

12 Dennis Komm. *An Introduction to Online Computation – Determinism, Randomization, Advice*. Springer, 2016. doi:10.1007/978-3-319-42749-2.

13 C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *J. ACM*, 32(3):562–572, 1985. doi:10.1145/3828.3833.

14 Alberto Marchetti-Spaccamela and Carlo Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995. doi:10.1007/BF01585758.

15 OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2026. Published electronically at http://oeis.org.

16 Prakash Ramanan, Donna J Brown, C.C Lee, and D.T Lee. On-line bin packing in linear time. *Journal of Algorithms*, 10(3):305–326, 1989. doi:10.1016/0196-6774(89)90031-X.

17 Steven S. Seiden. On the online bin packing problem. *J. ACM*, 49(5):640–671, 2002. doi:10.1145/585265.585269.

18 J. J. Sylvester. On a point in the theory of vulgar fractions. *American Journal of Mathematics*, 3(4):332–335, 1880. doi:10.2307/2369261.

19 Andrew C.-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)*, pages 222–227, 1977. doi:10.1109/SFCS.1977.24.