Online Simple Knapsack with Bounded Predictions

Matthias Gehnen
RWTH Aachen University, Germany

Henri Lotze[1]
RWTH Aachen University, Germany

Peter Rossmanith
RWTH Aachen University, Germany

Abstract

In the Online Simple Knapsack problem, an algorithm has to pack a knapsack of unit size as full as possible with items that arrive sequentially. The algorithm has no prior knowledge of the length or nature of the instance. Its performance is then measured against the best possible packing of all items of the same instance, over all possible instances.

In the classical model for online computation, it is well known that there exists no constant bound for the ratio between the size of an optimal packing and the size of an online algorithm's packing. A recent variation of the classical online model is that of *predictions*. In this model, an algorithm is given knowledge about the instance in advance, which is in reality distorted by some factor δ that is commonly unknown to the algorithm. The algorithm only learns about the actual nature of the elements of an input once they are revealed and an irrevocable and immediate decision has to be made. In this work, we study a slight variation of this model in which the error term, and thus the range of sizes that an announced item may actually lay in, is given to the algorithm in advance. It thus knows the range of sizes from which the actual size of each item is selected from.

We find that the analysis of the Online Simple Knapsack problem under this model is surprisingly involved. For values of $0 < \delta \leq \frac{1}{7}$, we prove a tight competitive ratio of 2. From there on, we are able to prove that there are at least three alternating functions that describe the competitive ratio. We provide partially tight bounds for the whole range of $0 < \delta < 1$, showing in particular that the function of the competitive ratio depending on δ is not continuous.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online problem, Simple Knapsack, Predictions, Machine-Learned Advice

Digital Object Identifier 10.4230/LIPIcs.STACS.2024.37

1 Introduction

The Online (Simple) Knapsack problem has received a lot of attention since it has been found to be non-competitive in the classical model for online computation by Marchetti-Spaccamela and Versellis [27].

In this model, an online algorithm receives a finite sequence of requests, one item after another. It has no knowledge of the nature of future items or the length of the sequence, and is tasked to either pack a given item into a knapsack of unit size or to reject it. Every decision is irrevocable. The objective of the algorithm is to maximize the sum of sizes of packed items in the knapsack, without this sum exceeding the knapsack size of 1. Its performance is measured against the best possible packing of the same sequence of items. The worst ratio between this optimal packing and the packing of an algorithm over all possible sequences is then called the *competitive ratio* of the algorithm, as first defined by Sleator and Tarjan [29]. For a more thorough introduction to competitive analysis, we refer to the books by Borodin and El-Yaniv [13] and by Komm [24].

[1] Corresponding author

The classical counterexample to show that even the Online Simple Knapsack problem is not competitive, is arguably pathological: An adversary chooses between two instances with an identical prefix of an arbitrarily small first item of size $\varepsilon > 0$. The first instance only contains this small first item, while a second instance contains an item of size 1 as the second item. Depending on whether a given algorithm deterministically discards or packs this first item, the first or the second instance is given to the algorithm, resulting in an unbounded competitive ratio.

Thus, many variations of the problem itself or indeed the classical online model itself have been studied, each aiming to disallow these pathological instances. We will only give an incomplete list of all studied variants. The variations of the problem itself include allowing an online algorithm to pack a slightly larger knapsack than the offline counterpart, called *resource augmentation* by Iwama and Zhang [22] and allowing an algorithm to intermediately store items in a *buffer* of a certain size, as introduced by Han et al. [18]. Thielen, Tiedemann and Westphal [31] studied a model in which the capacity of the knapsack increases step-wise over a given number of discrete time periods. Iwama and Taketomi [21] introduced a variation in which packed items could be removed (also called *preempted*) from the knapsack, not to be packed again. Building on this result, Han and Makino [19] additionally allowed for a limited number of *cuts*, i.e. splitting of items into two sub-items. Zhou, Chakrabarty and Lukose [33] analyzed the online knapsack problem under the assumption that the size of each item is much smaller than the knapsack capacity and the ratio between the value and the weight of an item is bounded within a given range $[L, U]$. Further variations include allowing for randomization and allowing for an oracle to communicate information about the instance via so-called *advice bits*. These variations were studied by Böckenhauer et al. [12]. The recently introduced model by Böckenhauer et al. [11] of *reservation costs* allows to pay a fee for a presented item in order to delay a decision on it for an arbitrary amount of time.

When acting in an online setting, the assumption that the future is completely unknown is often unrealistic. When buying boxes for moving to a different house, one usually has a pretty good idea of what one owns and can then buy a number of boxes on this estimate. When planning to drive a group of friends home, one can already pre-plan a route with the rough knowledge on where these persons actually live.

A common theme is thus that the knowledge of the world and the future is actually not unknown, on the contrary: it is often known but the details are uncertain. If we want to model this kind of behavior, we could thus assume that an instance is already given in advance: all elements are revealed beforehand. When the elements arrive, i.e. we have to actually pack the moving boxes or ask our friends where they live, we learn the *actual* nature of the element. For the sake of a simple model, let us assume that the actual number of elements is correctly given. We also assume that our estimates do not deviate beyond a certain bound, i.e., that the maximum uncertainty is bounded and this bound is known in advance.

2 Problem Definitions and Notation

We will abuse notation by using x_i for both the label of an item and for the size (or gain) of the same item.

▶ **Definition 1** (The Online Simple Knapsack with Bounded Predictions Problem). *Given a constant $\delta \in [0, 1]$, called* distortion. *Given a set of items $I = (x_1, \ldots, x_n)$ as a request sequence of items that arrive sequentially. Let $P = (x_1^-, \ldots, x_n^-)$ be a sequence of items called the* prediction *such that $x_i \in [x_i^-, \min(x_i^+, 1)]$, where $x_i^+ := \frac{1+\delta}{1-\delta} x_i^-$. Let K be an* *initially empty set called the* knapsack*. An algorithm is given P and δ before the first item of I is revealed. At each step $i \in \{1, \ldots, n\}$, the item x_i of the request sequence is given. An algorithm then has one of the following options:*

- ***Pack*** *If $\sum_{x_k \in K} x_k + x_i \leq 1$, set $K := K \cup x_i$.*
- ***Reject*** *Do nothing.*

The Online Simple Knapsack with Bounded Predictions *problem (*Oskp*) is then for an algorithm* `ALG` *to maximize the sum of items in K, i.e. $\sum_{x_k \in K} x_k$.*

We will write $P_{[i,n]}$ to denote the infix of P from index i to n, including both endpoints. When referring to an item x^-, we will sometimes speak of the *minimum (possible) size* of an item, as well as speaking of the *maximum (possible) size* of an item when referring to an item x^+. For every instance I with predictions P, we define $b^- := \max P$ as the *announced largest item* of the instance. This is of course not necessarily the item of largest *actual* size. If b^- is not unique, we refer to the first of such items regarding the ordering of the prediction sequence.

Note that restricting the maximum *actual* size of items to 1 is necessary to ensure competitiveness, as otherwise the classical counterexample by Marchetti-Spaccamela and Vercellis [27] can be reconstructed by letting $P = (\varepsilon, 1)$ for some $\varepsilon > 0$ and setting the second item to a value larger than one if an algorithm rejects the first item.

Strictly speaking, any competitive ratio given in this work is a function of a fixed value δ. To simplify notation, we will write c for $c(\delta)$, as it is always either clear from the context or not of relevance which concrete δ the competitive ratio refers to.

3 Related Work and Our Contributions

It is important to note the difference between this model and the one of *machine learned advice*, which has recently been called *untrusted predictions* or just *predictions*. The latter has been introduced by Lykouris and Vassilvitskii [26] and works as follows. An algorithm is given a prediction on the input, just as is in our model. However, no bound is given on the error, which is rather treated as a variable and algorithms are designed with the aim of them meeting three characteristics. The aim in algorithm design in this model is that they output an optimal solution when the given prediction is correct (*consistency*), that they perform as well as a regular online algorithm on the problem when the predictions become arbitrarily bad (*robustness*) and that they degrade with increasing unreliability of the prediction (*smoothness*).

The very important difference between this model and ours, the first one, is that the algorithm is given knowledge about the maximum error that it can expect, namely δ, as part of its input. Classically, this error term is unknown and an algorithm is to perform as well as possible. We believe this model to be equally reasonable as that without bounds on the predictions, as it is not unnatural for a person to be sure that their prediction will not be arbitrarily far off from the truth.

The classical prediction model has seen a big influx of results in the past few years, with the model being applied to several different online problems, such as scheduling [25, 14, 7], metric algorithms [2, 3], matching problems [16, 23], spanning tree problems [17, 10] and many more. Our modified model is, however, not our original modification. Azar et al. [5] recently discussed scheduling problems based on this model, where they developed algorithms that can learn about the maximum distortion and make decisions based upon this value. In a more recent work, they extended their analysis to the scheduling on multiple machines [6], in which they also analyzed the case in which an algorithm is oblivious to the actual distortion of the instance.

In order to avoid confusion, as Azar et al. still speak of *problems with predictions* in their works, we name this specific model *bounded predictions* in the context of this work. We believe that using this slightly modified version of the original prediction model can yield an even more fine-grained way of worst-case analysis, when one can assume that an oracle can only be wrong up to a known, bounded degree.

While we assume that an adversary is able to control both the predicted instance and the actual distortion of the items, there is a related model of *smoothed analysis*, in which an adversary can fix an instance, which is then subject to some random (commonly Gaussian) distortion, or *noise*. This model of an adversary not having complete control over its prepared instance was first made popular when showing that the simplex algorithm runs in expected polynomial time when its input is subjected to such random noise [30]. Since then, there was a large influx of results in this area for a wide range of problems, such as multi-level feedback algorithms [8], analyzing the k-means method [4] for clustering, and especially the 0/1 knapsack problem [9]. The model of smoothed analysis thus gives evidence that the worst-case running time or worst-case approximation ratios often seem to suffer from very specific and limited adversarial inputs which break down if even only a very slight perturbation of the instance is given. This model of an announced instance being perturbed has been studied in slight variations already. The *robust knapsack problem* by Monaci, Pferschy and Serafini [28] is very similar in that it also allow for an uncertain input with a multiplicative factor, but the authors look at *offline* algorithms that see the complete permuted instance at once and are compared to the performance of a non-perturbed instance. Im et al. [20] recently looked at the general knapsack problem, which they study under a model predicting the frequency of items of each size. Angelopoulos, Kamali and Shadkami [1] look at the online bin packing problem with predictions on the frequency of item sizes in the instance. Boyar, Favrholdt and Larsen [15] very recently studied the online simple knapsack problem with predictions, but working with predictions on the *average* size of the items an optimal solution would pack. Xu and Zhang [32] recently studied the simple knapsack problem in a learning-augmented setting, where they design algorithms that are able to learn and use the error of prediction.

We study the behavior of the OSKP problem with a distortion of $0 < \delta < 1$. The difference between the predicted size and the actual size of an item is determined by a relative error.

While we are not able to give tight bounds on the competitive ratio for all values of δ between 0 and 1, we are able to carve out the following, partial picture, which is visualized in Figure 1, with the bounds that we prove also given in Table 1. For reasons of readability, we will refrain from stating these terms found in the table in the following text. They will of course be stated in their respective theorems later.

Up to $\delta \leq \frac{1}{7}$, we give a tight bound of 2 on the competitive ratio. From there on, three bounds on the competitive ratio repeat periodically, given a fixed $k \in \mathbf{N}$: Between values of δ between $\frac{k^2+k-1}{k^2+3k+3}$ and δ_k (as defined in Table 1), we are able to prove a tight bound of $\sqrt{(2+k)\frac{1+\delta}{1-\delta}}$. From δ_k to $-1-k+\sqrt{4k+k^2}$, we are able to prove a non-matching lower and upper bound. For the penultimate segment of $-1-k+\sqrt{4k+k^2} \leq \delta \leq \frac{1}{k+2}$, we give a matching upper and lower bound. The final segment, which connects to the first segment for the next higher value of k, we are again only able to prove a non-matching pair of bounds. Noticeably, the function of the competitive ratio depending on δ is not continuous between segments two and three.

Table 1 Competitive ratios of the OSKP problem, given a fixed $k \in \mathbf{N}$ and distortion δ.

Distortion	Competitive ratio
$0 < \delta \leq \frac{1}{7}$	2
$\frac{k^2+k-1}{k^2+3k+3} < \delta \leq \delta_k$	$\sqrt{(2+k)\frac{1+\delta}{1-\delta}}$
$\delta_k \leq \delta < -1-k+\sqrt{4k+k^2}$	$\geq \sqrt{(2+k)\frac{1+\delta}{1-\delta}} \mid\leq \frac{(1+\delta)(\sqrt{-1+\delta^2}+\sqrt{-17-16\delta+\delta^2-16k})}{4\sqrt{-1+\delta^2}}$
$-1-k+\sqrt{4k+k^2} \leq \delta \leq \frac{k}{k+2}$	$\frac{(1+\delta)(\sqrt{-1+\delta^2}+\sqrt{-17-16\delta+\delta^2-16k})}{4\sqrt{-1+\delta^2}}$
$\frac{k}{k+2} < \delta < \frac{k^2+k-1}{k^2+3k+3}$	$\geq \frac{1+k}{2} - \frac{\sqrt{(-1+\delta)(-5-k(2+k)+\delta(-1+k)(3+k))}}{-2+2\delta} \mid\leq \frac{1+\delta+\sqrt{5+2\delta-3\delta^2}}{2-2\delta}$

where $\delta_k := \frac{12k+8}{32^{\frac{2}{3}}(3\sqrt{3}\sqrt{4k^3+11k^2+28k+44}-9k-34)^{\frac{1}{3}}} + \frac{1}{3}2^{\frac{2}{3}}(3\sqrt{3}\sqrt{4k^3+11k^2+28k+44}-9k-34)^{\frac{1}{3}} + 5/3$

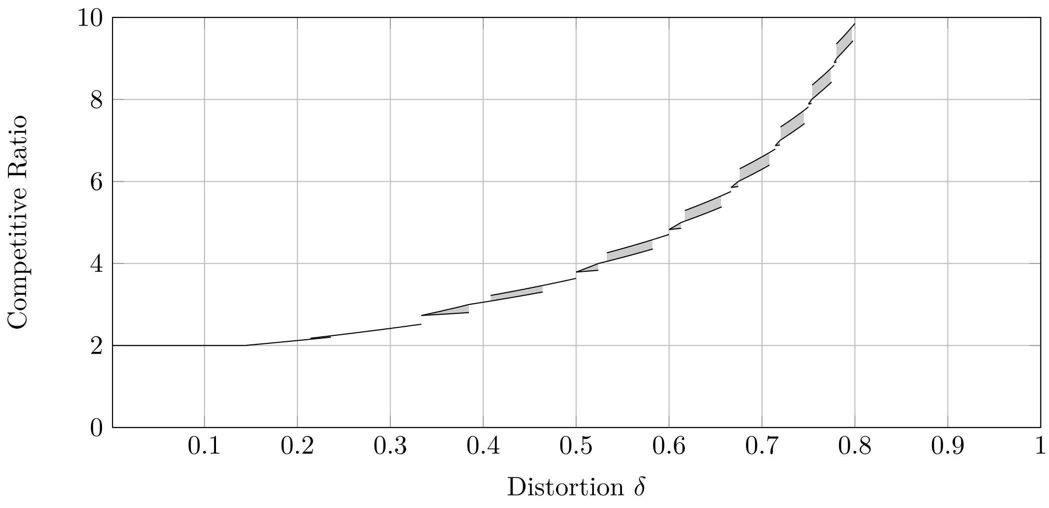

Figure 1 Competitive ratio of the OSKP problem, depending on the distortion δ. The gray areas signify a gap between the best known lower and upper bounds.

To the best of our knowledge, our work is the first one systematically analyzing a knapsack problem in the setting of predictions, where the distortion (or error) is linked to the size of the presented items. We show that the additional knowledge of a bound on the distortion can significantly help an online algorithm to choose an appropriate strategy, as e.g. the algorithms used for the repeating segments two and three differ in their nature. Thus, a much more fine-grained analysis is possible. Noticeably, even for an uncertain prediction where the actual value of an item may deviate by a factor of three from its prediction, one can guarantee a competitive ratio of less than four. Yet, even for arbitrarily small distortions, no competitive ratio better than two can be achieved.

The remainder of this work is structured as follows: We analyze the model of relative errors by first proving a number of helpful structural lemmata in Section 4, followed by upper bound proofs in Section 5 and lower bounds in Section 6. We conclude with a short discussion of open problems and possible future work in Section 7.

4 Structural Observations

When trying to design algorithms that are supposed to achieve some given competitive ratio $c \geq 1$, there are many instances for which a solution can be found more or less trivially. Since these solutions have to be handled explicitly by almost every of our algorithms, we handle them such that we can refer to them in our algorithm designs without risking redundancy. We first start with four very simple cases, that nevertheless need to be explicitly handled by our algorithms.

The first lemma deals with instances of very small total size.

▶ **Lemma 2.** *Any instance I of the* OSKP *problem with predictions P, such that $\sum_{x^- \in P} x^- \leq \frac{1-\delta}{1+\delta}$, can be solved optimally by packing all items greedily.*

Proof. Since $x^+ \leq \frac{1+\delta}{1-\delta}x^-$, it holds that $\sum_{x \in I} x \leq \sum_{x^- \in P} \frac{1+\delta}{1-\delta}x^- = \frac{1+\delta}{1-\delta}\sum_{x^- \in P} x^- \leq \frac{1+\delta}{1-\delta}\frac{1-\delta}{1+\delta} = 1$. Thus, all items of the instance are guaranteed to fit into the knapsack. ◀

The second lemma explicitly deals with instances that contain a subset of items that is both guaranteed to fit together and of sufficient size.

▶ **Lemma 3.** *Any instance I of the* OSKP *problem with predictions P such that there is some $S \subseteq P$ such that $\sum_{x^- \in S} x^- \in [\frac{1}{c}, \frac{1-\delta}{1+\delta}]$ can be solved with a competitive ratio of at most c by packing exactly S.*

Proof. By definition, the items of S are both guaranteed to fit together in the knapsack and guaranteed to be of total size at least $\frac{1}{c}$. ◀

The third lemma ensures that we will only need to deal with instances in which no single item yields the wanted competitive ratio.

▶ **Lemma 4.** *Any instance I of the* OSKP *problem with predictions P such that $x^- \in P$ with $x^- \geq \frac{1}{c}$ can be solved with a competitive ratio of at most c by packing exactly x.*

The fourth lemma is slightly more interesting. While the previous statement upper bounded the size of the announced largest item b^- by $\frac{1}{c}$, we are also able to lower bound the size of this item as follows.

▶ **Lemma 5.** *Any instance I of the* OSKP *problem with predictions P such that $b^- \leq (1-\frac{1}{c})/\frac{1+\delta}{1-\delta}$ admits a c-competitive, greedy algorithm.*

Proof. Since b^- is by definition the announced largest item of the instance, we know that the size of the largest actual item is upper bounded by $\frac{1+\delta}{1-\delta}b^-$. Thus, the largest item that a greedy algorithm can *not* fit into its knapsack is of size at most $\frac{1+\delta}{1-\delta}b^- \leq \frac{1+\delta}{1-\delta}(1-\frac{1}{c})/\frac{1+\delta}{1-\delta} = 1 - \frac{1}{c}$. If such an item does not fit, then the knapsack is already packed to size at least $\frac{1}{c}$. ◀

Thus, we can assume that $b^- \in [(1-\frac{1}{c})/\frac{1+\delta}{1-\delta}, \frac{1}{c}]$, which allows us to partition the predicted items into two categories, which we will call *small* and *large*. We call an item small if its announced size together with b^- does not guarantee a packing of sufficient size, assuming they are presented as small as possible. Formally, an item $x^- \in P$ is small iff $x^- + b^- < \frac{1}{c} \Leftrightarrow x^- < \frac{1}{c} - b^-$. On the other hand, we call an item large if, together with b^-, it *may* be the case that the two items do not fit together in the knapsack (or if the sum of their announced sizes already exceeds 1.) Formally, an item $x^- \in P$ is large iff $\frac{1+\delta}{1-\delta}x^- + \frac{1+\delta}{1-\delta}b^- > 1 \Leftrightarrow x^- > \frac{1-\delta}{1+\delta} - b^-$.

Note that the existence of any item that falls in neither category already implies a trivial solution of size at least $\frac{1}{c}$ by Lemma 3, with the exception of b^- itself. However, we can see that b^- exceeds the minimum size of a large item, as

$$b^- > \frac{1-\delta}{1+\delta} - b^- \Leftrightarrow 2(1-\frac{1}{c})/\frac{1+\delta}{1-\delta} > \frac{1-\delta}{1+\delta}$$

holds true for all δ iff $c \geq 2$ – even when substituting b^- by its minimum possible value – which will be the case for all bounds that we prove in this paper. Thus, the size of b^- is always large enough to call it a large item as well.

This partition has some very useful implications, with the very central one being that an algorithm can essentially ignore small items of the instance. To show this, we first prove that the total sum of small items in P is limited or the instance has a trivial solution.

▶ **Lemma 6.** *Let I be an instance of the* OSKP *problem with predictions P and $S := \{x^- \in P \mid x^- < \frac{1}{c} - b^-\}$ for $c \geq \frac{\delta^2+3}{2-2\delta}$. If P does not admit a solution due to Lemmata 2, 3, 4 or 5 and if $\sum_{x^- \in S} x^- \geq \frac{1}{c} - b^-$ holds, then there exists an algorithm that is c-competitive.*

Proof. The algorithm works as follows: It first selects an arbitrary subset of small items $S \subseteq P$ such that $\sum_{x^- \in S} x^- \in [\frac{1}{c} - b^-, 2(\frac{1}{c} - b^-)]$. Such a subset always exists and can be found greedily since each individual small item is announced smaller than $\frac{1}{c} - b^-$.

The algorithm packs b^- when it is revealed and small items from S greedily, but stopping to pack further small items as soon as their total size is at least $\frac{1}{c} - b^-$. Together with b^-, the gain is obviously at least $\frac{1}{c}$, so the only thing left to show is that such a subset is always guaranteed to fit in the knapsack and does not exceed its capacity.

In the worst case, the algorithm packs a small item of size $\frac{1}{c} - b^- - \varepsilon_1$ and is then presented an item of size $\frac{1+\delta}{1-\delta}(\frac{1}{c} - b^- - \varepsilon_2)$ for some $\varepsilon_1, \varepsilon_2 > 0$. Afterwards, b^- is presented of maximum possible size $\frac{1+\delta}{1-\delta}b^-$. Yet even in this case, all items fit into the knapsack, as

$$\begin{aligned}
&\frac{1}{c} - b^- - \varepsilon_1 + \frac{1+\delta}{1-\delta}(\frac{1}{c} - b^- - \varepsilon_2) + \frac{1+\delta}{1-\delta}b^- \\
&< \frac{1}{c} - b^- + \frac{1+\delta}{1-\delta}(\frac{1}{c} - b^-) + \frac{1+\delta}{1-\delta}b^- \\
&= \frac{1}{c} - b^- \frac{1+\delta}{1-\delta}\frac{1}{c} \\
&= (1 + \frac{1+\delta}{1-\delta})\frac{1}{c} - b^- \\
&< (1 + \frac{1+\delta}{1-\delta})\frac{1}{c} - (1 - \frac{1}{c})/\frac{1+\delta}{1-\delta}
\end{aligned}$$

Solving $(1 + \frac{1+\delta}{1-\delta})\frac{1}{c} - (1 - \frac{1}{c})/\frac{1+\delta}{1-\delta} \leq 1$ for c yields that these items fit if $c \geq \frac{\delta^2+3}{2-2\delta}$, which is lower than every upper bound proven in this work, for all δ. ◀

Thus, we can assume that the *sum* of all announced small items is smaller than $\frac{1}{c} - b^-$, or else a trivial solution exists. This in turn lets us show a nice relation between small and large items, which is that the larger the sum of small items is, the smaller any large item may be announced or there is again a trivial solution.

▶ **Lemma 7.** *Let I be an instance of the* OSKP *problem with predictions P and $S := \{x^- \in P \mid x^- < \frac{1}{c} - b^-\}$ with $c \geq 1$. If P does not admit a solution due to Lemmata 2, 3, 4 5 or 6 and $b^- \geq \frac{1}{c} - \sum_{x^- \in S} x^-$, then there exists an algorithm that is c-competitive.*

Proof. The algorithm packs b^- together with the items from S. Since $\sum_{x^- \in S} x^- < \frac{1}{c} - b^-$ due to Lemma 6, the total size is at least $b^- + \sum_{x^- \in S} x^- \geq \frac{1}{c} - \sum_{x^- \in S} x^- + \sum_{x^- \in S} x^- = \frac{1}{c}$. ◀

This lemma has the pleasant consequence that we will not have to worry about small items in most of our algorithm design beyond checking whether they are part of a trivial solution. Since in the upcoming analysis, we often bound the packed items of our algorithm against the largest item of the instance, it makes no difference to us whether there are no small items in the instance and thus a very large b^-, or if b^- is made smaller for the sake of additional small items in the instance.

The next lemma allows us to lower bound the announced size of large items depending on the size of b^-.

▶ **Lemma 8.** *Let I be an instance of the OSKP problem with predictions P and $x^- \in P$ be a large item such that $x^- \neq b^-$. If $x + \frac{1+\delta}{1-\delta} b^- \leq 1$, packing x together with b guarantees a gain of $\frac{1}{c}$ with $c \geq 1$.*

Proof. A large item has actual size at least $\frac{1-\delta}{1+\delta} - b^-$, thus $x + \frac{1+\delta}{1-\delta} b^- > \frac{1-\delta}{1+\delta} - b^- + b^- = \frac{1-\delta}{1+\delta} > \frac{1}{c}$. ◀

This means that whenever an algorithm is guaranteed to be able to pack any large item together with b^-, its gain is sufficient.

Using these observations, we define with Algorithm 1 a small subroutine that will be called by our other algorithms in order to rule out trivial solutions to an instance.

Algorithm 1 Subroutine *filter_trivial*.

```
1: if $\sum_{x^- \in P} x^+ \leq 1$ then                                   ▷ Lemma 2
2:     Pack $b$. END
3: if $b^- \geq \frac{1}{c}$ then                                         ▷ Lemma 4
4:     Pack $b$. END
5: if $b^- \leq \frac{1}{c} / \frac{1+\delta}{1-\delta}$ then              ▷ Lemma 5
6:     Greedily pack items. END
7: if $\exists S \in 2^P : \sum_{x^- \in S} x^- \in [\frac{1}{c}, \frac{1-\delta}{1+\delta}]$ then   ▷ Lemma 3
8:     Pack all items of $S$. END
9: $T := \{x^- \in P \mid x^- < \frac{1}{c} - b^-\}$
10: if $\sum_{x^- \in T} x^- \geq \frac{1}{c} - b^-$ then                  ▷ Lemma 6
11:     Pack a subset $R \in T$ with $\sum_{x^- \in R} x^- \in [\frac{1}{c} - b^-, 2(\frac{1}{c} - b^-)]$ and $b$. END
12: if $b^- \geq \frac{1}{c} - \sum_{x^- \in T} x^-$ then                   ▷ Lemma 7
13:     Pack all items of $T$ and $b$. END
```

5 Upper Bounds

The analysis of upper bound algorithms remains complicated, even with the filters of Section 4. While we are quite confident that the lower bounds of Section 6 should not be improvable, finding matching upper bounds is an aim that we can only fulfill for parts of the whole range of values for δ.

We start with a simple 2-competitive algorithm for all values of δ up to $\frac{1}{7}$.

▶ **Theorem 9.** *Given a fixed δ with $0 < \delta \leq \frac{1}{7}$. Algorithm 2 solves the OSKP problem with a competitive ratio of at most $c = 2$.*

Algorithm 2 2-competitive Algorithm for $0 < \delta \leq \frac{1}{7}$.

```
1: filter-trivial()
2: Reveal items up to and including the first large item $x_1$.
3: if $x_1 \leq 1 - \frac{1+\delta}{1-\delta} b^-$ and $x_1 \neq b$ then
4:     Pack $x_1$ with $b$. END
5: else
6:     Greedily pack large items, including $x_1$.
```

Proof. If the algorithm ends after the call of *filter_trivial*, the algorithm is at most 2-competitive. Assuming the condition in line 3 is met, the algorithm is at least 2-competitive by Lemma 8. Thus, we only have to prove that the algorithm is not worse than 2-competitive if it ends in line 6.

Let us first assume that $x_1 \neq b$. Then x_1 is packed and $x_1 > 1 - \frac{1+\delta}{1-\delta} b^-$. If any other large item x_i can be packed by the algorithm, the knapsack will be filled up to at least $\frac{1}{2}$, since

$$x_1 + x_i > 1 - \frac{1+\delta}{1-\delta} b^- + \frac{1-\delta}{1+\delta} - b^- > 1 - \frac{1+\delta}{1-\delta}\frac{1}{2} + \frac{1-\delta}{1+\delta} - \frac{1}{2} \geq \frac{1}{2} \,,$$

where the last inequality holds for $\delta \leq 3 - 2\sqrt{2} \sim 0.171$.

If no second large item fits into the algorithm's knapsack, an optimal solution cannot contain more than one large item either. This item of the optimal solution is bounded by $\frac{1+\delta}{1-\delta} b^- < \frac{1+\delta}{1-\delta}\frac{1}{2}$. The remaining items of an optimal solution then consist of all small items that the algorithm has ignored. We can bound the total announced size of these small items, by $\frac{1}{2} - b^-$, using Lemma 6 and in consequence their actual size by $\frac{1+\delta}{1-\delta}(\frac{1}{2} - b^-)$. Putting it all together, in the worst case Algorithm 2 packs exactly one large item of size slightly larger than $1 - \frac{1+\delta}{1-\delta} b^-$. The optimal solution packs one large item of size slightly smaller than $\frac{1+\delta}{1-\delta}\frac{1}{2}$ and the maximum sum of small items. This results in a competitive ratio of

$$\frac{\frac{1+\delta}{1-\delta} b^- + \frac{1+\delta}{1-\delta}(\frac{1}{2} - b^-)}{1 - \frac{1+\delta}{1-\delta} b^-} = \frac{\frac{1+\delta}{1-\delta}\frac{1}{2}}{1 - \frac{1+\delta}{1-\delta} b^-} < \frac{\frac{1+\delta}{1-\delta}\frac{1}{c}}{1 - \frac{1+\delta}{1-\delta}\frac{1}{c}} \leq 2 \,,$$

for $\delta \leq \frac{1}{7}$.

The only case left to handle is that $x_1 = b$. If $b \geq 1 - \frac{1+\delta}{1-\delta} b^-$, we can use the same argumentation as before. If however $b < 1 - \frac{1+\delta}{1-\delta} b^-$, then, since b is by definition the announced largest item, *all* large items are announced smaller or equal than $1 - \frac{1+\delta}{1-\delta} b^-$ and thus of actual size at most $\frac{1+\delta}{1-\delta}(1 - \frac{1+\delta}{1-\delta} b^-) < \frac{1+\delta}{1-\delta}(1 - \frac{1+\delta}{1-\delta}\frac{1}{2} / \frac{1+\delta}{1-\delta}) = \frac{1+\delta}{1-\delta}\frac{1}{2}$. As each large item is of size at least $\frac{1-\delta}{1+\delta} - b^-$, packing a second large item is still sufficient, since $2(\frac{1-\delta}{1+\delta} - b^-) > 2\frac{1-\delta}{1+\delta} - 1 \geq 1/2$ for $\delta \leq \frac{1}{7}$. Since no large item can exceed an actual size of $\frac{1+\delta}{1-\delta} b^-$ and since $x_1 < 1 - \frac{1+\delta}{1-\delta} b^-$, we are guaranteed that a second large item fits into the knapsack of the algorithm. The only case left is that there is no second large item. Then the algorithm and the optimal solution both pack the same large item $x_1 = b$, while the optimal solution can still add the maximum number of small items. The resulting competitive ratio is then $\frac{b^- + \frac{1+\delta}{1-\delta}(\frac{1}{2} - b^-)}{b^-} < 2$ for all δ in the given range. ◀

The second repeating upper bound matches the lower bound of Theorem 14 for almost the complete range of δ. For brevity, let $\delta_k := \frac{12k+8}{3 2^{\frac{2}{3}} (3\sqrt{3}\sqrt{4k^3+11k^2+28k+44} - 9k - 34)^{\frac{1}{3}}} + \frac{1}{3} 2^{\frac{2}{3}} (3\sqrt{3}\sqrt{4k^3 + 11k^2 + 28k + 44} - 9k - 34)^{\frac{1}{3}} + 5/3$.

Algorithm 3 $\sqrt{(2+k)\frac{1+\delta}{1-\delta}}$-competitive Algorithm for a $k \in \mathbf{N}$ and $\delta < \delta_k$.

```
1: filter_trivial()
2: Reveal items up to and including the first large item $x_1$.
3: if $x_1 \geq \frac{1}{k+2}$ or $x_1 = b$ then
4:     Greedily pack large items, including $x_1$. END
5: if $x_1 \leq 1 - \frac{1+\delta}{1-\delta} b^-$ then
6:     Pack $x_1$ with $b$. END
7: Let $B := \{x^- \in P \mid x^- > \frac{1-\delta}{1+\delta} - b^-\}$
8: if $\exists x_j^- \in B : x_j^- \leq (1-x_1)/\frac{1+\delta}{1-\delta}$ then
9:     Pack $x_1$ and $x_j$. END
10: for Reveal next $x_i \in P_{[i,n-1]}$ do
11:     if $\exists S \subseteq P_{[i,n-1]} : x_1 + \sum_{x^- \in S} x^- \geq \frac{1}{c_k} \wedge x_1 + \sum_{x^- \in S} x^+ \leq 1$ then
12:         Pack $x_1$ and $S$. END
13: Pack $x_n$.
```

▶ **Theorem 10.** *Given a fixed $k \in \mathbf{N}$ and a fixed δ with $\delta < \delta_k$. Algorithm 3 solves the* OSKP *problem with a competitive ratio of at most $c_k = \sqrt{(2+k)\frac{1+\delta}{1-\delta}}$.*

Proof. If the algorithm ends after the call of *filter_trivial*, the algorithm is at most c_k-competitive. The algorithm first reveals and discards small items and reveals the first large item x_1. If $x_1 \geq \frac{1}{k+2}$, the algorithm packs x_1 and from there on greedily any large item that still fits into the knapsack. If any second large item fits into the knapsack of the algorithm, its gain is at least $\frac{1}{k+2} + \frac{1-\delta}{1+\delta} - b^- > \frac{1}{c_k}$. Thus, an optimal solution consists of at most one large item of maximum size or a sum of large and small items adding up to at most the same size, due to Lemma 7. The competitive ratio is then $\frac{1+\delta}{1-\delta}\frac{1}{c_k}/\frac{1}{k+2} = c_k$.

Assuming $x_1 = b$, we know from Lemma 8 that any large item that is guaranteed to fit with b yields a ratio of at most c_k. Again, the algorithm packs x_1 and large items greedily. If any second large item fits into the knapsack of the algorithm, its gain is at least $b + (1-b)/\frac{1+\delta}{1-\delta} > \frac{1}{c_k}$ for the complete range of δ. Thus, an optimal solution consists of at most one large item of maximum size. The competitive ratio is then at worst $\frac{1+\delta}{1-\delta}b/b < c_k$ for the complete range of α.

Finally, if $x_1 \leq 1 - \frac{1+\delta}{1-\delta}b^-$, the algorithm packs x_1 together with b and has sufficient gain by Lemma 8.

Thus, we assume that $1 - \frac{1+\delta}{1-\delta}b^- < x_1 < \frac{1}{k+2}$. The algorithm next checks whether any other large item is of announced size such that it is both guaranteed to fit together with x_1 and guaranteed to fit into the knapsack together with x_1, i.e. if any $x_j^- \in B$ exists with $x_j^- \in [\frac{1}{c_k} - x_1, \frac{1-x_1}{\frac{1+\delta}{1-\delta}}]$. Note that $\frac{1}{c_k} - x_1 < \frac{1-\delta}{1+\delta} - b^-$, i.e. there exists no large item on the lower end of this bound if $\delta < \delta_k$. Thus, each large item apart from x_1 is of actual size larger than $(1-x_1)/\frac{1+\delta}{1-\delta}$.

If all of these bounds hold and thus do not admit a solution, the algorithm discards x_1 and continues to reveal large items. It only packs anything before the last large item if either the revealed item itself is of size $\frac{1}{c_k}$ or if it admits a packing that is guaranteed to fit and of size at least $\frac{1}{c_k}$. If neither is the case, the algorithm packs the last large item of size larger than $(1-x_1)/\frac{1+\delta}{1-\delta}$.

In the worst case, the optimal solution then consists of $k+1$ large items of size slightly smaller than $\frac{1}{c_k}$ each. The competitive ratio is then bounded by

$$\frac{k+1}{\sqrt{(2+k)\frac{1+\delta}{1-\delta}}} \Big/ \frac{1-\frac{1}{k+2}}{\frac{1+\delta}{1-\delta}} = \sqrt{(2+k)\frac{1+\delta}{1-\delta}}\,.$$

Note that the first item is too large to be packed on top of the $k+1$ items as

$$\frac{k+1}{\sqrt{(2+k)\frac{1+d}{1-d}}} + (1 - \frac{1+d}{1-d}\frac{1}{\sqrt{(2+k)\frac{1+d}{1-d}}}) > 1$$

for $\delta < \frac{k}{k+2}$. ◀

The next segment's proof is the most involved one. It tightly matches the lower bound of Theorem 15 for $-1-k+\sqrt{4k+k^2} \le \delta \le \frac{k}{k+2}$, for any $k \in \mathbf{N}$.

Algorithm 4 $\frac{(1+\delta)(\sqrt{-1+\delta^2}+\sqrt{-17-16\delta+\delta^2-16k})}{4\sqrt{-1+\delta^2}}$-competitive Algorithm for a $k \in \mathbf{N}$ and $\delta \le \frac{k}{k+2}$.

```
1: *filter_trivial()*
2: Reveal items up to and including the first *large* item $x_1$.
3: Let $\ell_k := \frac{8}{9+8\delta-\delta^2-\sqrt{-1+\delta^2}\sqrt{-17-16\delta+\delta^2-16k}+8k}$
4: **if** $x_1 \ge \ell_k$ or $x_1 = b$ **then**
5:     Greedily **pack** large items, including $x_1$. END
6: **if** $x_1 \le 1 - \frac{1+\delta}{1-\delta}b^-$ **then**
7:     **Pack** $x_1$ with $b$. END
8: Let $B := \{x^- \in P \mid x^- > \frac{1-\delta}{1+\delta} - b^-\}$
9: **if** $\exists x_j^-, x_l^- \in B : x_1 + x_j^+ + x_l^+ \le 1$ **then**
10:     **Pack** $x_1, x_j$ and $x_l$. END
11: **else if** $\exists x_L^- \in B : x_1 + x_L^- \ge \frac{1}{c_k} \wedge x_1 + x_L^+ \le 1$ **then**
12:     **Pack** $x_1$ and $x_L$. END
13: **for** Reveal next $x_i \in P_{[i,n-1]}$ **do**
14:     **if** $\exists S \subseteq P_{[i,n-1]} : x_1 + \sum_{x^- \in S} x^- \ge \frac{1}{c_k} \wedge x_1 + \sum_{x^- \in S} x^+ \le 1$ **then**
15:         **Pack** $x_1$ and $S$. END
16: **Pack** $x_n$.
```

▶ **Theorem 11.** *Given a fixed $k \in \mathbf{N}$ and a fixed δ with $\delta \le k/(k+2)$. Algorithm 4 solves the* OSKP *problem with a competitive ratio of at most $c_k = \frac{(1+\delta)(\sqrt{-1+\delta^2}+\sqrt{-17-16\delta+\delta^2-16k})}{4\sqrt{-1+\delta^2}}$.*

Proof. If the algorithm ends after the call of *filter_trivial*, the algorithm is at most c_k-competitive. Let $\ell_k = \frac{8}{9+8\delta-\delta^2-\sqrt{-1+\delta^2}\sqrt{-17-16\delta+\delta^2-16k}+8k}$, which is the solution to the equation $\frac{1+\delta}{1-\delta}\frac{1}{c_k}/\ell_k = c_k$.

The algorithm first reveals and discards small items and reveals the first large item x_1. If $x_1 \ge \ell_k$, the algorithm packs x_1 and from there on greedily any large item that still fits into the knapsack. If any second large item fits into the knapsack of the algorithm, its gain is at least $\ell_k + \frac{1-\delta}{1+\delta} - b^- > \frac{1}{c_k}$. Thus, an optimal solution consists of at most one large item of maximum size. The competitive ratio is then $\frac{1+\delta}{1-\delta}\frac{1}{c_k}/\ell_k = c_k$.

Assuming $x_1 = b$, we know from Lemma 8 that any large item that is guaranteed to fit with b yields a ratio of at most c_k. Again, the algorithm packs x_1 and large items greedily. If any second large item fits into the knapsack of the algorithm, its gain is at least

$b + (1-b)/\frac{1+\delta}{1-\delta} > \frac{1}{c_k}$ for the complete range of δ. Thus, an optimal solution consists of at most one large item of maximum size. The competitive ratio is then at worst $\frac{1+\delta}{1-\delta}b/b < c_k$ for the complete range of α.

Finally, if $x_1 \leq 1 - \frac{1+\delta}{1-\delta}b^-$, the algorithm packs x_1 together with b and has sufficient gain by Lemma 8.

Thus, we assume that $1 - \frac{1+\delta}{1-\delta}b^- < x_1 < \ell_k$. If another two large items are guaranteed to fit together with x_1, the gain of the algorithm is again sufficient: $1 - \frac{1+\delta}{1-\delta}b^- + 2(\frac{1-\delta}{1+\delta} - b^-) > 1 - \frac{1+\delta}{1-\delta}\frac{1}{c_k} + 2(\frac{1-\delta}{1+\delta} - \frac{1}{c_k}) > \frac{1}{c_k}$. Thus, we may assume that there is at most one other large item x_j that is guaranteed to fit into the knapsack together with x_1. Its announced size is then in between $\frac{1-\delta}{1+\delta} - b^-$ and $\frac{1}{c_k} - x_1 < \frac{1}{c_k} - (1 - \frac{1+\delta}{1-\delta}b^-) < \frac{1}{c_k} - (1 - \frac{1+\delta}{1-\delta}\frac{1}{c_k}) = (1 + \frac{1+\delta}{1-\delta})\frac{1}{c_k} - 1$. Obviously, it has to hold that $x_1 + x_j < \frac{1}{c_k}$, otherwise the two items together are sufficient. Since the remaining large items $x_L \in I$ must not be announced with a size that guarantees that they fit into the knapsack together with x_1 *and* guarantee a packing of at least $\frac{1}{c_k}$, they have to be of announced size $x_L^- > \frac{1-x_1}{\frac{1+\delta}{1-\delta}}$. Since $x_1 < \ell_k < \frac{1}{k+2}$, we can lower bound the size of these large items by $x_L^- > (1-x_1)/\frac{1+\delta}{1-\delta} > (1 - \frac{1}{k+2})/\frac{1+\delta}{1-\delta} \geq \frac{1}{k+2}$, for all $\delta \leq \frac{1}{k+2}$. Since $\frac{1}{k+2} + \frac{1-\delta}{1+\delta} - b^- > \frac{1}{c_k}$, the item x_j that was guaranteed to fit together with x_1 must not be guaranteed to fit together with any other large item as well. This lower bounds its size by $x_j^- > (1 - \frac{1}{c_k})/(\frac{1+\delta}{1-\delta}) > x_1/\frac{1+\delta}{1-\delta}$.

If all of these bounds hold and thus do not admit a solution, the algorithm discards x_1 and continues to reveal large items. It only packs anything before the last large item if either the revealed item itself is of size $\frac{1}{c_k}$ or if it admits a packing that is guaranteed to fit and of size at least $\frac{1}{c_k}$. If neither is the case, the algorithm packs the last large item of size larger than $(1-x_1)/\frac{1+\delta}{1-\delta}$.

Recall that $x_L^- > (1-x_1)/\frac{1+\delta}{1-\delta}$, which means that $(k+1)x_L^- + x_1 > 1$ for $\delta \leq \frac{1}{k+2}$. Since also $x_1 + x_j > \frac{1}{c_k} > x_L^-$, the largest packing of a knapsack consists of k items x_L or size slightly smaller than $\frac{1}{c_k}$, together with both items x_1 and x_j. The competitive ratio is then bounded by

$$(x_1 + x_j + k\frac{1}{c_k})/\frac{1-x_1}{\frac{1+\delta}{1-\delta}} \ .$$

We are now interested in maximizing this ratio by choosing appropriate values for x_1 and x_j. The gain of the algorithm is minimized if the value of x_1 is maximized. Since $x_j^- > x_1/\frac{1+\delta}{1-\delta}$ and $x_1 + x_j < \frac{1}{c_k}$ both have to hold, the ratio is maximized to a value of c_k when choosing $x_1 = \frac{2\sqrt{-1+\delta^2}}{\sqrt{-1+\delta^2}+\sqrt{-17-16\delta+\delta^2-16k}}$ and thus $x_j = \frac{1+\delta}{1-\delta}x_1$. ◀

For the remainder of unmatched lower bounds, we re-use Algorithm 2 that was used to prove the upper bound of 2 for $0 < \delta \leq \frac{1}{7}$. The only difference is the competitive ratio that is aimed for, which is $\frac{1+\delta+\sqrt{5+2\delta-3\delta^2}}{2-2\delta}$ for $\frac{1}{7} \leq \delta$. The proof is almost identical to that of Theorem 9.

▶ **Theorem 12.** *Given a fixed $k \in \mathbf{N}$ and a fixed δ with $\frac{1}{7} \leq \delta$. Algorithm 2 solves the* OSKP *problem with a competitive ratio of at most $c = \frac{1+\delta+\sqrt{5+2\delta-3\delta^2}}{2-2\delta}$.*

Proof in Appendix. ◀

6 Lower Bounds

We are able to provide lower bounds for all values of δ. We can observe that different bounds dominate one another in an alternating fashion, which is a consequence of the scaling factor $\frac{1+\delta}{1-\delta}$ crossing certain thresholds.

Intuitively, the lower bounds – after the initial one of 2 – each work once sufficient distance between the lowest possible size and the largest possible size of an announced item crosses certain thresholds. They are then valid for higher values of δ as well, but are quickly dominated by yet other lower bounds, that again use certain thresholds between the range of possible sizes of an item. These bounds turn out to be rather tricky to find but easy to verify with the assistance of computer algebra systems. We found them by fixing concrete values of δ and carefully building a decision tree regarding an announced instance of variable values. We were then able to generalize them to surrounding values of δ using computer algebra systems.

We start with a simple lower bound of 2 for all values of $0 < \delta < 1$. This lower bound works quite similar to the bound by Marchetti-Spaccamela and Vercellis [27] used to show non-competitiveness. The main difference is that the predictions have to be almost correct for very small δ, thus presenting two items that have a large size difference allow an algorithm to gravitate towards the larger one.

▶ **Theorem 13.** *Given a fixed δ with $0 < \delta < 1$. There exists no algorithm solving the* OSKP *problem with a competitive ratio better than* 2.

Proof. Let $\varepsilon > 0$ such that $\varepsilon < \frac{\delta}{4}$. The algorithm is given the following prediction:

$$P = (\frac{1}{2} + \varepsilon, \frac{1}{2} - \varepsilon, \frac{1}{2})$$

We do a full case distinction on the possible behaviors of an algorithm. The first item is revealed to be of size $x_1 = \frac{1}{2} + \varepsilon$.

Case 1: An algorithm packs x_1 with $x_1 = \frac{1}{2} + \varepsilon$. The next two items are presented of size $x_2 = x_3 = \frac{1}{2}$, fitting together with one another but not with x_1. The competitive ratio is then $\frac{1}{\frac{1}{2}+\varepsilon} = 2\frac{1}{1+2\varepsilon}$.

Case 2: An algorithm rejects x_1 with $x_1 = \frac{1}{2} + \varepsilon$. The second item is then presented to be of size $x_2 = \frac{1}{2} - \varepsilon$, with the last item presented of size $x_3 = \frac{1}{2} + 2\varepsilon$, independent of the algorithm's decision on the item x_2. The optimal solution consists of the first two items, while the biggest packing an algorithm can achieve is $\frac{1}{2} + 2\varepsilon$. The competitive ratio is then $\frac{1}{\frac{1}{2}+2\varepsilon} = 2\frac{1}{1+4\varepsilon}$. ◀

The next bound connects seamlessly to the previous bound of 2. As with all following bounds, the following theorem describes a set of bounds that only become valid once a certain value of δ is reached.

▶ **Theorem 14.** *For every $k \in \mathbf{N}$ and $\frac{k^2+k-1}{k^2+3k+3} \leq \delta$, there exists no algorithm solving the* OSKP *problem with a competitive ratio better than $c_k = \sqrt{(2+k)\frac{1+\delta}{1-\delta}}$-competitive.*

Proof. Let $\varepsilon > 0$. The algorithm is given the following prediction:

$$P = (\frac{1}{2+k}, \underbrace{\frac{1}{c_k}, \ldots, \frac{1}{c_k}}_{k+1 \text{ many}}, \frac{1 - \frac{1}{2+k}}{\frac{1+\delta}{1-\delta}})$$

We do a full case distinction on the possible behaviors of an algorithm. The first item is presented as $\frac{1}{2+k}+\varepsilon$.

Case 1: An algorithm packs $\frac{1}{2+k}+\varepsilon$. The next items are all presented of maximum possible size. Since $\frac{1+\delta}{1-\delta}\frac{1}{c\,k}+\frac{1}{2+k}+\varepsilon>1$ for each choice of k and all $\delta\geq\frac{k^2+k-1}{k^2+3k+3}$, this means that no two items fit together into the knapsack. The biggest item of an optimal packing is then $\frac{1}{c_k}\frac{1+\delta}{1-\delta}$. The competitive ratio is then $\frac{1+\delta}{1-\delta}\frac{1}{c_k}/(\frac{1}{2+k}+\varepsilon)\overset{\varepsilon\to 0}{=}c_k$.

Case 2: An algorithm rejects $\frac{1}{2+k}+\varepsilon$. The next item is presented of size $\frac{1}{c_k}$.

Case 2.1: An algorithm packs $\frac{1}{c_k}$. The next items are each presented of size $1-\frac{1}{2+k}-\varepsilon<\frac{1+\delta}{1-\delta}\frac{1}{c_k}$, allowing an optimal packing to reach a size of 1. Since $\frac{1}{c_k}>\frac{1}{2+k}$ for all values of δ in the given range for the given k, the algorithm cannot pack either of these items, resulting in the target competitive ratio.

Case 2.2: An algorithm rejects $\frac{1}{c_k}$. The next items, up to and including the penultimate one, are presented to be of size $\frac{1}{c_k}$ as well. If an algorithm packs one of these items, Case 2.1 applies. We thus assume that all these items are rejected. The last item is then presented to be of size $\frac{1-\frac{1}{2+k}}{\frac{1+\delta}{1-\delta}}$.

If the algorithm does not pack this item, it is not competitive. An optimal solution can pack all $k+1$ items of size $\frac{1}{c_i}$ if $\delta\geq\frac{k^2+k-1}{k^2+3k+3}$. The competitive ratio is then $\frac{k+1}{c_k}/\frac{1-\frac{1}{2+k}}{\frac{1+\delta}{1-\delta}}=c_k$. ◀

We continue with a lower bound superseding the previous ones for (relative) higher values of δ.

▶ **Theorem 15.** *For every $k\in\mathbb{N}$ and $-1-k+\sqrt{4k+k^2}\leq\delta$, there exists no algorithm solving the* OSKP *problem with a competitive ratio better than $c_k=\frac{(1+\delta)(\sqrt{-1+\delta^2}+\sqrt{-17-16\delta+\delta^2-16k})}{4\sqrt{-1+\delta^2}}$.*

Proof. Let $i_k=\frac{2\sqrt{-1+\delta^2}}{\sqrt{-1+\delta^2}+\sqrt{-17-16\delta+\delta^2-16k}}$, which is the solution to the equation $i_k+\frac{i_k}{\frac{1+\delta}{1-\delta}}=\frac{1}{c\,k}$ and $\varepsilon>0$. The algorithm is given the following prediction:

$$P=(i_k,\frac{i_k}{\frac{1+\delta}{1-\delta}}+\varepsilon,\underbrace{\frac{1-i_k}{\frac{1+\delta}{1-\delta}}+\varepsilon,\dots,\frac{1-i_k}{\frac{1+\delta}{1-\delta}}+\varepsilon}_{i\text{ many}})$$

Note that $\frac{1-i_k}{\frac{1+\delta}{1-\delta}}\leq\frac{1}{c_k}$ only if $\delta\geq-1-k+\sqrt{4k+k^2}$.

We do a full case distinction on the possible behaviors of an algorithm. The first item is presented as i_k.

Case 1: An algorithm packs i_k. The next item is revealed as $i_k/\frac{1+\delta}{1-\delta}+\varepsilon$.

Case 1.1: An algorithm packs $i_k/\frac{1+\delta}{1-\delta}+\varepsilon$. The next item is revealed as $1-i_k$, thus fitting together with the first item, but not into the knapsack of the algorithm. All subsequent items are presented as $1-i_k$ as well. The optimal packing is thus of size 1, while the algorithm only achieves a gain of $i_k+i_k/\frac{1+\delta}{1-\delta}+\varepsilon=\frac{1}{c_k}+\varepsilon$.

Case 1.2: An algorithm rejects $i_k/\frac{1+\delta}{1-\delta}+\varepsilon$. All subsequent items are each revealed to be of size $1-i_k+\varepsilon$, thus not fitting into the knapsack of the algorithm. An optimal strategy packs the items $i_k/\frac{1+\delta}{1-\delta}+\varepsilon$ and $1-i_k+\varepsilon$, while the algorithm only has the first item of size i_k in its knapsack, resulting in a ratio of $(\frac{i_k}{\frac{1+\delta}{1-\delta}}+1-i_k+2\varepsilon)/(i_k) > c_k$.

Case 2: An algorithm rejects i_k. The second item is revealed of size $i_k+\varepsilon$.

Case 2.1: An algorithm packs $i_k+\varepsilon$. The remaining items are each revealed to be of size $1-i_k$, thus not fitting into the knapsack of the algorithm. An optimal packing is the first item together with one of the last items, adding up to a gain of 1. The algorithm can only pack $i_k+\varepsilon$, resulting in a ratio of $\frac{1}{i_k} \geq c_i$.

Case 2.2: An algorithm rejects $i_k+\varepsilon$. The next item is revealed of size $\frac{1}{c_k}$.

Case 2.2.1: An algorithm packs $\frac{1}{c_k}$. The remaining items are revealed of size $1-i_k$, not fitting into the knapsack of the algorithm but fully filling an optimal knapsack together with the first item, resulting exactly in a ratio of c_k.

Case 2.2.2: An algorithm rejects $\frac{1}{c_k}$. Up to and including the penultimate item, the next items are also revealed as $\frac{1}{c_k}$. If an algorithm packs any of them, the rest of the instance behaves as in Case 2.2.1.

Assuming this is not the case, the last item is presented as $(1-i_k)/(\frac{1+\delta}{1-\delta})+\varepsilon$, which is the only item the algorithm can pack. An optimal algorithm packs all items except the last one. These items are of sum at most one if $\delta \geq -1-k+\sqrt{4k+k^2}$. The resulting competitive ratio is

$$\frac{2i_k+\varepsilon+(k-1)\frac{1}{c_k}}{\frac{1-i_k}{\frac{1+\delta}{1-\delta}}+\varepsilon},$$

which converges to c_k for ε going to 0. ◀

Just like with the two previous bounds, the last lower bound dominates the other two for yet (relative) higher values of δ.

▶ **Theorem 16.** *For every $k \in \mathbb{N}_{>1}$ and $\frac{k-1}{k+1} < \delta < \frac{k}{k+2}$, there exists no algorithm solving the* OSKP *problem with a competitive ratio better than $c_k = \frac{1+k}{2} - \frac{\sqrt{(-1+\delta)(-5-k(2+k)+\delta(-1+k)(3+k))}}{-2+2\delta}$.*

Proof. Let $i_k = \frac{2(\delta-1)}{3\delta-\sqrt{-(\delta-1)(-4\delta(k-1)+k^2+2k+5)}+\delta k-k-3}$ be the solution to the equation $c_k = \frac{1}{i_k}-1$ and let $j_k = \frac{i_k^2}{1-(k+2)i_k}$. The algorithm is given the following prediction:

$$P = (i_k+\varepsilon, \underbrace{j_k}_{i \text{ many}})$$

We do a full case distinction on the possible behaviors of an algorithm. The first item is presented as i_k.

Case 1: An algorithm packs $i_k + \varepsilon$. Each of the next items are presented of maximum possible size $\frac{1+\delta}{1-\delta} j_k = 1 - i_k$, which is the counterpart to the first item, except for the additional ε. Since $1 - i_k > \frac{1}{2}$, no two items fit into the knapsack in this case. The optimal solution thus packs an item of size $1 - i_k$. The competitive ratio is then $\frac{1-i_k}{i_k+\varepsilon} \overset{\varepsilon \to 0}{=} c_k$.

Case 2: An algorithm rejects $i_k + \varepsilon$. The next items is presented to be of size j_k.

Case 2.1: An algorithm packs j_k. If an algorithm packs such an item before the last item, each subsequent item is presented to be of almost maximum size $\frac{1+\delta}{1-\delta} j_k - \varepsilon$. For $\delta < \frac{k}{k+2}$ it holds that $j_k + \frac{1+\delta}{1-\delta} j_k - \varepsilon > 1$. Thus, the algorithm only packs an item of size j_k, while the optimal solution is 1, as such an item of almost maximum size perfectly fits together with the first item of the instance. The competitive ratio is then $\frac{1}{j_k} > c_k$, as $j_k < \frac{1}{c_k}$ for $\delta > \frac{1}{3}$.

Case 2.2: An algorithm rejects j_k. Up to and including the penultimate item, the next items are also revealed as j_k. If an algorithm packs any of them, the rest of the instance behaves as in Case 2.1.

Assuming this is not the case, the last item is presented as j_k, which is the only item the algorithm can pack. An optimal algorithm packs all items except the last one. These items are of sum at most 1 if $\delta > \frac{1}{3}$. The competitive ratio is then $\frac{i_k+\varepsilon+kj_k}{j_k} \overset{\varepsilon \to 0}{=} c_k$. ◀

7 Open Problems

The Oskp problem turns out to be very involved problem. While we were able to give tight bounds on the competitive ratio for some values of δ, the analysis is not complete. We are quite confident that the lower bounds that we determined should reflect the actual competitive ratio in the remaining open segments, but finding a proof to do so is beyond our capability.

A logical next step would be to look at the general Online Knapsack problem in the setting of predictions. However, one would first have to determine which part of the input is predicted: The size, the weight or the density of the items. The model of predictions offers to be applied to – and already has been applied to – further problems beyond knapsack problems. To our knowledge, more discretized problems, such as graph problems like the Minimum Vertex Cover problem have not been studied in the setting of predictions. Finding an appropriate, unifying model on what to predict in such problems would be interesting.

References

1 Spyros Angelopoulos, Shahin Kamali, and Kimia Shadkami. Online bin packing with predictions. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4574–4580. ijcai.org, 2022. `doi:10.24963/ijcai.2022/635`.

2 Antonios Antoniadis, Christian Coester, Marek Eliáš, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 345–355. PMLR, 2020. URL: `http://proceedings.mlr.press/v119/antoniadis20a.html`.

3 Antonios Antoniadis, Christian Coester, Marek Eliáš, Adam Polak, and Bertrand Simon. Mixing predictions for online metric algorithms. *CoRR*, abs/2304.01781, 2023. `doi:10.48550/arXiv.2304.01781`.

4 David Arthur, Bodo Manthey, and Heiko Röglin. k-means has polynomial smoothed complexity. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 405–414. IEEE Computer Society, 2009. `doi:10.1109/FOCS.2009.14`.

5 Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1070–1080. ACM, 2021. `doi:10.1145/3406325.3451023`.

6 Yossi Azar, Eldad Peretz, and Noam Touitou. Distortion-oblivious algorithms for scheduling on multiple machines. In Sang Won Bae and Heejin Park, editors, *33rd International Symposium on Algorithms and Computation, ISAAC 2022, December 19-21, 2022, Seoul, Korea*, volume 248 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.ISAAC.2022.16`.

7 Eric Balkanski, Vasilis Gkatzelis, and Xizhi Tan. Strategyproof scheduling with predictions. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 11:1–11:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPIcs.ITCS.2023.11`.

8 Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, Guido Schäfer, and Tjark Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback algorithm. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 462–471. IEEE Computer Society, 2003. `doi:10.1109/SFCS.2003.1238219`.

9 René Beier and Berthold Vöcking. Random knapsack in expected polynomial time. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 232–241. ACM, 2003. `doi:10.1145/780542.780578`.

10 Magnus Berg, Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Online minimum spanning trees with weight predictions. *CoRR*, abs/2302.12029, 2023. `doi:10.48550/arXiv.2302.12029`.

11 Hans-Joachim Böckenhauer, Elisabet Burjons, Juraj Hromkovic, Henri Lotze, and Peter Rossmanith. Online simple knapsack with reservation costs. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.STACS.2021.16`.

12 Hans-Joachim Böckenhauer, Dennis Komm, Richard Královic, and Peter Rossmanith. The online knapsack problem: Advice and randomization. *Theor. Comput. Sci.*, 527:61–72, 2014. `doi:10.1016/j.tcs.2014.01.027`.

13 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

14 Joan Boyar, Lene M. Favrholdt, Shahin Kamali, and Kim S. Larsen. Online interval scheduling with predictions. *CoRR*, abs/2302.13701, 2023. `doi:10.48550/arXiv.2302.13701`.

15 Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Online unit profit knapsack with untrusted predictions. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2022, June 27-29, 2022, Tórshavn, Faroe Islands*, volume 227 of *LIPIcs*, pages 20:1–20:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.SWAT.2022.20`.

16 Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 10393–10406, 2021. URL: `https://proceedings.neurips.cc/paper/2021/hash/5616060fb8ae85d93f334e7267307664-Abstract.html`.

17 Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. Learning-augmented query policies for minimum spanning tree with uncertainty. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 49:1–49:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.49.

18 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, December 8-11, 2019, Shanghai University of Finance and Economics, Shanghai, China*, volume 149 of *LIPIcs*, pages 28:1–28:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.28.

19 Xin Han and Kazuhisa Makino. Online removable knapsack with limited cuts. *Theor. Comput. Sci.*, 411(44-46):3956–3964, 2010. doi:10.1016/j.tcs.2010.08.009.

20 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Online knapsack with frequency predictions. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 2733–2743, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/161c5c5ad51fcc884157890511b3c8b0-Abstract.html.

21 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, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 293–305. Springer, 2002. doi:10.1007/3-540-45465-9_26.

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

23 Billy Jin and Will Ma. Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model. In *NeurIPS*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/5d68a3f05ee2aae6a0fb2d94959082a0-Abstract-Conference.html.

24 Dennis Komm. *An Introduction to Online Computation - Determinism, Randomization, Advice*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016. doi:10.1007/978-3-319-42749-2.

25 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1859–1877. SIAM, 2020. doi:10.1137/1.9781611975994.114.

26 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3302–3311. PMLR, 2018. URL: http://proceedings.mlr.press/v80/lykouris18a.html.

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

28 Michele Monaci, Ulrich Pferschy, and Paolo Serafini. Exact solution of the robust knapsack problem. *Comput. Oper. Res.*, 40(11):2625–2631, 2013. doi:10.1016/j.cor.2013.05.005.

29 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update rules. In Richard A. DeMillo, editor, *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 488–492. ACM, 1984. doi:10.1145/800057.808718.

30 Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 296–305. ACM, 2001. `doi:10.1145/380752.380813`.

31 Clemens Thielen, Morten Tiedemann, and Stephan Westphal. The online knapsack problem with incremental capacity. *Math. Methods Oper. Res.*, 83(2):207–242, 2016. `doi:10.1007/s00186-015-0526-9`.

32 Chenyang Xu and Guochuan Zhang. Learning-augmented algorithms for online subset sum. *J. Glob. Optim.*, 87(2):989–1008, 2023. `doi:10.1007/S10898-022-01156-W`.

33 Yunhong Zhou, Deeparnab Chakrabarty, and Rajan M. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In Christos H. Papadimitriou and Shuzhong Zhang, editors, *Internet and Network Economics, 4th International Workshop, WINE 2008, Shanghai, China, December 17-20, 2008. Proceedings*, volume 5385 of *Lecture Notes in Computer Science*, pages 566–576. Springer, 2008. `doi:10.1007/978-3-540-92185-1_63`.

A Additional Proofs

Proof of Theorem 12. If the algorithm ends after the call of *filter_trivial*, the algorithm is at most c-competitive. Assuming the condition in line 3 is met, the algorithm is at least c-competitive by Lemma 8. Thus, we only have to prove that the algorithm is not worse than c-competitive if it ends in line 6.

Let us first assume that $x_1 \neq b$. Then x_1 is packed and $x_1 > 1 - \frac{1+\delta}{1-\delta}b^-$. If any other large item x_i can be packed by the algorithm, the knapsack will be filled up to at least $\frac{1}{2}$, since

$$x_1 + x_i > 1 - \frac{1+\delta}{1-\delta}b^- + \frac{1-\delta}{1+\delta}b^- > 1 - \frac{1+\delta}{1-\delta}\frac{1}{c} + \frac{1-\delta}{1+\delta}\frac{1}{c} \geq \frac{1}{c}\,,$$

where the last inequality holds for all values of δ between 0 and 1.

If no second large item fits into the algorithm's knapsack, an optimal solution cannot contain more than one large item either. This item of the optimal solution is bounded by $\frac{1-\delta}{1+\delta}b^- < \frac{1-\delta}{1+\delta}\frac{1}{c}$. The remaining items of an optimal solution then consist of all small items that the algorithm has ignored. We can bound the total announced size of these small items, by $\frac{1}{c} - b^-$, using Lemma 6 and in consequence their actual size by $\frac{1+\delta}{1-\delta}(\frac{1}{c} - b^-)$. Putting it all together, in the worst case Algorithm 2 packs exactly one large item of size slightly larger than $1 - \frac{1+\delta}{1-\delta}b^-$. The optimal solution packs one large item of size slightly smaller than $\frac{1-\delta}{1+\delta}\frac{1}{c}$ and the maximum sum of small items. This results in a competitive ratio of

$$\frac{\frac{1+\delta}{1-\delta}b^- + \frac{1+\delta}{1-\delta}(\frac{1}{c} - b^-)}{1 - \frac{1+\delta}{1-\delta}b^-} = \frac{\frac{1+\delta}{1-\delta}\frac{1}{c}}{1 - \frac{1+\delta}{1-\delta}b^-} < \frac{\frac{1+\delta}{1-\delta}\frac{1}{c}}{1 - \frac{1+\delta}{1-\delta}\frac{1}{c}} = c\,,$$

for $\delta \leq \frac{1}{7}$.

The only case left to handle is that $x_1 = b$. If $b \geq 1 - \frac{1+\delta}{1-\delta}b^-$, we can use the same argumentation as before. If however $b < 1 - \frac{1+\delta}{1-\delta}b^-$, then, since b is by definition the announced largest item, *all* large items are announced smaller than $1 - \frac{1+\delta}{1-\delta}b^-$ and thus of actual size at most $\frac{1+\delta}{1-\delta}(1 - \frac{1+\delta}{1-\delta}b^-) < \frac{1+\delta}{1-\delta}(1 - \frac{1+\delta}{1-\delta}\frac{1}{c}/\frac{1+\delta}{1-\delta}) = \frac{1+\delta}{1-\delta}\frac{1}{c}$. Since no large item can exceed an actual size of $\frac{1+\delta}{1-\delta}b^-$ and since $x_1 < 1 - \frac{1+\delta}{1-\delta}b^-$, we are guaranteed that a second large item fits into the knapsack of the algorithm. In the worst case, an optimal solution packs b and an item of size $\frac{1+\delta}{1-\delta}b^-$, while the algorithm only packs b and a large item of minimal size. The competitive ratio is then

$$(\frac{1+\delta}{1-\delta}b^- + b^-)/(b^- + \frac{1-\delta}{1+\delta} - b^-) < c\ ,$$

where the last inequality holds for all values of δ between 0 and 1.

The only case left is that there is no second large item. Then the algorithm and the optimal solution both pack the same large item $x_1 = b$, while the optimal solution can still add the maximum number of small items. The resulting competitive ratio is then $\frac{b^- + \frac{1+\delta}{1-\delta}(\frac{1}{c} - b^-)}{b^-} < c$ for all δ between 0 and 1. ◀