Online Knapsack Problems with Estimates

Jakub Balabán
Masaryk University, Brno, Czech Republic

Matthias Gehnen
RWTH Aachen University, Germany

Henri Lotze
RWTH Aachen University, Germany

Finn Seesemann
RWTH Aachen University, Germany

Moritz Stocker
ETH Zürich, Switzerland

Abstract

Imagine you are a computer scientist who enjoys attending conferences or workshops within the year. Sadly, your travel budget is limited, so you must select a subset of events you can travel to. When you are aware of all possible events and their costs at the beginning of the year, you can select the subset of the possible events that maximizes your happiness and is within your budget. On the other hand, if you are blind about the options, you will likely have a hard time when trying to decide if you want to register somewhere or not, and will likely regret decisions you made in the future. These scenarios can be modeled by knapsack variants, either by an offline or an online problem. However, both scenarios are somewhat unrealistic: Usually, you will not know the exact costs of each workshop at the beginning of the year. The online version, however, is too pessimistic, as you might already know which options there are and how much they cost roughly. At some point, you have to decide whether to register for some workshop, but then you are aware of the conference fee and the flight and hotel prices.

We model this problem within the setting of online knapsack problems with estimates: in the beginning, you receive a list of potential items with their estimated size as well as the accuracy of the estimates. Then, the items are revealed one by one in an online fashion with their actual size, and you need to decide whether to take one or not. In this article, we show a best-possible algorithm for each estimate accuracy δ (i.e., when each actual item size can deviate by $\pm\delta$ from the announced size) for both the simple knapsack (also known as subset sum problem) and the simple knapsack with removability.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Knapsack, Online Knapsack, Removability, Estimate, Prediction

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.12

Related Version *Previous Version*: `https://arxiv.org/abs/2504.21750`

Funding *Jakub Balabán*: Brno Ph.D. Talent Scholarship Holder – Funded by the Brno City Municipality.
Matthias Gehnen: Funded by RWTH Research ENHANCE.R.

1 Introduction

In the ONLINE (SIMPLE) KNAPSACK problem, items are presented one after another to an algorithm, which then has to decide on the spot whether to include the current item into the knapsack of unit size, or not. In the classical setting, it is easy to see that this problem is non-competitive: A tiny item can be presented at the beginning and an algorithm cannot know whether it should pack this item (as it might be the only one) or reject it (as there might be an item of size 1 next, which the algorithm cannot pack when having packed the tiny item). Therefore, the ratio between a solution of an online algorithm and an optimal (offline) solution on the same instance (the so-called *competitive ratio*) can be arbitrarily high.

Arguably, one can say that this counterexample is rather pathological, as the restriction for deterministic, irrevocable decisions in a setting without any knowledge about the future is harsh. So even though this (negative) example by Marchetti-Spaccamela and Versallis [29] might seem demotivating for further research at first glance, it indeed has resulted in even more attention to online knapsack problems and their variants. For example, by allowing (limited) decisions to be revoked or delayed, or by relaxing the complete blindness of an online algorithm, many of those modified settings aim to avoid pathological counterexamples and therefore to model real-world knapsack applications more realistic. We will present some of those variants in the following section. In several cases, a variant studied for knapsack problems later turned out to be realistic for other online problems. For a more thorough introduction to competitive analysis, we refer to the books by Borodin and El-Yaniv [13] and by Komm [27].

When acting in a real-world setting, the assumption of complete blindness is as unrealistic as the assumption of having correct information available: When navigating to some place, one usually is not aware of every detail (e.g., a traffic jam that might occur) but still has some information about the structure of the map and how long which routes roughly will take. The details then get revealed when arriving at some street, and seeing how the situation is. Or, when considering a packing problem such as bin packing or knapsack, one usually does not know the exact sizes of all items that are in the instance - but by experience, one might have a rough idea about, e.g., the items that need to be distributed into moving boxes for a moving - even though one likely does not want to measure all of them before packing to ensure that a packing is optimal in the end. Even if one has all the necessary information of some problem measured and, in theory, could compute an optimal solution, the exact values likely get lost when saving them on memory with limited size at the latest.

In a sense, the setting of online problems with estimates lies in between offline problems with complete knowledge about the instance and classical online problems without any information about the future.

For the online (simple) knapsack problem, we assume an algorithm is given a list of items and their estimated size, together with a constant δ as the estimate accuracy. Each item with its actual size then gets revealed one after another, like in the classical online knapsack setting. Each item size can deviate by up to δ from their estimate. An algorithm then has to decide about whether to pack an item or not:

In the case of irrevocable decisions, we show that no algorithm can achieve a competitive ratio better than $\frac{1}{\min(p,q)}$, with

$$p = -\frac{0.5}{\lfloor k \rfloor} + \sqrt{\frac{1}{4\lfloor k \rfloor^2} + \frac{1-2\delta}{\lfloor k \rfloor}} \text{ and } q = 1 - 2\delta - \frac{1}{\lceil k \rceil}\text{ , where } k = \frac{2}{1-2\delta}.$$

for all $0 < \delta < \frac{1}{2}$ using three different instance constructions for different δ. For a $\delta \geq \frac{1}{2}$, we show that the problem becomes uncompetitive. For $\delta < \frac{1}{2}$, where competitive algorithms can exist, we also present an algorithm that matches our lower bound and prove the tightness of of the behavior.

In the second part of the article, we consider the online simple knapsack problem with removability. Here an item, that was packed by an algorithm, can also be discarded at a later point. This very classical online knapsack variant avoids the pathological counterexample mentioned earlier: the tiny item can be packed without concerns first, and just be replaced by an item of size 1 if one is presented in the instance. This setting was found to be Φ-competitive by Iwama and Taketomi [24], with Φ denoting the golden ratio. In this case, we show that no algorithm can achieve a better competitive ratio than $\frac{3-2\delta}{2-2\delta}$ for all $0 < \delta < \frac{3}{4} - \frac{\sqrt{5}}{4}$, and also present an algorithm that achieves this ratio as a tight upper bound. For all $\delta \geq \frac{3}{4} - \frac{\sqrt{5}}{4}$, it turns out that the estimates will not help any longer, therefore the best possible algorithm is given by Iwama and Taketomi and achieves a competitive ratio of Φ.

Finally, we conclude with some final remarks and open questions that arose while working in this knapsack setting.

1.1 Online Knapsack Problems

One way to deal with those artificial lower bounds is to allow an algorithm to fill a knapsack in a way a classical knapsack algorithm is not allowed to do: A variant where the online algorithm is allowed to overpack the knapsack slightly is called *resource augmentation* by Iwama and Zhang [25]. Han and Makino [22] allowed a limited number of *cuts*, i.e. splitting of items into two sub-items.

Another approach is to delay decisions for some time. We already mentioned a variant introduced by Iwama and Taketomi, in which packed items could be removed (also called *preempted*) from the knapsack, but not to be packed again [24]. Another variant allows an algorithm to intermediately store items in a *buffer* of a certain size, as introduced by Han et al. [21]. The recently introduced model of *reservation costs* allows an algorithm pay a fee to delay a decision for an arbitrary amount of time [10, 16].

Other variants allow the knapsack to grow over time, as in Thielen, Tiedemann, and Westphal [34], where a model in which the capacity of the knapsack increases step-wise over a given number of periods is studied.

It is also possible to avoid artificial instances by restricting the instance classes as in Zhou, Chakrabarty and Lukose [37], where they 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.

Further variations include randomization or an oracle to communicate information about the instance via so-called *advice bits* as introduced by Böckenhauer et al. [11, 12].

One of the main criticisms of the advice model is that the existence of an almighty oracle is not realistic in practice. One way to deal with this setting is the model of *machine learned advice*, which has recently been called *untrusted predictions* or just *predictions*. Here an algorithm is usually given a prediction on each piece of the input, which tells the algorithm what to do with the piece of input. However, these predictions might be wrong and no bound is given on the error. Therefore, the goal of an algorithm in this setting is to deal with those possibly wrong hints, and ideally compute an optimal solution in case the predictions are correct (*consistency*), but also performs as well as a regular online algorithm on the problem when the predictions become arbitrarily bad (*robustness*) and ideally degrades with increasing unreliability of the prediction (*smoothness*).

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 [28, 14, 7], metric algorithms [3, 2], matching problems [17, 26], spanning tree problems [18, 9].

Im et al. [23] recently looked at the general knapsack problem, which they studied 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] 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 [35] recently studied the simple knapsack problem in a learning-augmented setting, where they design algorithms that can learn and use the error of prediction. Zeynali et al. [36] also studied influence of predictions with real life data with which machine-learning tools have been learned.

Even though the prediction model feels aligned with the research in this work, the focus lies on different aspects of inaccuracy: In the prediction setting, the measure of accuracy is commonly defined by the number of correct and wrong hints. In the setting of this article, we assume that the whole input might not be as announced, but still lies in some surrounding of the announcement (which size defines the accuracy).

While we assume that an adversary can 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 without 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 [33]. Since then, there has been a large influx of results in this area for a wide range of problems, for example, the 0/1 knapsack problem [8]. 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 - just as our pathological counterexample for the online simple knapsack.

Furthermore, our model is related to robust optimization: The *robust knapsack problem* by Monaci, Pferschy and Serafini [30] is very similar in that it also allows 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.

1.2 Online Problems with Estimates

The setting of online problems with estimates, where first a rough idea of the instance is given, and then the actual values gets revealed in an online manner, is rather new in online computation.

Azar et al. considered a scheduling variant where the size of a job presented upon its revelation might not be given exactly [4]. Here, they studied algorithms for the scenario where the accuracy of the estimate is known to the algorithm, as well as the scenario where this is not the case [5]. These results later got extended to a setting with multiple machines [6]. While in these cases (as in our article) the estimates are given adversarially, Scully et al. [31] analyzed a setting where the actual durations of each job were picked randomly instead. Azar et al. also speak of *problems with predictions* within their works.

To avoid confusion with the previously mentioned model of predictions, which usually refers to a setting where suggestions are given which might be wrong, we name this specific type of prediction *estimate* in the context of this work.

As packing problems are natural for a setting with estimated item sizes, we see that the knapsack setting with multiplicative accuracy behaves differently than the additive setting [20]: For example, while also not achieving a competitive ratio better than 2 even for very small δ, in the multiplicative setting an algorithm can achieve this ratio even up to a $\delta = \frac{1}{7}$. Furthermore, the divergence did not start at $\frac{1}{2}$ but at 1. This different behavior is mainly caused by the fact that in the multiplicative setting, an adversary cannot present an item with the size of 0 when the given estimate allowed also allowed to present the item as a non-zero item.

Apart from scheduling and packing problems, situations where the details are only revealed over time can also occur on graph problems. For the traveling salesman/graph exploration problem, it was shown that no algorithm can achieve a competitive ratio better than the estimate accuracy when the algorithm is aware of the graph structure with the estimated edge weights in the beginning, and the exact values are only revealed in a graph-exploration manner [19].

1.3 Formal Definition

To avoid confusion, we start with a formal definition of the problems we investigate and of the competitive ratio in the version that is referred to for the results. In this article, notation will be slightly abused by using x_i for both the label of an item and the size of the same item, as its meaning is also clear in the given context.

▶ **Definition 1** (The Online Simple Knapsack with Item Size Estimates Problem). *Given an estimate accuracy $\delta \geq 0$, an instance of the* Online Simple Knapsack with Item Size Estimate *consists of a series of* items $I = (x_1, \dots, x_n)$*, where each* item *is a real number in* $[0, 1]$*. At the beginning, the accuracy δ is announced as well as the* item size estimates $P = (x'_1, \dots, x'_n)$*, such that for each $i \in \{1, \dots, n\}$, $x'_i - \delta \leq x_i \leq x'_i + \delta$. At each step $i \in \{1, \dots, n\}$, the actual item size x_i of the request sequence gets revealed. An algorithm then has the option to either pack the item in an initially empty knapsack K, if it fits or to reject the item:*

- ***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 Item Size Estimates *problem is then for an algorithm* `ALG` *to minimize the competitive ratio between the size of his packing, compared to the optimal solution in the same instance.*

The online simple knapsack with removability and item size estimates can be defined analogously:

▶ **Definition 2** (The Online Simple Knapsack with Removability and Item Size Estimates Problem). *Given an estimate accuracy $\delta \geq 0$, an instance of the* Online Simple Knapsack with Removability and Item Size Estimate *consists of a series of* items $I = (x_1, \dots, x_n)$*, where each* item *is a real number in* $[0, 1]$*. At the beginning, the accuracy δ is announced as well as the* item size estimates $P = (x'_1, \dots, x'_n)$*, such that for each $i \in \{1, \dots, n\}$, $x'_i - \delta \leq x_i \leq x'_i + \delta$. At each step $i \in \{1, \dots, n\}$, the actual item size x_i of the request sequence gets revealed. An algorithm then has the option to remove a subset of items from the initially empty knapsack K, and then to either pack the current item into K, if it fits, or to reject the item:*

- ***Remove*** *Discard $S \subseteq K$ from the knapsack, set $K := K - S$.*
- ***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 Removability and Item Size Estimates *problem is then for an algorithm* `ALG` *to minimize the competitive ratio between the size of his packing, compared to the optimal solution in the same instance.*

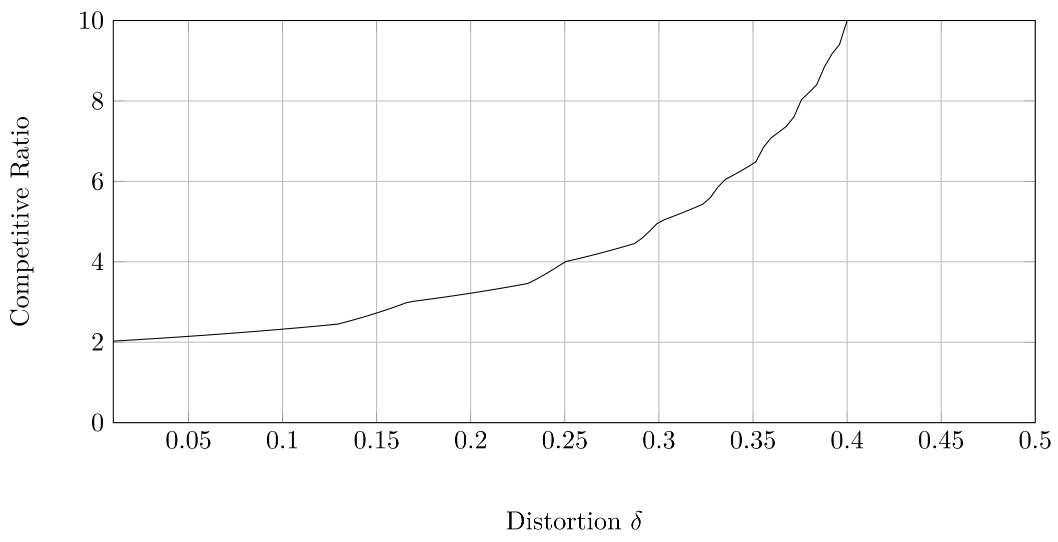

Figure 1 Competitive ratio in the *absolute error* model, depending on δ.

Online algorithms are generally analyzed using competitive analysis, as introduced by Sleator and Tarjan [32]. Simply speaking, it compares the solution quality of an online algorithm to the solution of an optimal offline algorithm in the same instance. As the knapsack problems are maximization problems, the following definition is suitable:

▶ **Definition 3.** *The* (strict) competitive ratio *of an online algorithm A is the highest ratio of any request sequence S between the gain of A on S and the gain of an algorithm* OPT *solving the problem optimally on the same request S,*

$$\mathrm{CR}(\mathtt{A}) = \sup_S \left\{ \frac{\mathrm{gain}_{\mathtt{OPT}}(S)}{\mathrm{gain}_{\mathtt{A}}(S)} \right\} .$$

In other settings, a non-strict variant of the competitive ratio is commonly used as well, where an additional constant k is allowed when comparing the solution quality of an online algorithm to the offline solution. Furthermore, a similar variant can also used to define the competitive ratio for minimization problems.

2 Online Simple Knapsack

In this section, we first show that the competitive ratio of the online simple knaspack problem with item size estimates is not less than 2 for any $\delta > 0$ and monotonously rises until it gets unbounded for any algorithm for $\delta \geq 0.5$.

Afterwards, we provide an algorithm that matches the competitive ratio of the lower bound.

We will see that the competitive ratio for all $0 < \delta < \frac{1}{2}$ is given by $\frac{1}{\min(p,q)}$, where

$$p = -\frac{0.5}{\lfloor k \rfloor} + \sqrt{\frac{1}{4\lfloor k \rfloor^2} + \frac{1-2\delta}{\lfloor k \rfloor}} \text{ and } q = 1 - 2\delta - \frac{1}{\lceil k \rceil} \text{ , where } k = \frac{2}{1-2\delta}. \tag{1}$$

This ratio is visualized in Figure 1.

Note that p is the positive solution of the following equation.

$$\left\lfloor \frac{2}{1-2\delta} \right\rfloor \cdot \frac{p}{1-p-2\delta} = \frac{1}{p} \tag{2}$$

The following inequalities, which hold for $0 \leq \delta \leq 0.5$, will turn out to be useful:

$$1-p-2\delta \leq p \text{ and } 1-q-2\delta = \left\lceil \frac{2}{1-2\delta} \right\rceil^{-1} \leq q \tag{3}$$

2.1 Lower Bounds

The following two lower bounds combined are best possible for all values of δ up to $\frac{1}{2}$. Both functions have jumps induced by rounding effects and dominate one another periodically. See Equation (1) for the definition of p and q. In the end, we note that for $\delta \geq \frac{1}{2}$, no constant competitive online algorithm can exist.

While the construction for the first bound is slightly more involved, it defines a bound for the whole range of δ for $0 < \delta < \frac{1}{2}$. The second bound, as defined in Theorem 5, however only works starting from $\delta \geq \frac{1}{6}$. For values smaller than $\frac{1}{6}$, a different construction as in Theorem 6 is necessary.

The crucial case in the following proof is Case 1.2, where the knapsack of the optimal solution can be forced by a clever algorithm not to be completely filled.

▶ **Theorem 4.** *For every $0 < \delta < 0.5$, there exists no algorithm solving the* OSKE *problem with a competitive ratio better than $\frac{1}{p}$.*

Proof. Let $0 < \varepsilon < \min(p, \delta)$ such that $1/\varepsilon \in \mathbf{N}$, and let $k = \left\lfloor \frac{2}{1-2\delta} \right\rfloor$. Let us consider an arbitrary algorithm for the OSKE problem. The following instance is announce as a prediction:

$$P = \left(\underbrace{\varepsilon,}_{1/\varepsilon \text{ many}} \ \underbrace{\frac{1}{2},}_{k \text{ many}} \ 1-p-\delta \right)$$

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

Case 1: The algorithm packs ε. The subsequent $1/\varepsilon - 1$ items are presented as 0. The next item is revealed with size p. This is possible as $p+\delta \geq 1/2$, which follows from Equation (3).

Case 1.1: The algorithm packs p. All remaining items are presented as $1-p$, which the algorithm can not pack due to the ε-item. As we have seen, it is possible that an item which was announced of size $1/2$ can be presented as p; therefore presenting those items as $1-p$ is possible as well due to the symmetry of the deviation.

An optimal solution can pack both, p and its counterpart, while the algorithm has packed p and an item of size ε. The competitive ratio is then $1/p$ for ε converging to 0.

Case 1.2: The algorithm rejects p. The subsequent $k-1$ items will be presented as p if the algorithm continues to reject them. If an algorithm should pack any of these items, we can use the same argumentation as in Case 1.1.

Should the algorithm reject all k items of size p, the last item is presented as $1 - p - 2\delta$. The optimal solution consists of all k items of size p, whereas the algorithm only has the item of size ε and the last item of size $1 - 2\delta - p$ in its knapsack. Since $1 - p - 2\delta \leq p$ by Equation (3) and $kp/(1 - p - 2\delta) = 1/p$ by Equation (2) is valid, the competitive ratio is at least $1/p$ for ε converging to 0.

Case 2: The algorithm rejects ε. The subsequent $1/\varepsilon - 1$ items will be presented as ε if the algorithm continues to reject them. If the algorithm should pack any of these items, the remaining items are presented as 0 and we use the same argumentation as in Case 1.

Should the algorithm reject all $1/\varepsilon - 1$ items of size ε, the subsequent k items will be presented as $p + \varepsilon$. If such an item is packed, the argumentation is the same as in case 1.1, just with the difference that the optimal solution contains the ε-items with a total size of 1.

If no such item is packed, the last item is presented of size $1 - 2\delta - p \leq p$ (see Equation (3)). The optimal solution packs the full knapsack using items of size ε. Again, the competitive ratio is at least $\frac{1}{p}$. ◀

The following theorem gives a better bound than Theorem 4 when $q < p$ happens. Note that $q > p$ is true for $\frac{1}{6} < \delta < \frac{1}{24}(9 - 2\sqrt{3}) \approx 0.23$ and the relation $\frac{3}{16} \in [\frac{1}{6}, 0.23]$ holds. Consequently, after Theorem 5, it remains to inspect the lower q-bound for $\delta < 1/6$.

▶ **Theorem 5.** *For every $\frac{3}{16} < \delta < 0.5$, there exists no algorithm solving the* Oske *problem with a competitive ratio better than $\frac{1}{q}$.*

Proof. Let $\delta \leq \varepsilon > 0$ such that $1/\varepsilon \in \mathbf{N}$, let $k = \left\lceil \frac{2}{1-2\delta} \right\rceil$, and recall that $q = 1 - 2\delta - 1/k$. Let us consider an arbitrary algorithm for the Oske problem. The following prediction is announced to the algorithm:

$$P = \left(\underbrace{\varepsilon,}_{1/\varepsilon \text{ many}} \quad \underbrace{\delta,}_{k \text{ many}} \quad q + \delta \right)$$

We do a case distinction on the potential behaviors of the algorithm. The first item is presented as ε.

Case 1: The algorithm packs ε. The subsequent $1/\varepsilon - 1$ items are presented as 0. The next presented item is of size $1/k$. This is possible if $2\delta \geq q$ since Equation (3) yields $q \geq 1/k$. The former holds for $\delta \geq 3/16$.

Case 1.1: The algorithm packs $1/k$. The remaining $k - 1$ items are presented as 0, with the last item presented as $1 - 1/k = q + 2\delta$. An optimal solution can pack both $1/k$ and its counterpart, while the algorithm has packed $1/k$ and an item of size ε. The competitive ratio is then k for ε converging to 0 which yields the wished ratio by $k \geq 1/q$ tanks to Equation (3).

Case 1.2: The algorithm rejects $1/k$. The subsequent $k - 1$ items will be presented as $1/k$, with the last item presented as q, if the algorithm continues to reject them. If an algorithm should pack any of these items, we can use the same argumentation as in Case 1.1.

The optimal solution consists of all k items of size $1/k$, which add up to exactly 1, whereas the algorithm only has the item of size ε and the last item q in its knapsack. The competitive ratio is again at least $1/q$ for ε converging to 0.

Case 2: The algorithm rejects ε. The subsequent $1/\varepsilon - 1$ items will be presented as ε as long as they get rejected. If an algorithm packs any of these items, we can use the same argumentation as in Case 1.

Should the algorithm reject all $1/\varepsilon - 1$ items of size ε, the subsequent k items will be presented as 0. The last item is presented as q which directly yields the competitive ratio of $1/q$. Here an optimal solution consists of a full knapsack with ε items. ◀

The crucial issue of Theorem 5 for small δ are the items of announced size δ, where an adversary must be able to present them as a 0 or as $\frac{1}{3}$ for all $\delta < \frac{1}{6}$. As this is not possible, we need to handle the setting $\delta < \frac{1}{6}$ as a special case. In particular, $q > p$ for $0 < \delta < \frac{1}{12}(4 - \sqrt{6}) \approx 0.129$ is true, and since $1/12 < 0.129$ holds, starting with delta at $1/12$ is no limitation.

▶ **Theorem 6.** *For every $\frac{1}{12} < \delta < \frac{1}{6}$, there exists no algorithm solving the* Oske *problem with a competitive ratio better than $\frac{1}{q}$.*

Proof. We define $a = \frac{1}{3} - 2\delta$, and let $\varepsilon > 0$ be arbitrary such that $1/\varepsilon \in \mathbf{N}$ and $a/\varepsilon \in \mathbf{N}$ holds. Let us consider an arbitrary algorithm for the Oske problem. The algorithm receives the following prediction:

$$P = \left(\underbrace{\varepsilon,}_{1/\varepsilon \text{ many}} \quad \underbrace{\frac{1}{3} + \delta}_{3 \text{ many}} \right)$$

We do a similar full case distinction on the possible behaviors of the algorithm, like in Theorem 4 and Theorem 5. The first item is presented as ε, but this time we stop presenting ε-items when the algorithm has packed exactly an amount of y between a and $a + \varepsilon$.

Case 1: The algorithm packs an amount of y with ε-items. The subsequent ε-items are presented as 0. The next item is presented of size $1/3$.

Case 1.1: The algorithm packs $1/3$. The remaining 2 items are presented as $\frac{2}{3} - a = \frac{1}{3} + 2\delta$. An optimal solution can pack $1/3$ together with one large item and several ε-items such that it achieves a packing of at least $1 - \varepsilon$, while the algorithm has packed $1/3$ and some ε-items of total size less than $a + \varepsilon$. The competitive ratio is then $(2/3 - 2\delta)^{-1} = (1 - 1/3 - 2\delta)^{-1} = 1/q$ for ε converging to 0 which yields the wished competitive ratio.

Case 1.2: The algorithm rejects $1/3$. The subsequent 2 items will be presented as $1/3$ until one gets accepted. If an algorithm accepts one of the first two items, we can use the same argumentation as in Case 1.1.

Otherwise, the algorithm accepts the last $1/3$-item or none of them. The optimal solution consists of all 3 items of size $1/3$, which add up to exactly 1, whereas the algorithm again has only the ε-items of total size $a + \varepsilon$ and at most the last $1/3$ item in its knapsack. The calculation of the competitive ratio is analogous to case 1.1.

Case 2: The algorithm do not pack at least $y > a$ with ε-items. We presented all the ε-items as ε and we know after these items that the algorithm got at most $y \leq a$ in the knapsack.

Next, we start to reveal $1/3 + (a - y) + \varepsilon$ items. This is possible because of the relation

$$\frac{1}{3} + \varepsilon \leq \frac{1}{3} + (a - y) + \varepsilon \leq \frac{1}{3} + a + \varepsilon = \frac{2}{3} - 2\delta + \varepsilon \leq \frac{1}{3} + 2\delta \quad \text{for } \delta > \frac{1}{12}.$$

Should the algorithm pick one of these 3 items, we are in the same setting as in case 1.1, but with an optimal solution consisting of just ε-items. ◀

As the construction of Theorem 5 cannot work for small δ, it is also worth noting that the construction of Theorem 6 is not working for larger δ. This is mainly because the items which are announced with size $\frac{1}{3} + \delta$ must be presented of size $\frac{1}{3}$ at least. Even if the 3 would be replaced with some k, it would allow an algorithm to pack multiple of those items without a chance of an adversary to hinder him.

Finally, we see that the case $\delta \geq 0.5$ can be handled using a standard construction by Marchetti-Spaccamela and Vercellis [29], thus no algorithm can achieve a bounded competitive ratio here.

▶ **Theorem 7.** *For every $\delta \geq 0.5$, there exists no algorithm solving the* OSKE *problem with a constant competitive ratio.*

Proof. Consider an arbitrary algorithm and the prediction $P = (0.5, 0.5)$. The first item arrives as $\varepsilon > 0$. If the algorithm rejects it, the second item arrives as 0. Otherwise, the second item arrives as $1 - \varepsilon/2$ so the algorithm cannot pack it, whereas the optimum solution can. This construction shows that no algorithm can be competitive. ◀

2.2 Upper Bounds

We start this section with a simple algorithm, that either takes the largest announced item or packs the knapsack in a greedy way.

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

```
if $b$ is the largest announced item and $b \geq 0.5$ then
    Pack only $b$. END
else
    Greedily pack items. END
```

It turns out that this algorithm already matches the lower bound for all accuracies δ of the form $\frac{1}{2} - \frac{1}{k}$ for integers $k \geq 3$.

▶ **Theorem 8.** *Given a fixed δ with $0 < \delta < \frac{1}{2}$, Algorithm 1 solves the* OSKE *problem with a competitive ratio of at most $\frac{2}{1-2\delta}$.*

Proof. Assume that there exists an announced item of size at least 0.5. Then the algorithm waits for it, packs it, and achieves a gain of $0.5 - \delta$, which gives us the claimed bound.

Thus, assume that such an item does not exists but that there exists an item that the algorithm cannot fit into its knapsack when packing greedily. This item can be at most of actual size $0.5+\delta$, meaning our gap due to not packing this item is at most $1-(0.5+\delta) = 0.5-\delta$, which again gives us our wanted bound. ◀

In the rest of this section, we will present a more refined algorithm and prove that it matches the lower bounds of Theorems 4 and 5. Let us fix an instance (P, δ), where $0 < \delta < 0.5$ holds, and $P = (x'_1, \ldots, x'_n)$ are the announced item sizes of the OSKE problem. As in Definition 1, if x is an item, then x' denotes its announced size. Let $c = 1/\min(p, q)$ and $\overline{c} = 1/c$, see Equation (1) for the definition of p and q.

It is easy to see that the algorithm achieves the desired competitive ratio in many instances.

▶ **Lemma 9.** *If the largest announced size is not in $(1 - \overline{c} - \delta, \overline{c} + \delta)$, then Algorithm 2 achieves the competitive ratio of at least $\overline{c}$.*

Algorithm 2 c-competitive algorithm for $0 < \delta < 0.5$.

```
1: if there is an item $x$ such that $x' \geq \bar{c} + \delta$ then
2:     Pack only $x$. END.
3: if all items have announced size $\leq 1 - \bar{c} - \delta$ then
4:     Pack greedily. END.
5: Let $x_l$ be the last item such that $x'_l \in (1 - \bar{c} - \delta, \bar{c} + \delta)$.
6: greedy := false.
7: while there is a next item $y$ in the queue do
8:     if $y = x_l$ then
9:         greedy := true
10:    Let $m$ be the size of already packed items.
11:    if greedy then
12:        Pack $y$ if it fits into the knapsack.
13:    else if $m \in [\bar{c} - (x'_l - \delta), 1 - (x'_l + \delta)]$ or $y + m \in (1 - (x'_l + \delta), \bar{c})$ then
14:        Skip $y$.
15:    else
16:        Pack $y$ if it fits into the knapsack.
```

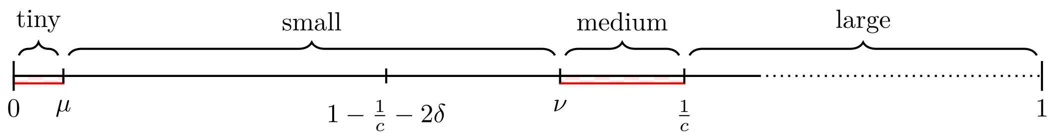

Figure 2 Range of item sizes. *Forbidden* ranges of item sizes to be packed are marked in red.

Proof. First, suppose there is an item x of announced size at least $\bar{c} + \delta$. In this case, x is the only item packed, see line 2. Since $x \geq \bar{c}$ is true, the competitive ratio is ensured.

Second, suppose that all items have announced size at most $1 - \bar{c} - \delta$. In this case, we pack greedily, see line 4. Let x be the first item that cannot be packed by the algorithm; if x does not exist, we have found the optimal solution. Otherwise, $x \leq 1 - \bar{c}$ holds, which means that we have packed more than $\bar{c}$ and the competitive ratio is also ensured. ◀

For the rest of the section, suppose that the largest announced item is in $(1 - \bar{c} - \delta, \bar{c} + \delta)$, and let x_l be the last announced item such that $x'_l \in (1 - \bar{c} - \delta, \bar{c} + \delta)$. In this case, the algorithm can guarantee a packing of at least $\bar{c}$ for most instances. To see this, we classify each item y as either *tiny, small, medium* or *large* depending on the actual size of y and the sum of the already packed items when y arrives. We use the substitutions $\mu := \bar{c} - (x'_l - \delta)$ and $\nu := 1 - (x'_l + \delta)$. To give an overview of the variables and relations we provide Figure 2.

▶ **Definition 10.** *Let y be an item and let m be the sum of the already packed items when y arrives.*

We say that an item y is tiny *if $y + m \in [0, \mu)$,* small *if $y + m \in [\mu, \nu]$,* medium *if $y + m \in (\nu, \bar{c})$, and* large *otherwise.*

Observe that before x_l arrives, medium items are skipped by the algorithm, see line 13. The case when a small or large item arrives is handled by the following lemma.

▶ **Lemma 11.** *If a small or large item y arrives before x_l, then Algorithm 2 reaches a packing of $\bar{c}$.*

Proof. Let y be the first small or large item that arrives and let m be the size of the packing when y arrives. Since medium items are skipped, only tiny items were packed before y, which means $m < \mu$.

First, suppose that y is small. In this case, y is packed and after that, the knapsacks packing is of size in $[\mu, \nu]$, which means that no item is packed until x_l arrives because the condition on line 13 is satisfied. Observe that when x_l arrives, the algorithm tries to pack it. Note that $m + y + x_l \leq m + (\nu - m) + (x'_l + \delta) = 1$, which means that x_l fits into the knapsack when it arrives. Now observe that after x_l is packed, the knapsack has size at least $m + y + x_l \geq m + (\mu - m) + (x'_l - \delta) = \bar{c}$, as required.

Second, suppose that y is large. Note that $y < \bar{c} + 2\delta$ since otherwise we would be in the case handled by Lemma 9.

Now observe that $m + y \leq \mu + (\bar{c} + 2\delta) = 2\bar{c} - x'_l + 3\delta \leq 2\bar{c} - (1 - \delta - \bar{c}) + 3\delta = 3\bar{c} + 4\delta - 1 \leq 3q + 4\delta - 1$, where the last inequality follows from $\bar{c} = \min(p, q)$. Now we expand the definition of q, see Equation (1):

$$3q + 4\delta - 1 = 4\delta - 1 + 3(1 - 2\delta - \frac{1}{\lceil \frac{2}{1-2\delta} \rceil}) = 2 - 2\delta - \frac{3}{\lceil \frac{2}{1-2\delta} \rceil} \leq 2 - 2\delta - (1 - 2\delta) = 1$$

Hence, y fits into the knapsack. By Definition 10, $y + m \geq \bar{c}$ is valid, as required.

In both cases, the algorithm packs y and can guarantee a packing of at least $\bar{c}$. ◀

Recall that the instance of OSKE we are solving is (P, δ). The following lemma says that we can assume that x_l is the last item.

▶ **Lemma 12.** *Let P' be the prefix of P ending with x_l and suppose that Algorithm 2 achieves a competitive ratio of at most c on (P', δ). Then, it also achieves a competitive ratio of at most c on (P, δ).*

Proof. If all items after x_l are packed, then the competitive ratio can only decrease. Suppose that an item x comes after x_l and is not packed. By choice of x_l, we know that $x' \leq 1 - \bar{c} - \delta$, which means that $x \leq 1 - \bar{c}$. Hence, the algorithm has packed more than $\bar{c}$, and the competitive ratio is at most c. ◀

By Lemma 11 and 12, we may assume that x_l is the last item and all other items are either tiny or medium. We split the analysis into two cases depending on whether at most $\lfloor \frac{2}{1-2\delta} \rfloor$ non-tiny items are packed in an optimal solution.

▶ **Lemma 13.** *Assume there are no small or large items, and x_l is the last item. If there is an optimal solution packing at most $k := \left\lfloor \frac{2}{1-2\delta} \right\rfloor$ non-tiny items, then Algorithm 2 achieves a competitive ratio of at most $\frac{1}{p}$.*

Proof. Recall that the algorithm packs exactly the tiny items and x_l, which means we may assume that there are no tiny items (their presence would decrease the competitive ratio). Since $x_l \geq 1 - \bar{c} - 2\delta$ holds due to the choice of x_l, we may assume $x_l = 1 - \bar{c} - 2\delta$ without decreasing the ratio. By Equation (3), $1 - \bar{c} - 2\delta \leq \bar{c}$ is valid, since $\bar{c} = \min(p, q)$ is defined. Hence, each item is of size at most $\bar{c}$ and the ratio is at most $\frac{k\bar{c}}{1-\bar{c}-2\delta}$. By Equation (2), we have

$$\frac{k\bar{c}}{1 - \bar{c} - 2\delta} \leq \frac{kp}{1 - p - 2\delta} = \frac{1}{p}.$$

Hence, the algorithm achieves a competitive ratio of $\frac{1}{p}$. This matches the lower bound given by Theorem 4 for the case $p \leq q$. ◀

For the other case, we can guarantee that the knapsack is filled with at least $\frac{1}{q}$.

▶ **Lemma 14.** *Assume there are no small or large items, and x_l is the last item. If there is an optimal solution packing at least $k := \left\lceil \frac{2}{1-2\delta} \right\rceil$ non-tiny items, then Algorithm 2 achieves a competitive ratio of at most $\frac{1}{q}$.*

Proof. Since at least k non-tiny items are packed by an optimal solution, there is a non-tiny item of size at most $1/k$. First, suppose that there is a medium item $y \neq x_l$ such that $y \leq 1/k$. Let m be the size of the packed items immediately before y arrives. Since y is medium, we have $1/k + m \geq y + m > \nu$. Hence, after x_l is packed, the algorithm has packed

$$m + x_l \geq (\nu - 1/k) + (x_l' - \delta) = 1 - (x_l' + \delta) - 1/k + x_l' - \delta = 1 - 2\delta - 1/k = q,$$

which implies a competitive ratio of at most $1/q$.

The remaining case is $y > 1/k$ for each medium item $y \neq x_l$ and $x_l \leq 1/k$. Observe that

$$x_l' \leq 1/k + \delta = 1 - (1 - 2\delta - 1/k) - \delta = 1 - q - \delta \leq 1 - \bar{c} - \delta,$$

which is a contradiction since $x_l' > 1 - \bar{c} - \delta$. Hence, this case cannot occur, and the algorithm achieves a competitive ratio of at most $\frac{1}{q}$. ◀

With the previous observations, the following theorem immediately follows.

▶ **Theorem 15.** *For every $\delta \in (0, \frac{1}{2})$, Algorithm 2 achieves a competitive ratio of $\frac{1}{\min(p,q)}$.*

3 Online Simple Knapsack with Removability

In this section, the algorithm is allowed to discard previously packed items. We consider not only additive estimate accuracy but will later also note that multiplicative accuracy behaves very similarly.

3.1 Lower Bounds

▶ **Theorem 16.** *For every $\delta \in (0, \frac{3}{4} - \frac{\sqrt{5}}{4}]$, there exists no algorithm solving the* OSKE *problem with removability and item size estimate with a competitive ratio better than $1/x$, where*

$$x = \frac{2 - 2\delta}{3 - 2\delta}$$

Proof. Let $\varepsilon > 0$ be small enough so that $x + 2\varepsilon \leq x + \delta$ and $1 - x - \varepsilon \geq 1 - x + \varepsilon - \delta$. An arbitrary algorithm is given the following prediction:

$$(1 - x, x + \varepsilon, x, 1 - x + \varepsilon - \delta)\,.$$

The first two items presented are $1 - x$ and $x + \varepsilon$. The algorithm can pack either of these items but not both.

Case 1: The algorithm packs $x + \varepsilon$. The third item is presented as x and the last one as $1 - x + \varepsilon$. Since $x > 0.5$, the algorithm can pack at most one item from the set $\{x, x + \varepsilon, 1 - x + \varepsilon\}$ and its best choice is keeping $x + \varepsilon$, whereas an optimal solution is to pack x and $1 - x$. Hence, the competitive ratio goes to $1/x$ as ε goes to 0.

Case 2: The algorithm packs $1 - x$. The next item is presented as $x + 2\varepsilon$. Now the algorithm either keeps $1 - x$ or discards it and packs $x + 2\varepsilon$.

Case 2.1: The algorithm packs $x + 2\varepsilon$. The final item is presented as $1 - x - \varepsilon$. Since $x + 2\varepsilon \geq 1 - x - \varepsilon$, the algorithm can pack at most $x + 2\varepsilon$, whereas the optimal solution packs the full knapsack by taking $x + \varepsilon$ and $1 - x - \varepsilon$. Hence, the competitive ratio goes to $1/x$ as ε goes to 0.

Case 2.2: The algorithm keeps $1 - x$. The final item is presented as small as possible and presented as $y := 1 - x - 2\delta + \varepsilon$. Now the algorithm can pack at most $1 - x + y$ whereas the optimal solution packs $x + 2\varepsilon + y$. Hence, the competitive ration goes to $(1 - x + y)/(x + y)$ as ε goes to 0. By definition of x, we see that also, in this case, the algorithm cannot achieve a better competitive ratio than $\frac{1}{x} = c$. ◀

Note that the construction of Theorem 16 will yield a bound of Φ for $\delta = \frac{3}{4} - \frac{\sqrt{5}}{4}$, which can also be achieved without estimates as in Iwama and Taketomi [24].

▶ **Theorem 17.** *For all $\delta > \frac{3}{4} - \frac{\sqrt{5}}{4}$, no algorithm for the online simple knapsack problem with removability and estimates can achieve a competitive ratio of less than Φ.*

3.2 Upper Bounds

In this section, we will present an algorithm that matches the lower bound of the previous part. Again, define $x = \frac{2-2\delta}{3-2\delta}$. We call items of size at most $1 - x$ *small* items, items of size strictly between $1 - x$ and x *medium* items, and items of size at least x *large* items.

Algorithm 3 for Online Knapsack with Removability and Estimates.

```
1: Let $x_l$ be the last item such that $x'_l > 1 - x - \delta$.
2: while there is an item $y$ in the queue do
3:     if at least $x$ has been packed then END.
4:     if $y$ is large then Remove everything. Pack $y$. END.
5:     if $y$ is small then Pack $y$ if it fits.
6:     if $y$ is medium then
7:         if no medium item is packed then
8:             Pack* $y$.
9:         else
10:            Let $z$ be the medium item packed.
11:            if $y + z \leq 1$ then Remove everything but $z$. Pack $y$. END.
12:            else if $y < z$ or $y = x_l > z$ then Remove $z$. Pack* $y$.
13:            else Ignore $y$.
```

* If it is not possible to pack y on lines 8 or 12, we keep removing small items until it becomes possible.

Note that once two medium items are packed, Algorithm 3 terminates, see line 11. Hence, line 10 is well defined as at most one medium item can be packed at that point.

Now we prove that Algorithm 3 matches the lower bound of Theorem 16.

▶ **Theorem 18.** *Algorithm 3 achieves a competitive ratio at least $1/x$ for $\delta \leq \frac{3}{4} - \frac{\sqrt{5}}{4}$.*

Proof. First, suppose that y is the first large item in the instance. Observe that the algorithm packs y on line 4 and terminates. Since $y \geq x$, a competitive ratio of at least $1/x$ is achieved. From now on, suppose that there are no large items.

Second, suppose that there are two medium items y_1 and y_2 such that $y_1 + y_2 \leq 1$. Observe that until x_l arrives, if there is only one medium item packed, then it is the smallest medium item that has arrived so far (see line 12). Hence, there is an iteration of the while loop in which the condition on line 11 is satisfied, i.e., two medium items are packed and the algorithm terminates. Since $2(1-x) > x$ holds for $\delta > 0$, the algorithm has again packed at least x as required. From now on, suppose that no two medium items fit together, no matter their position in the instance.

Third, suppose there is a small item a that is not part of the knapsack at the end, i.e., a was ignored on line 5 or removed on line 8 or 12 before packing y. Let m be the value in the knapsack at the end of the iteration in which a is discarded, and observe that $m + a > 1$ (otherwise a would be packed, resp. not removed). Since $a \leq 1 - x$, we have $m > 1 - a \geq 1 - (1 - x) = x$. Hence, the algorithm has packed at least x (and terminates in the next iteration, see line 3). From now on, we may assume that the algorithm packs all small items. In particular, if there is at most one medium item, the algorithm finds the optimal solution. Hence, assume that there are at least two medium items.

Observe that by definition of x_l, there are no medium items after x_l. Hence, when x_l arrives, there is a medium item y in the knapsack. Let s be the size of all small items except for x_l (note that x_l is small or medium). Assume x_l is small, i.e., the algorithm packs at least $x_l + (1 - x) + s$. Since no two medium items fit together, an optimal solution packs at most $x_l + x + s$. Observe that $x_l \geq 1 - x - \delta - \delta$, as it was announced with size at least $1 - x - \delta$. Hence the competitive ratio is at most:

$$\frac{x_l + x + s}{x_l + (1 - x) + s} \leq \frac{x + 1 - x - 2\delta}{1 - x + 1 - x - 2\delta} = \frac{1}{x} \text{ as desired.}$$

Finally, suppose that x_l is medium. Observe that the algorithm packs $\max(y, x_l) + s > 0.5 + s$, as y and x_l combined exceed 1. Since no two medium items fit together, an optimal solution packs at most $x + s$. Hence, the competitive ratio is at most $x/0.5 < 1/x$ as desired. ◀

Again, if $\delta \leq \frac{3}{4} - \frac{\sqrt{5}}{4}$, an optimal algorithm does not need to consider the estimates given.

▶ **Theorem 19.** *For all $\delta > 0$, there is an algorithm that achieves a competitive ratio of Φ for the online simple knapsack problem with removability and estimates [24].*

3.3 Multiplicative Estimate Accuracy

While the previous research was dedicated to additive estimate accuracy, it turned out that the results easily translate to the multiplicative model (as studied in [20]) with removability. This is not surprising, as the estimate accuracy is only needed for one item at the end in both the lower and the upper bounds, which has a medium size.

Therefore, it is possible to use an analogous lower bound construction and upper bound algorithm for the multiplicative case when the aimed competitive ratio is changed accordingly. In the multiplicative setting, x, which is defined as $\frac{2-2\delta}{3-2\delta}$ in the additive setting, can be redefined to $x = \frac{\sqrt{\delta^2+10\delta+9}+\delta-3}{4\delta}$, achiving a competitive ratio of $\frac{1}{x}$ then. As in the additive case, both bounds are tight until they reach Φ where the estimates become worthless due to the bound of Φ without estimates.

4 Final Remarks and Open Problems

In this article, we were able to provide tight bounds for all values of δ for both, the additive estimate accuracy for the online simple knapsack with irrevocable decisions, and for the variant with the option to remove items.

In comparison to the setting of the online simple knapsack with estimates and a multiplicative accuracy factor as already covered [20], it turned out that additive errors are relatively harder to deal with for an algorithm, in particular for the case without removability. This mainly follows from the fact that, in the additive setting, an adversary can announce items and decide later, when presenting them, if they should be presented as 0 or not. As already the pathological counterexample from the beginning has trouble dealing with such items, it is not surprising that our lower bound constructions consist of multiple tiny items that might turn out to be 0. In the multiplicative model however such constructions are not possible.

The additive and multiplicative model with removability are much more closely connected to each other, which aligns with our observation about the crucial role of the potentially 0-items: As in the removability setting all small items can be packed without concerns, those items turn out to support an online algorithm in its goal of achieving a low competitive ratio.

As the variant of online knapsack with estimates can be motivated easily by practical applications, other variants can be motivated by real-world scenarios as well. Therefore, given some practical problems, it would not be surprising if they could be modeled best with a combination of the estimate model with another knapsack variant.

For example, in the advice setting an algorithm is allowed to ask a limited amount of questions that will be answered truthfully. When you are given the list of estimates, you might again realize that there are just details missing which hinder you from significantly improving your solution quality. Therefore, you might decide to measure a few objects more precisely, as you realize that they can turn out to be crucial.

Connections are also conceivable when additionally allowing to delay some decisions for reservation costs, to exceed the capacity of the knapsack, or to use randomized algorithms. Furthermore, so far we assume that the estimates are correct, e.g., all items are actually close to their estimate. It would be interesting to analyze the behavior in the setting of online algorithms with predictions when the estimates can turn out to be wrong at least partly.

Another approach could be to make an own accuracy δ_i available for each item x_i, instead of just measuring the maximum distortion as we investigated so far.

Finally it would be interesting to extend the setting with estimates to the general knapsack problem. Here several variants are thinkable, for example, which part of the input is predicted, e.g., the size, the weight, or the density of the items.

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 Eliás, Adam Polak, and Bertrand Simon. Mixing predictions for online metric algorithms. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 969–983. PMLR, 2023. URL: https://proceedings.mlr.press/v202/antoniadis23b.html.

3 Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. *ACM Trans. Algorithms*, 19(2):19:1–19:34, 2023. doi:10.1145/3582689.

4 Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1070–1080, 2021. doi:10.1145/3406325.3451023.

5 Yossi Azar, Stefano Leonardi, and Noam Touitou. Distortion-oblivious algorithms for minimizing flow time. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 252–274. SIAM, 2022. doi:10.1137/1.9781611977073.13.

6 Yossi Azar, Eldad Peretz, and Noam Touitou. Distortion-oblivious algorithms for scheduling on multiple machines. In *33rd International Symposium on Algorithms and Computation (ISAAC 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

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 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.

9 Magnus Berg, Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Online minimum spanning trees with weight predictions. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 - August 2, 2023, Proceedings*, volume 14079 of *Lecture Notes in Computer Science*, pages 136–148. Springer, 2023. doi:10.1007/978-3-031-38906-1_10.

10 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.

11 Hans-Joachim Böckenhauer, Fabian Frei, and Peter Rossmanith. Removable Online Knapsack and Advice. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.STACS.2024.18.

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. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 - August 2, 2023, Proceedings*, volume 14079 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 2023. doi:10.1007/978-3-031-38906-1_14.

15 Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Online unit profit knapsack with predictions. *Algorithmica*, 86(9):2786–2821, 2024. doi:10.1007/S00453-024-01239-Y.

16 Elisabet Burjons, Matthias Gehnen, Henri Lotze, Daniel Mock, and Peter Rossmanith. The Online Simple Knapsack Problem with Reservation and Removability. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.MFCS.2023.29.

17 Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In Marc'Aurelio Ranzato, Alina Beygelzi-

mer, 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.

18 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.

19 Matthias Gehnen, Ralf Klasing, and Émile Naquin. Graph exploration with edge weight estimates. *arXiv preprint arXiv:2501.18496*, 2025. doi:10.48550/arXiv.2501.18496.

20 Matthias Gehnen, Henri Lotze, and Peter Rossmanith. Online Simple Knapsack with Bounded Predictions. In *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289, 2024. doi:10.4230/LIPICS.STACS.2024.37.

21 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.

22 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.

23 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.

24 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.

25 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.

26 Billy Jin and Will Ma. Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model. In *NeurIPS*, 2022.

27 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.

28 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.

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

30 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.

31 Ziv Scully, Isaac Grosof, and Michael Mitzenmacher. Uniform bounds for scheduling with job size estimates. *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, 2022.

32 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.

33 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.

34 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.

35 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.

36 Ali Zeynali, Bo Sun, Mohammad Hassan Hajiesmaili, and Adam Wierman. Data-driven competitive algorithms for online knapsack and set cover. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 10833–10841. AAAI Press, 2021. doi:10.1609/AAAI.V35I12.17294.

37 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.