Algorithmica (2024) 86:2786–2821
https://doi.org/10.1007/s00453-024-01239-y

Online Unit Profit Knapsack with Predictions

Joan Boyar[1] · Lene M. Favrholdt[1] · Kim S. Larsen[1]

Received: 11 July 2022 / Accepted: 14 May 2024 / Published online: 13 June 2024

Abstract

A variant of the online knapsack problem is considered in the setting of predictions. In Unit Profit Knapsack, the items have unit profit, i.e., the goal is to pack as many items as possible. For Online Unit Profit Knapsack, the competitive ratio is unbounded. In contrast, it is easy to find an optimal solution offline: Pack as many of the smallest items as possible into the knapsack. The prediction available to the online algorithm is the average size of those smallest items that fit in the knapsack. For the prediction error in this hard online problem, we use the ratio $r = \frac{a}{\hat{a}}$ where a is the actual value for this average size and $\hat{a}$ is the prediction. We give an algorithm which is $\frac{e-1}{e}$-competitive, if $r = 1$, and this is best possible among online algorithms knowing a and nothing else. More generally, the algorithm has a competitive ratio of $\frac{e-1}{e}r$, if $r \leq 1$, and $\frac{e-r}{e}r$, if $1 \leq r < e$. Any algorithm with a better competitive ratio for some $r < 1$ will have a worse competitive ratio for some $r > 1$. To obtain a positive competitive ratio for all r, we adjust the algorithm, resulting in a competitive ratio of $\frac{1}{2r}$ for $r \geq 1$ and $\frac{r}{2}$ for $r \leq 1$. We show that improving the result for any $r < 1$ leads to a worse result for some $r > 1$.

Keywords Online algorithms · Predictions · Knapsack problem · Competitive analysis

1 Introduction

In this paper, we consider the Online Unit Profit Knapsack Problem. The request sequence consists of n items with sizes in $(0, 1]$. An online algorithm receives them

Joan Boyar, Lene M. Favrholdt and Kim S. Larsen have contributed equally to this work.

✉ Kim S. Larsen
kslarsen@imada.sdu.dk

Joan Boyar
joan@imada.sdu.dk

Lene M. Favrholdt
lenem@imada.sdu.dk

[1] Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark

one at a time, with no knowledge of future items, and makes an irrevocable decision for each, either accepting or rejecting the item. It cannot accept an item if its size, plus the sum of the sizes of the already accepted items, is greater than 1. The goal is to accept as many items as possible. The obvious greedy algorithm solves the offline Unit Profit Knapsack Problem, since the set consisting of as many of the smallest items as fit in the knapsack is an optimal solution. Let OPT_s denote this optimal solution. Even for this special case of the Knapsack Problem, no competitive online algorithm can exist.

We study the Online Unit Profit Knapsack Problem under the assumption that (an approximation of) the average item size, a, in OPT_s is known to the algorithm. This particular information could likely be estimated from historical data. The measure we use for the prediction error is the ratio $r = \frac{a}{\hat{a}}$, where a is the actual value for this average size in OPT_s and $\hat{a}$ is the prediction. We consider the case, where the exact value of $\hat{a} = a$ is given to the algorithm by an oracle, calling this an *accurate prediction*, as well as the case where $\hat{a}$ could possibly be incorrect. Predictions could be inaccurate, for example, because the characteristics of the input may be different depending on the time of day the input is produced, which source produced the input, etc.

For an algorithm using predictions, the competitive ratio obtained with accurate predictions is called the *consistency* of the algorithm [1, 2]. Note that in our case, the prediction is accurate when $r = 1$. The *robustness* of the algorithm is its competitive ratio in the case where there is no guarantee on the quality of the prediction. We obtain algorithms that are at least $\frac{1}{2}$-consistent and have a smooth transition between consistency and robustness that depends on the value of the error measure, r. Since no online algorithm can be competitive without predictions, no online algorithm with predictions can be better than 0-robust.

1.1 Preliminaries

For any input sequence, σ, and algorithm, ALG, for the Online Unit Profit Knapsack Problem, $\text{ALG}(\sigma)$ denotes the number of items accepted by ALG when processing σ.

We use the asymptotic competitive ratio throughout this paper. Thus, an algorithm ALG, is c-competitive if there exists a constant b such that for all request sequences σ, $\text{ALG}(\sigma) \geq c\text{OPT}(\sigma) - b$. ALG's competitive ratio is then $\sup\{c \mid \text{ALG} \text{ is } c\text{-competitive}\}$. Note that this is a maximization problem and all competitive ratios are in the interval [0, 1].

We use the notation $\mathbb{N} = \{1, 2, 3, \ldots\}$. At any given time during the processing of the input sequence, the *level* of the knapsack denotes the sum of the sizes of the items already accepted.

1.2 Previous Work

The Knapsack Problem is well studied and comes in many variants; see Kellerer et al. [3]. Cygan et al. [4] refer to the online version we study, where all items give the same profit, as the *unit* case. They mention that it is well-known that no online algorithm for this version of the problem is competitive, i.e., has a finite competitive ratio, and they present a sequence showing this for the strict competitive ratio (where the additive constant b must be zero). The sequence starts with an item of size 1. The sequence either ends there or is followed by $1/\varepsilon$ items of size ε – if and only if the algorithm accepts the item of size 1. Thus, the ratio is either $0/1 = 0$ or $1/(1/\varepsilon) = \varepsilon$. Since the adversary can choose ε, the algorithm cannot have a strict competitive ratio better than zero.

In the *General* Knapsack Problem, each item comes not only with a size, but also with a profit, and the goal is to accept as much profit as possible given that the total size must be at most 1. The ratio of the profit to the size is the *importance* of an item. (This is sometimes called *value*, but we want to avoid confusion with other uses of that word.)

The *Online* General Knapsack Problem was first studied by Marchetti-Spaccamela and Vercellis [5]; they prove that the problem does not allow for competitive online algorithms, even for Relaxed Knapsack (fractions of items may be accepted), where all item sizes are 1. They concentrate on a stochastic version of the problem, where both the profit and size coefficients are random variables. They present both a fixed and an adaptive threshold algorithm.

The Online General Knapsack Problem with oracle-based *advice* was studied in [6]. In this model, the algorithm is given information, called advice, about the input, and the goal is to determine how many bits of advice are necessary and sufficient to achieve a given competitive ratio. The fundamental issues and many of the early results on oracle-based advice algorithms, primarily in the direction of advice complexity, can be found in [7, 8], though many newer results for specific problems have been published since. The advice used in [6] includes the maximum importance, v, for items to accept. Since there might be many items of that importance, but the optimal solution may contain very few of them, they also include further advice, the fraction of the knapsack filled by items of importance v. In addition, they limit the amount of advice by giving k-bit approximations to these two values, the maximum importance and the fraction of the knapsack needed for these items. They show that no online algorithm using fewer than $\log n$ bits of advice is competitive, but for $\varepsilon > 0$, there exists a $(1 + \varepsilon)$-competitive algorithm using $O(\log n)$ bits of advice.

The authors of [6] also study the Online *Unweighted* (or Simple) Knapsack Problem, where the profit of each item is equal to its size. This version of the Knapsack Problem is also called the proportional or uniform case. In [6], it is shown that 1 bit of advice is sufficient to be $\frac{1}{2}$-competitive, $\Omega(\log n)$ bits are necessary to be better than $\frac{1}{2}$-competitive, and $n - 1$ advice bits are necessary and sufficient to be optimal.

In [9], the Online General Knapsack Problem is considered with the assumption that all items are much smaller than the unit capacity of the knapsack. The results depend on the ratio, γ, between known upper and lower bounds on the importance of the items. For this setting, a threshold algorithm of optimal competitive ratio $\frac{1}{\ln(\gamma)+1}$ is known [10]. The threshold depends on γ and the current level of the knapsack and it lower bounds the importance of accepted items. This is in contrast to our threshold function which depends on the number of relatively large items accepted so far. In [9], a parameter, α, is added to the threshold function, resulting in a family of algorithms with a competitive ratio which is proven to be at most a certain factor, depending on α, worse than the optimal competitive ratio. For $\alpha = 1$, this factor, called the degradation factor, is 1. Within this family, the subset of algorithms with a degradation factor no larger than a given factor, ϕ, is called the policy class of ϕ-degraded algorithms. Within a given policy class, standard machine learning techniques are used on data from a real world application to choose an algorithm, balancing worst-case guarantees (competitive ratio) against the performance in typical cases.

The Online General Knapsack Problem is also considered in [11], again with upper and lower bounds on the possible importance of items. To obtain a finite number of importance values, each importance is rounded to the nearest power of $1+\varepsilon$, for some $\varepsilon > 0$.

For each rounded importance, v, upper and lower bounds on the total size of items with importance v are predicted. The authors present an algorithm which has some similarities to ours. In particular their budget function has a similar function to our threshold function; both specify the maximum number of the low importance, large items that need to be accepted to obtain the proven competitive ratios. Their algorithm obtains what they prove to be the best possible competitive ratio (for the given predictions without rounding), up to an additive factor that goes to zero as the size of the largest item goes to zero. The paper also considers two generalizations of the Knapsack Problem, called the Generalized One-Way Trading Problem and the Two-Stage Online Knapsack Problem.

The Bin Packing Problem is closely related to the Knapsack Problem. This is especially true for the dual variant where the number of bins is fixed and the objective is to pack as many items as possible [12]; the Unit Price Knapsack Problem is Dual Bin Packing with one bin. The standard Bin Packing Problem was considered with predictions in [13]. They give an algorithm with a parameter that can be adjusted to obtain consistency r and robustness $\max\{33 - 18r, 7/4\}$, for any $1.5 < r \leq 1.75$.

Bin Packing is also studied in [14] in the standard setting for online algorithms with predictions, giving a trade-off between consistency and robustness, with the performance degrading as a function of the prediction error. They assume discrete item sizes, and the prediction is the number of items of each possible size. They also have experimental results.

Much additional work has been done for other online problems, studying variants with predictions (machine-learned advice, for instance), inspired by the work of Lykouris and Vassilvitskii [1, 15] and Purohit et al. [2] in 2018. Some of this further work is in the directions of search-like problems [16–20], scheduling [21–32], rental problems [33–35], caching/paging [36–40], and other problems [14, 41–45], while some papers attack multiple problems [13, 46–49]. For a survey, see [50].

We note here that the error measure we use, $r = \frac{a}{\hat{a}}$, has similarities to the *distortion* used for a non-clairvoyant flow problem in [24]. The distortion is based on the maximum ratios of the estimated-to-real and real-to-estimated processing times. In [24], the algorithm presented does not have or use knowledge of the distortion, and the competitive ratio found is a function of the distortion. They call this type of algorithm *distortion-oblivious*. Similarly, the algorithms we present do not have or use knowledge of the error ratio, and the competitive ratios we find are functions of the error ratio, so these algorithms could be called *error ratio oblivious*. Note that the non-clairvoyant flow time problem considered in [24] is one of the few problems

Table 1 The competitive ratios obtained by the two algorithms are shown.

Ranges of r	CAT $(0 < r < e)$	RAT $(0 < r)$
$r = 1$	$\frac{e-1}{e} \approx 0.6321$	$\frac{1}{2}$
$r \leq 1$	$\frac{r(e-1)}{e}$	$\frac{r}{2}$
$r \geq 1$	$\frac{e-r}{e}$	$\frac{1}{2r}$
Tightness	Pareto-like optimality (optimal for $r = 1$)	Pareto-like optimality

The target ranges for the algorithms are shown in parentheses. If CAT is used outside the range, the bound does not give additional information and the competitive ratio is just zero

considered with predictions, where there is no algorithm without predictions with a constant competitive ratio, and [24] also uses ratios for the error measure/distortion.

1.3 Our Results

We consider both the case where the prediction $\hat{a}$ is known to be accurate, so $r = 1$, and the case where it might not be accurate. Different algorithms are presented, but they have a common form.

For our algorithm Consistent Adaptive Threshold (CAT) where the prediction is accurate and, thus, $\hat{a} = a$, the competitive ratio is $\frac{e-1}{e}$, and we prove a matching upper bound that applies to any deterministic algorithm knowing a. This upper bound limits how well any algorithm using accurate predictions can do; the competitive ratio cannot be better than $\frac{e-1}{e} \approx 0.6321$ for $r = 1$.

If CAT is used for possibly inaccurate predictions, it obtains a competitive ratio of $r\frac{e-1}{e}$ for $r \leq 1$, $\frac{e-r}{e}$ for $1 \leq r \leq e$, and 0 for $r \geq e$. No deterministic algorithm can be better than this for both $r < 1$ and $1 < r < e$.

For another algorithm, Robust Adaptive Threshold (RAT), we have results for two cases: for $r \leq 1$ the competitive ratio is $\frac{r}{2}$, and for $r \geq 1$ the competitive ratio is $\frac{1}{2r}$. Thus, for accurate predictions, the competitive ratio of RAT is $\frac{1}{2}$, slightly worse than for the other algorithm. We show a negative result implying that a deterministic online algorithm cannot both be $\frac{1}{2r}$-competitive for a range of large r-values and better than $\frac{1}{2}$-competitive for $r = 1$. We summarize these results in Table 1.

In Sect. 6, we consider two upper bounds on the advice complexity of the Online Unit Price Knapsack Problem. One is based on an algorithm in this paper, relating missing bits in advice to errors in predictions. The other is based on techniques from [6], with different predictions, the maximum size for items to accept, plus the fraction of the knapsack filled by items of that size. A lower bound for the amount of advice necessary for optimality is also presented.

The section on advice complexity is brief, since our main focus is on predictions of information that may be easily obtainable. It seems believable that the average size of items in OPT_s of requests could be estimated: Since OPT_s contains the smallest items in the request sequence, it is easy to calculate an average from previous request sequences. It is a single number to collect and store, as opposed to more detailed information about a distribution. Since so little storage is required, one could keep multiple copies if, for instance, the expected average changes during the day. Our results also hold if the prediction is $\frac{1}{\text{OPT}}$. Thus, it is sufficient to store either the average size of the items in OPT_s or the value OPT.

2 Online Unit Profit Knapsack Without Predictions

As mentioned earlier, it is well-known that no Online Unit Profit Knapsack algorithm can have a constant competitive ratio. For completeness, we give a proof for the asymptotic competitive ratio here, since we have not found it in the literature.

Theorem 1 *No algorithm for the Online Unit Profit Knapsack Problem can have a constant competitive ratio.*

Proof Assume for the sake of contradiction that there exists an algorithm, ALG, and constants, b and c, such that for any input sequence, σ, $\text{ALG}(\sigma) \geq c\text{OPT}(\sigma) - b$.

Consider a family of input sequences, σ_r, $r \in \mathbb{N}$, each consisting of r rounds, $i = 1, 2, \dots, r$. In Round i, $\left\lfloor \frac{i}{c} \right\rfloor$ items of size $\frac{c}{i}$ arrive.

OPT will accept all items of the last round, so after i rounds, ALG must have accepted at least $c \left\lfloor \frac{i}{c} \right\rfloor - b > i - 1 - b$ items. Since ALG can only accept an integer number of items, this means that ALG has accepted at least $i - b$ items. Thus, the ith largest item accepted by ALG must have size at least $\frac{c}{b+i}$. Therefore, after r rounds, the total size of items accepted by ALG must be at least

$$\sum_{k=b+1}^{r} \frac{c}{k} = c\left(\sum_{k=1}^{r} \frac{1}{k} - \sum_{k=1}^{b} \frac{1}{k}\right)$$
$$> c(\ln(r) - \ln(b) - 1), \text{ since } \ln(n) < \sum_{k=1}^{n} \frac{1}{k} < \ln(n) + 1, n \in \mathbb{N}.$$

However, since $\ln(b)$ is a constant and $\lim_{r\to\infty} \ln(r) = \infty$, there must exist an r such that this total size is larger than 1. □

3 Threshold Algorithms

The simplest way to use information about the average item size, a, in an optimal solution seems to be to choose some $q > 1$ and accept items of size at most qa, as long as they fit in the knapsack. The algorithm ONE- THRESHOLD does exactly this, with $q = 2$.

Theorem 2 *The competitive ratio of* ONE- THRESHOLD*, as defined in Algorithm 1, is* $\frac{1}{2}$.

Proof For the lower bound, consider any input sequence, σ. We first consider the case where ONE- THRESHOLD rejects an item of size less than $2a$. In this case, the knapsack

Algorithm 1 Algorithm ONE- THRESHOLD.

```
1: $\hat{a} \leftarrow$ predicted average size of items in OPT$_s$
2: level $\leftarrow 0$
3: for each input item $x$ do
4:     if size($x$) $\leq 2\hat{a}$ and level + size($x$) $\leq 1$ then
5:         Accept $x$
6:         level += size($x$)
7:     else
8:         Reject $x$
```

is filled to more than $1 - 2a$. Since all accepted items have size at most $2a$, this means that the algorithm has accepted more than $\frac{1}{2a} - 1$ items. Hence, ONE- THRESHOLD$(\sigma) > \frac{1}{2}$OPT$(\sigma) - 1$.

We now consider the case where ONE- THRESHOLD rejects no items of size at most $2a$. We let ℓ and s denote the number of items in OPT's knapsack that are larger than $2a$ and no larger than $2a$, respectively. Note that ONE- THRESHOLD$(\sigma) \geq s$, since the algorithm rejects no items of size at most $2a$. Since the average size of the items in OPT's knapsack is a, $\ell < s$, so OPT$(\sigma) = \ell + s < 2s \leq 2 \cdot$ ONE- THRESHOLD(σ).

For the upper bound, consider the sequence σ_1 consisting of $\left\lfloor \frac{1}{2a} \right\rfloor$ items of size $2a$ followed by $\left\lfloor \frac{1}{a} \right\rfloor$ items of size a. We note that ONE- THRESHOLD$(\sigma_1) < \frac{1}{2}$OPT$(\sigma_1) + 1$.

For an alternative proof of the upper bound, consider the sequence σ_2 consisting of ℓ items of size $2a + \varepsilon_\ell$ and s items of size ε_s, where $\ell = s - 1$, $s = \left\lfloor \frac{1}{2a} - \frac{1}{2} \right\rfloor$, $\varepsilon_\ell < a/\ell$, and $\varepsilon_s = (a - \ell\varepsilon_\ell)/s$. The number of items in σ_2 is $\ell + s = 2s - 1$ and their total size is $\ell(2a + \varepsilon_\ell) + s\varepsilon_s = \ell(2a + \varepsilon_\ell) + a - \ell\varepsilon_\ell = 2\ell a + a = (2s - 1)a$, so the average size is a and the total size is $(2\left\lfloor \frac{1}{2a} - \frac{1}{2} \right\rfloor - 1)a \leq 1$. Thus, OPT$(\sigma_2) = \ell + s = 2s - 1 = 2 \cdot$ ONE- THRESHOLD$(\sigma_2) - 1$. □

Considering σ_1 in the proof of Theorem 2, it is clear that to improve the competitive ratio, one would have to lower the threshold to a value smaller than $2a$. However, with a threshold smaller than $2a$, the sequence σ_2 of that same proof could be adapted to prove a competitive ratio lower than $\frac{1}{2}$. Thus, working with just one threshold is not enough to obtain a competitive ratio better than $\frac{1}{2}$. However, an algorithm starting with a threshold larger than $2a$ can obtain a competitive ratio better than $\frac{1}{2}$ by sometimes lowering the threshold to a value below $2a$ at some point during execution. The algorithm TWO- THRESHOLDS outlined in Algorithm 2 is such an algorithm. It guards against both sequences σ_1 and σ_2 by allowing some items larger than $2a$ to be packed in the knapsack, but carefully limiting the number of such items.

Theorem 3 *The competitive ratio of* TWO- THRESHOLDS*, as defined in Algorithm 2, is* $\frac{5}{9}$.

Proof For the lower bound, consider any input sequence, σ. We first consider the case where TWO- THRESHOLDS rejects an item, x, even though its size is below the current threshold. Since the highest threshold is $\frac{9a}{4}$, size$(x) \leq \frac{9a}{4}$ and the knapsack is filled to more than $1 - \frac{9a}{4}$. Since TWO- THRESHOLDS accepts at most $\left\lceil \frac{2}{9a} \right\rceil$ items larger than $\frac{3a}{2}$,

Algorithm 2 Algorithm Two- Thresholds.

```
1: $\hat{a} \leftarrow$ predicted average size of items in $\mathrm{OPT}_s$
2: level $\leftarrow 0$
3: $n_\ell \leftarrow 0$
4: for each input item $x$ do
5:     if size($x$) $\leq \frac{3\hat{a}}{2}$ and level + size($x$) $\leq 1$ then
6:         Accept $x$
7:         level += size($x$)
8:     else if size($x$) $\leq \frac{9\hat{a}}{4}$ and $n_\ell < \frac{2}{9\hat{a}}$ and level + size($x$) $\leq 1$ then
9:         Accept $x$
10:        level += size($x$)
11:        $n_\ell$+ = 1
12:    else
13:        Reject $x$
```

$$\text{Two- Thresholds}(\sigma) \geq \left\lceil \frac{2}{9a} \right\rceil + \frac{1 - \left\lceil \frac{2}{9a} \right\rceil \frac{9a}{4}}{\frac{3a}{2}} = \frac{2}{3a} - \frac{1}{2}\left\lceil \frac{2}{9a} \right\rceil > \frac{5}{9a} - 1$$
$$\geq \frac{5}{9}\mathrm{OPT}(\sigma) - 1\,.$$

We now consider the case, where Two- Thresholds never rejects an item of size no more than the current threshold. We let T be the final threshold of Two- Thresholds. Thus, $T = \frac{9a}{4}$, if the algorithm never lowers the threshold, and otherwise, $T = \frac{3a}{2}$. We let ℓ and s denote the number of items in Opt's knapsack of size more than T and at most T, respectively.

If $T = \frac{9a}{4}$, then $\ell < \frac{4s}{5}$, because the average size of the items in Opt's knapsack is a. Thus, $\mathrm{OPT}(\sigma) = \ell + s < \frac{9s}{5}$. Since Two- Thresholds never rejects an item no larger than $\frac{9a}{4}$, Two- Thresholds$(\sigma) \geq s \geq \frac{5}{9}\mathrm{OPT}(\sigma)$. Otherwise, $T = \frac{3a}{2}$, which means that $\ell < 2s$, and $\mathrm{OPT}(\sigma) = \ell + s < 3s$. Thus, we arrive at Two- Thresholds$(\sigma) \geq \frac{2}{9a} + s > \frac{2}{9}\mathrm{OPT}(\sigma) + \frac{1}{3}\mathrm{OPT}(\sigma) = \frac{5}{9}\mathrm{OPT}(\sigma)$.

For the matching upper bound, the adversary uses a sequence, σ, consisting of $\left\lfloor \frac{2}{9a} \right\rfloor$ items of size $\frac{9a}{4}$ followed by $\left\lfloor \frac{1}{3a} \right\rfloor$ items of size $\frac{3a}{2}$, ending with $\left\lfloor \frac{1}{a} \right\rfloor$ items of size a. The first $\left\lfloor \frac{2}{9a} \right\rfloor + \left\lfloor \frac{1}{3a} \right\rfloor$ items have a total size of more than $1 - \frac{15a}{4} > 1 - 4a$. Hence, Two- Thresholds$(\sigma) \leq \frac{5}{9a} + 3 = \frac{5}{9}\mathrm{OPT}(\sigma) + 3$. □

As it turns out in Sect. 3.1, working with an unbounded number of different threshold values, an algorithm of optimal competitive ratio $\frac{e-1}{e} \approx 0.63$ can be obtained.

3.1 The Adaptive Threshold Algorithm

In Algorithm 3, we introduce an algorithm template, which is used to establish algorithms with predictions in the two following sections. The template omits the definition of a threshold function, $T(i)$, since it is different for the two algorithms. $T(i)$ is a threshold function in the sense that all items larger than this threshold are rejected. It depends on the predictions given, though not in the same manner. The threshold decreases over time: In both algorithms, the threshold functions have the property that

$T(i) > T(i+1)$ for $i \geq 1$. The role of the threshold function is to ensure that, for any $i \geq 1$, items larger than $T(i)$ are accepted, only if fewer than $i-1$ items larger than $T(i)$ have already been accepted. Note that the threshold could be lowered step by step from $T(i)$ to $T(i+1)$, $i \geq 1$, by accepting an item of size between $T(1)$ and $T(2)$, then an item between $T(2)$ and $T(3)$, and so on. However, it could also, for instance, be lowered directly from $T(1)$ to $T(10)$. This could be because of having six items of sizes between $T(8)$ and $T(9)$ and three items between $T(9)$ and $T(10)$ in the knapsack. The intuition is that the more relatively large items we already have in the knapsack, the more selective we can be, since there is a limit to how many items above a certain size OPT can accept while getting an average size of a. An example showing how the algorithm works is presented below as Example 1.

We use the notation n_x to denote the number of accepted items strictly larger than x.

Algorithm 3 Algorithm ADAPTIVE THRESHOLD, AT.

```
1: $\hat{a} \leftarrow$ predicted average size of items in OPT$_s$
2: level $\leftarrow 0$
3: for each input item $x$ do
4:     $i = \max_{j \geq 0} \{n_{T(j+1)} = j\}$
5:     if size($x$) $\leq T(i+1)$ and level + size($x$) $\leq 1$ then
6:         Accept $x$
7:         level += size($x$)
8:     else
9:         Reject $x$
```

Note that using $\max_{j \geq 0} \{n_{T(j+1)} \geq j\}$ instead of $\max_{j \geq 0} \{n_{T(j+1)} = j\}$ in Line 4 would result in the same algorithm, since we stop accepting items above a given size when the quota set by $T(i)$ has been reached. Thus, i is nondecreasing through the processing of the input sequence, and the value of the threshold function, $T(i)$, is decreasing in i, so larger items cannot be accepted after i increases.

For the $\frac{e-1}{e}$-consistent algorithm, the threshold function is $T(i) = \hat{a}e/(\hat{a}e(i-1)+1)$ (see Algorithm 4), and for the $\frac{1}{2}$-consistent algorithm, the threshold function is $T(i) = \sqrt{\hat{a}/(2i)}$ (see Algorithm 7).

Example 1 Suppose a request sequence is

$$I = \left\langle \frac{3}{16}, \frac{1}{24}, \frac{3}{8}, \frac{1}{3}, \frac{1}{4}, \frac{1}{3}, \frac{1}{12}, \frac{1}{6}, \frac{1}{16}, \frac{1}{9}, \frac{2}{9} \right\rangle.$$

Note that an optimal solution contains the seven smallest items, filling the knapsack to level $\frac{3}{16} + \frac{1}{24} + \frac{1}{12} + \frac{1}{6} + \frac{1}{16} + \frac{1}{9} + \frac{2}{9} = \frac{7}{8}$. The average size in OPT$_s$ is $\frac{1}{8}$. Suppose the prediction is correct, so $\hat{a} = 0.125$. Using the thresholds from Algorithm 4, we get that

$$T(1) = \frac{e}{8} \approx 0.340,$$

$$T(2) = \frac{e}{e+8} \approx 0.254,$$
$$T(3) = \frac{e}{2e+8} \approx 0.202,$$
$$T(4) = \frac{e}{3e+8} \approx 0.168,$$
$$T(5) = \frac{e}{4e+8} \approx 0.144,$$

and the smaller threshold values are not relevant. Figure 1 shows the sizes of the items in I compared to the threshold values.

The algorithm processes I as follows:

- The first item is *accepted*, since $\frac{3}{16} \le T(1)$. Since $\frac{3}{16} \le T(2)$, the threshold remains $T(1)$.
- The second item is also *accepted*, since $\frac{1}{24} \le T(1)$ and $\frac{3}{16} + \frac{1}{24} \le 1$.
- The third item is *rejected*, since $\frac{3}{8} > T(1)$.
- The fourth item is *accepted*, since $\frac{1}{3} \le T(1)$ and $\frac{3}{16}+\frac{1}{24}+\frac{1}{3} \le 1$. Since $\frac{1}{3} > T(2)$, the threshold is lowered to $T(2)$.
- The fifth item is *accepted*, since $\frac{1}{4} \le T(2)$ and $\frac{3}{16}+\frac{1}{24}+\frac{1}{3}+\frac{1}{4} \le 1$. The threshold is lowered to $T(4)$, since each of the three items of sizes $\frac{3}{16}$, $\frac{1}{3}$, and $\frac{1}{4}$ is larger than $T(4)$.
- The sixth item is *rejected*, since $\frac{1}{3} > T(4)$.
- The seventh item is *accepted*, since $\frac{1}{12} < T(4)$ and $\frac{3}{16}+\frac{1}{24}+\frac{1}{3}+\frac{1}{4}+\frac{1}{12} = \frac{43}{48} \le 1$.
- The eighth item is *rejected*, since the remaining empty space in the knapsack is only $\frac{5}{48} < \frac{1}{6}$.
- The ninth item is *accepted*, since $\frac{1}{16} < T(4)$ and $\frac{3}{16}+\frac{1}{24}+\frac{1}{3}+\frac{1}{4}+\frac{1}{12}+\frac{1}{16} = \frac{23}{24} \le 1$.
- The two remaining items are *rejected*, since they do not fit in the knapsack.

Algorithm 4 accepts six items.

4 Accurate Predictions

In this section, we give an $\frac{e-1}{e}$-competitive algorithm which receives a, the exact average size of the items in OPT_s, as a prediction and prove that it is best possible among algorithms that gets the exact value of a and no other information.

4.1 Positive Result–Lower Bound

To define an algorithm for accurate predictions, we define a threshold function; see Algorithm 4. Throughout this section, we assume that $\hat{a} = a$, but the algorithm is also used for possibly inaccurate predictions in Sect. 5.1.

First, we prove that CAT with $\hat{a} = a$ has competitive ratio at least $\frac{e-1}{e} \approx 0.6321$. For that, we need two simple lemmas, the first of which involves an obvious generalization of Harmonic numbers to non-integers:

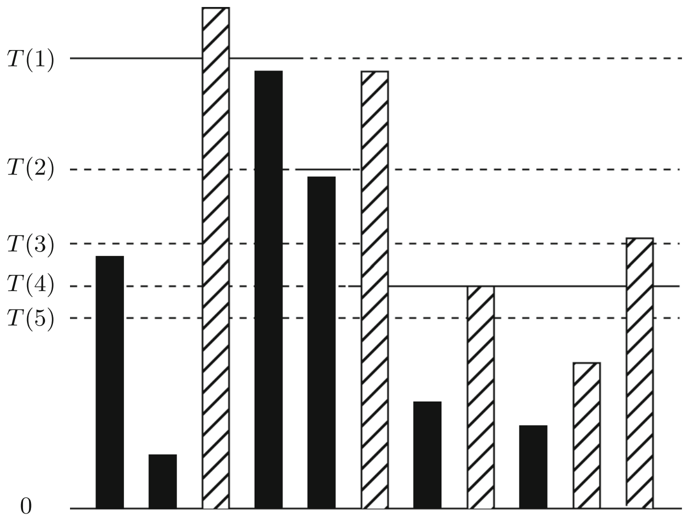

Fig. 1 Illustrating Example 1. Items are shown in the order they appear in I. Black items are accepted by Algorithm 4 and shaded items are rejected

```
Algorithm 4 CONSISTENT ADAPTIVE THRESHOLD, CAT.
1: Define $T(i) = \frac{\hat{a}e}{\hat{a}e(i-1)+1}$ for $i \geq 1$
2: Run ADAPTIVE THRESHOLD, Algorithm 3
```

Definition 1 *For a function f and real-valued x and y such that $y - x \in \mathbb{N} \cup \{0\}$, define $\sum_{i=x}^{y} f(i) = f(x) + f(x+1) + \cdots + f(y)$.*

For any real-valued $k \geq 1$, define the generalized Harmonic number, H_k, by $H_k = \sum_{i=1+k-\lfloor k \rfloor}^{k} \frac{1}{i}$.

The proofs of the following two lemmas consist of simple calculations and can be found in Appendix A.

Lemma 1 *If $k \geq p \geq 1$ and $k - p \in \mathbb{N} \cup \{0\}$, then*

$$\ln k - \ln(p+1) \leq H_k - H_p \leq \ln k - \ln p.$$

Lemma 2 *For any $a > 0$, $e^{1-ae} \geq e - e^2 a$.*

With those two lemmas, we can now prove the following theorem.

Theorem 4 *For $\hat{a} = a$,* CAT*, as defined in Algorithm 4, is $\frac{e-1}{e}$-competitive.*

Proof Consider a request sequence σ. Considering the conditional statement in the algorithm, if CAT rejects an item, x, then either $\text{size}(x) > T(i+1)$ or $\text{level}+\text{size}(x) > 1$.

Case 1: This is the case where, at some point, CAT rejects an item, x, with a size no larger than the current threshold, because $\text{level} + \text{size}(x) > 1$. The value of $T(k)$

from Algorithm 4 is an upper bound on the size of the kth largest item accepted by the algorithm. Thus, the kth largest accepted item has size at most

$$T(k) = \frac{ae}{ae(k-1)+1} = \frac{1}{k-1+\frac{1}{ae}}.$$

Using the generalization of the Harmonic function over non-integer values given in Definition 1, we obtain an upper bound on the total size of items accepted by CAT of

$$\text{level} \le \sum_{k=1}^{\text{CAT}(\sigma)} \frac{1}{k-1+\frac{1}{ae}} = \sum_{k=\frac{1}{ae}}^{\text{CAT}(\sigma)+\frac{1}{ae}-1} \frac{1}{k} = H_{\text{CAT}(\sigma)+\frac{1}{ae}-1} - H_{\frac{1}{ae}-1}.$$

By Lemma 1,

$$\begin{aligned} H_{\text{CAT}(\sigma)+\frac{1}{ae}-1} - H_{\frac{1}{ae}-1} &< \ln\left(\text{CAT}(\sigma) + \tfrac{1}{ae} - 1\right) - \ln\left(\tfrac{1}{ae} - 1\right) \\ &= \ln\left(\frac{\text{CAT}(\sigma)+\frac{1}{ae}-1}{\frac{1}{ae}-1}\right). \end{aligned}$$

By assumption, $\text{level} + \text{size}(x) > 1$, and since $\text{level} \le \ln\left(\frac{\text{CAT}(\sigma)+\frac{1}{ae}-1}{\frac{1}{ae}-1}\right)$, we have

$$\begin{aligned} &\ln\left(\frac{\text{CAT}(\sigma)+\frac{1}{ae}-1}{\frac{1}{ae}-1}\right) > 1 - \text{size}(x) \\ \Leftrightarrow\ &\frac{\text{CAT}(\sigma)+\frac{1}{ae}-1}{\frac{1}{ae}-1} > e^{1-\text{size}(x)} \\ \Leftrightarrow\ &\text{CAT}(\sigma) > \left(\frac{1}{ae}-1\right)e^{1-\text{size}(x)} - \frac{1}{ae} + 1 \\ \Leftrightarrow\ &\text{CAT}(\sigma) > \frac{e^{1-\text{size}(x)}-1}{ae} + 1 - e^{1-\text{size}(x)}. \end{aligned}$$

In the algorithm, i is at least zero, so it does not consider accepting items larger than $T(1) = ae$. Thus, $\text{size}(x) \le ae$.

$$\begin{aligned}
\text{CAT}(\sigma) &> \frac{e^{1-\text{size}(x)}-1}{ae}+1-e, && \text{since } -e^{1-\text{size}(x)} > -e \\
&> \frac{e^{1-ae}-1}{ae}+1-e, && \text{by the observation above} \\
&\geq \frac{e-e^2a-1}{ae}+1-e, && \text{by Lemma 2} \\
&= \frac{e-1}{ae}-2e+1 \\
&\geq \frac{e-1}{e}\text{OPT}(\sigma)-2e+1, && \text{since } \text{OPT}(\sigma) \leq \frac{1}{a}.
\end{aligned}$$

So, $\lim_{\text{OPT}\to\infty} \frac{\text{CAT}(\sigma)}{\text{OPT}(\sigma)} \geq \frac{e-1}{e}$.

Case 2: This is the case where CAT never rejects any item, x, when $\text{size}(x) \leq T(i+1)$. Let i_t denote the final value of i as the algorithm terminates. Suppose OPT accepts ℓ items larger than $T(i_t+1)$ and s items of size at most $T(i_t+1)$. Since OPT accepts ℓ items larger than $T(i_t+1)$ and $\ell+s$ items in total, we have $a > \ell \cdot T(i_t+1)/(\ell+s)$, which is equivalent to

$$s > \left(\frac{T(i_t+1)}{a}-1\right)\ell\,. \tag{1}$$

By the definition of T, we have that $T(i_t+1) = \frac{ae}{aei_t+1}$. Solving for the i_t on the right-hand side, we get

$$i_t = \frac{1}{T(i_t+1)} - \frac{1}{ae}. \tag{2}$$

Thus, CAT has accepted at least $i_t = \frac{1}{T(i_t+1)} - \frac{1}{ae}$ items of size greater than $T(i_t+1)$. Further, due to the assumption in this second case, CAT has accepted all of the s items no larger than $T(i_t+1)$. To see this, note that the values of the variable i of the algorithm can only increase, so at no point has there been a size demand more restrictive than $T(i_t+1)$.

We split in two subcases, depending on how $T(i_t+1)$ relates to the average size, a, in OPT_s.

Subcase 2a: $T(i_t+1) > a$. In this subcase, the lower bound on s of Ineq. (1) is positive.

$$\begin{aligned}
\frac{\text{CAT}(\sigma)}{\text{OPT}(\sigma)} &\geq \frac{\left(\frac{1}{T(i_t+1)}-\frac{1}{ae}\right)+s}{\ell+s}, && \text{by Eq. (2)} \\
&> \frac{\left(\frac{1}{T(i_t+1)}-\frac{1}{ae}\right)+\left(\frac{T(i_t+1)}{a}-1\right)\ell}{\ell+\left(\frac{T(i_t+1)}{a}-1\right)\ell}, && \text{by Ineq. (1)}
\end{aligned}$$

$$= \frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{ae}\right) + \left(\frac{T(i_t+1)}{a} - 1\right)\ell}{\frac{T(i_t+1)}{a}\ell}\,.$$

The second inequality follows since the ratio is smaller than one and s is replaced by a smaller, positive term in the numerator as well as the denominator.

We prove that this is bounded from below by $\frac{e-1}{e}$:

$$\begin{aligned}
&\frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{ae}\right) + \left(\frac{T(i_t+1)}{a} - 1\right)\ell}{\frac{T(i_t+1)}{a}\ell} \geq \frac{e-1}{e}\\
\Leftrightarrow\ & \frac{e}{T(i_t+1)} - \frac{1}{a} + e\left(\frac{T(i_t+1)}{a} - 1\right)\ell \geq e\frac{T(i_t+1)}{a}\ell - \frac{T(i_t+1)}{a}\ell\\
\Leftrightarrow\ & \frac{e}{T(i_t+1)} - \frac{1}{a} \geq \left(e - \frac{T(i_t+1)}{a}\right)\ell\\
\Leftrightarrow\ & \frac{ea - T(i_t+1)}{aT(i_t+1)} \geq \frac{ea - T(i_t+1)}{a}\ell\\
\Leftrightarrow\ & \frac{1}{T(i_t+1)} \geq \ell\,.
\end{aligned}$$

For the last biimplication, we must argue that $ea - T(i_t+1) \geq 0$, but this holds since $T(1) = ea$ and T is decreasing. Finally, the last statement, $\frac{1}{T(i_t+1)} \geq \ell$ holds regardless of the relationship between $T(i_t+1)$ and a, since the knapsack obviously cannot hold more than $\frac{1}{T(i_t+1)}$ items of size greater than $T(i_t+1)$.

Subcase 2b: $T(i_t+1) \leq a$.

$$\begin{aligned}
\frac{\mathrm{CAT}(\sigma)}{\mathrm{OPT}(\sigma)} &\geq \frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{ae}\right) + s}{\ell + s}\,, && \text{by Eq. (2)}\\
&\geq \frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{ae}\right)}{\ell}\,, && \text{since } s \geq 0 \text{ and } \frac{\mathrm{CAT}(\sigma)}{\mathrm{OPT}(\sigma)} \leq 1\\
&> \frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{ae}\right)}{\frac{1}{T(i_t+1)}}\,, && \text{since, as above, } \ell \leq \frac{1}{T(i_t+1)}\\
&= 1 - \frac{T(i_t+1)}{ae}\\
&\geq 1 - \frac{a}{ae}\,, && \text{by the subcase we are in}\\
&= \frac{e-1}{e}\,.
\end{aligned}$$

This concludes the second case, and, thus, the proof. □

4.2 General Negative Result–Upper Bound

Now, we show that CAT is best possible among online algorithms knowing a and nothing else, using Algorithm 5 for the adversary.

Algorithm 5 Adversarial sequence establishing the competitive ratio of a best possible online algorithm knowing a.

```
    ▷ Assume $a < \frac{1}{2e}$ and $\frac{1}{a} \in \mathbb{N}$
1:  $\varepsilon \leftarrow \frac{a^2}{10}$
2:  $k \leftarrow \left\lfloor \frac{1}{ae} \right\rfloor$
3:  while ALG's level $\leq 1 - \frac{1}{k} - k\varepsilon$ do
4:      for $k$ times do
5:          Give an item of size $\frac{1}{k} - \varepsilon$
6:          if ALG accepts then
7:              $k$++
8:              continue (* the while-loop *)
        ▷ ALG did not accept any of the $k$ items of this round.
9:      Give $\frac{1}{a} - k$ items of size $\frac{ka\varepsilon}{1-ka}$
10:     terminate                                              ▷ Case 1
11: Give $\frac{1}{a}$ items of size $a$                                ▷ Case 2
```

Before continuing with the analysis, consider the following example execution of Algorithm 5. The algorithm is used in the proof of Theorem 5.

Example 2 If $a = \frac{1}{6}$, then $\varepsilon = \frac{1}{12960}$ and $k = 2$ initially. Suppose that while $k = 2$, ALG accepts only the second item. Then, the adversarial sequence starts with $I_2 = \{\frac{1}{2} - \varepsilon, \frac{1}{2} - \varepsilon\}$. We consider two of the possible cases after this:

1. If ALG accepts none of the three items when $k = 3$, then the sequence ends with $I_3 = \{\frac{1}{3} - \varepsilon, \frac{1}{3} - \varepsilon, \frac{1}{3} - \varepsilon, \varepsilon, \varepsilon, \varepsilon\}$. OPT accepts all six items in I_3 and none in I_2, while ALG accepts one item from I_2 and at most the last three from I_3.
2. If ALG accepts the first item when $k = 3$, then the sequence ends with $I_3 = \{\frac{1}{3} - \varepsilon, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\}$. OPT accepts the last six items, while ALG accepts one item from I_2, plus the item of size $\frac{1}{3} - \varepsilon$ and at most one item of size $\frac{1}{6}$ from I_3.

Theorem 5 *Any deterministic algorithm getting only the correct value of a as a prediction has a competitive ratio of at most* $\frac{e-1}{e}$.

Proof Let ALG denote the online algorithm knowing a, and let σ be the adversarial sequence defined by Algorithm 5, which explains how the adversary defines its sequence based on ALG's actions.

The adversary strategy is to make ALG accept one item of each size $\frac{1}{k(i)}$, $i = 0, 1, \ldots$, where $k(i) = \left\lfloor \frac{1}{ae} \right\rfloor + i$, until an item of the next size, size $\frac{1}{k(i+1)}$, would not fit in ALG's knapsack (Case 2), or $k(i)$ items of size $\frac{1}{k(i)}$ have been given without ALG accepting any of them (Case 1). In Case 1, the sequence finishes with $\frac{1}{a} - k(i)$ very small items, giving an average size of a when combined with the $k(i)$ items of size $\frac{1}{k(i)}$

rejected by ALG. These items are the items accepted by OPT. In Case 2, the sequence ends with $\frac{1}{a}$ items of size a accepted by OPT. Since ALG's knapsack is already almost full, ALG can fit at most two of these items in its knapsack.

Let k_t be the value of k at the beginning of the last iteration of the while-loop. We perform a case analysis based on how the generation of the adversarial sequence terminates.

Case 1:
OPT accepts the k_t items of size $\frac{1}{k_t} - \varepsilon$ in the last iteration of the while-loop and the $\frac{1}{a} - k_t$ items of size $\frac{k_t a\varepsilon}{1-k_t a}$ for a total of $\frac{1}{a}$ items of total size

$$k_t\left(\frac{1}{k_t} - \varepsilon\right) + \left(\frac{1}{a} - k_t\right)\frac{k_t a\varepsilon}{1-k_t a} = 1 - k_t\varepsilon + (1-k_t a)\frac{k_t\varepsilon}{1-k_t a} = 1.$$

Note that the average size of the items accepted by OPT is a, consistent with the prediction.

ALG accepts one item in each iteration of the while-loop, except the last iteration, and at most $\frac{1}{a} - k_t$ items after that, so no more than

$$k_t - \left\lfloor\frac{1}{ae}\right\rfloor + \frac{1}{a} - k_t < \frac{1}{a} - \frac{1}{ae} + 1 = \frac{e-1}{e}\cdot\frac{1}{a} + 1.$$

Thus,

$$\text{ALG}(\sigma) \le \frac{e-1}{e}\cdot\frac{1}{a} + 1 = \frac{e-1}{e}\text{OPT}(\sigma) + 1.$$

Case 2:
OPT accepts the $\frac{1}{a}$ items of size a. For the analysis of ALG, we start by establishing an upper bound on k_t. The following inequality holds since ALG accepts one item per round, and ALG's level just before the last round is at most $1 - \frac{1}{k_t} - k_t\varepsilon$ before the last item of size $\frac{1}{k_t} - \varepsilon$ is accepted.

$$\begin{aligned}\sum_{k=\left\lfloor\frac{1}{ae}\right\rfloor}^{k_t}\left(\frac{1}{k} - \varepsilon\right) \le 1 - (k_t+1)\varepsilon \Rightarrow H_{k_t} - H_{\left\lfloor\frac{1}{ae}\right\rfloor - 1} - k_t\varepsilon < 1 - k_t\varepsilon \\ \Rightarrow \ln(k_t) - \ln\left(\left\lfloor\frac{1}{ae}\right\rfloor\right) < 1, \text{ by Lemma 1} \\ \Leftrightarrow k_t < e\left\lfloor\frac{1}{ae}\right\rfloor. \qquad (3)\end{aligned}$$

In the case we are treating, ALG leaves the while-loop because its level is more than $1 - \frac{1}{k_t+1} - (k_t+1)\varepsilon$. Now, we give a bound on the amount of space available at that point. For the first inequality, note that by the initialization of k in the algorithm, $k_t \ge \left\lfloor\frac{1}{ae}\right\rfloor$.

$$\frac{1}{k_t+1}+(k_t+1)\varepsilon < \frac{1}{\left\lfloor \frac{1}{ae}\right\rfloor+1}+\left(e\left\lfloor \frac{1}{ae}\right\rfloor+1\right)\varepsilon < ae+\left(\frac{1}{a}+1\right)\frac{a^2}{10}$$
$$< \left(e+\left(\frac{1+a}{10}\right)\right)a < 3a\,.$$

Thus, after the while-loop, Alg can accept at most two of the items of size a. Clearly, the number of rounds in the while-loop is $k_t - \left\lfloor \frac{1}{ae}\right\rfloor + 1$. Using $k_t < e\left\lfloor \frac{1}{ae}\right\rfloor$ from Eq. (3), we can now bound Alg's profit:

$$\text{Alg}(\sigma) \le k_t - \left\lfloor \frac{1}{ae}\right\rfloor + 1 + 2 < (e-1)\left\lfloor \frac{1}{ae}\right\rfloor + 3$$
$$\le \frac{e-1}{e}\frac{1}{a} + 3 = \frac{e-1}{e}\text{Opt}(\sigma) + 3.$$

This establishes the bound on the competitive ratio of $\frac{e-1}{e}$.

Finally, to ensure that our proof is valid, we must argue that the number of rounds we count in the algorithm and the sizes of items we give are non-negative. For the remainder of this proof, we go through the terms in the algorithm, thereby establishing this.

The largest value of k in the algorithm is k_t, and by Eq. (3) $k_t < e\left\lfloor \frac{1}{ae}\right\rfloor < \frac{1}{a}$. Additionally, from the start value of k, we know that $\left\lfloor \frac{1}{ae}\right\rfloor \le k$. Using these facts, together with the assumption from the algorithm that $a < \frac{1}{2e}$, we get the following bounds on the various terms.

$$1-\frac{1}{k}-k\varepsilon > 1-\frac{1}{\left\lfloor \frac{1}{ae}\right\rfloor}-\frac{1}{a}\frac{a^2}{10} > 1-\frac{1}{\left\lfloor \frac{1}{1/2}\right\rfloor}-\frac{1}{20e} > 0.$$

Further, $\frac{1}{k}-\varepsilon \ge \frac{1}{k_t}-\varepsilon > \frac{1}{\frac{1}{a}}-\frac{a^2}{10} > 0$ and $\frac{1}{a}-k \ge \frac{1}{a}-k_t > \frac{1}{a}-\frac{1}{a}=0$.

For the last relevant value, $1-ka \ge 1-k_ta > 1-\frac{1}{a}a = 0$ and from Case 1, we know that the $\frac{1}{a}-k_t$ items given in Line 9 of the algorithm sum up to at most one. □

5 Predictions

For the case where the predictions may be inaccurate, the algorithm CAT can be used with $\hat{a}$ possibly not being a, obtaining a positive competitive ratio as long as $r < e$, see Subsect. 5.1. In Subsect. 5.2, we give an adaptive threshold algorithm, RAT, that has a positive competitive ratio for all positive r.

For $r < \frac{1}{2}(e+\sqrt{e^2-2e}) \approx 2.06$, CAT has a better competitive ratio than RAT. Thus, if an upper bound on r of approximately 2 (or lower) is known, CAT may be preferred, and if a guarantee for any r is needed, RAT should be used.

5.1 Predictions with $r < e$

In this section, we consider the algorithm CAT with a prediction, $\hat{a}$, guaranteed to be less than e times as large as the correct value, a.

5.1.1 Positive Result–Lower Bound

In this section, we consider the algorithm CAT with $\hat{a}$ instead of a. Note that the lower bound of the theorem below is positive only when $r < e$. For $r \geq e$, the algorithm may not accept any items, and, hence, its competitive ratio is 0.

Theorem 6 CAT *has a competitive ratio of at least*

$$c_{\text{CAT}}(r) \geq \begin{cases} \frac{e-1}{e} \cdot r, & \text{if } r \leq 1 \\ \frac{e-r}{e}, & \text{if } r \geq 1\,. \end{cases}$$

Proof The proof is analogous to the proof of Theorem 4.

In Case 1, replacing a by $\hat{a}$, since the algorithm bases its actions on $\hat{a}$ instead of a, and setting $\text{OPT}(\sigma) = \frac{1}{r\hat{a}}$, results in a ratio of

$$\frac{\text{CAT}(\sigma)}{\text{OPT}(\sigma)} \geq \frac{e-1}{e} \cdot r$$

instead of $\frac{e-1}{e}$.

In Case 2, the lower bound on s given in Ineq. (1) depends on the actual average size, a, whereas the value of i_t given in Eq. (2) depends on $\hat{a}$, since the algorithm uses $\hat{a}$. The subcase distinction is still based on a.

In Subcase 2a, we obtain

$$\frac{\text{CAT}(\sigma)}{\text{OPT}(\sigma)} \geq \frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{\hat{a}e}\right) + s}{\ell + s} \geq \frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{\hat{a}e}\right) + \left(\frac{T(i_t+1)}{a} - 1\right)\ell}{\frac{T(i_t+1)}{a}\ell}\,.$$

Going through the same calculations as in the proof of Theorem 4, we get that

$$\begin{aligned} &\frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{\hat{a}e}\right) + \left(\frac{T(i_t+1)}{a} - 1\right)\ell}{\frac{T(i_t+1)}{a}\ell} \geq \frac{e-r}{e} \\ \Leftrightarrow\ &\frac{e}{T(i_t+1)} - \frac{1}{\hat{a}} \geq \left(e - r \cdot \frac{T(i_t+1)}{r\hat{a}}\right)\ell \\ \Leftrightarrow\ &\frac{1}{T(i_t+1)} \geq \ell\,. \end{aligned}$$

In subcase 2b, we obtain

$$\frac{\text{CAT}(\sigma)}{\text{OPT}(\sigma)} \geq \frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{\hat{a}e}\right) + s}{\ell + s} > 1 - \frac{T(i_t+1)}{\hat{a}e} \geq 1 - \frac{a}{\hat{a}e} = 1 - \frac{r\hat{a}}{\hat{a}e} = \frac{e-r}{e}.$$

Thus, we obtain a lower bound of $\frac{e-1}{e} \cdot r$ in Case 1 and a lower bound of $\frac{e-r}{e}$ in Case 2. For $r \leq 1$, $\frac{e-1}{e} \cdot r \leq \frac{e-r}{e}$, and for $r \geq 1$, $\frac{e-1}{e} \cdot r \geq \frac{e-r}{e}$. □

5.1.2 General Negative Result–Upper Bound

The following result shows that, for $r < e$, no algorithm can be better than CAT for both $r < 1$ and $r > 1$.

Theorem 7 *If a deterministic algorithm is $\frac{e-r}{e}$-competitive for all $1 \leq r < e$, it cannot be better than $r \cdot \frac{e-1}{e}$-competitive for any $r \leq 1$.*

Proof Consider an algorithm, ALG, and assume that ALG is $\frac{e-r}{e}$-competitive for all $1 \leq r < e$. Then there exists a constant, b, such that $\text{ALG}(\sigma) \geq \frac{e-r}{e}\text{OPT}(\sigma) - b$, for any sequence σ and any $1 \leq r < e$. This constant b is given as a parameter to Algorithm 6, constructing an adversarial sequence, σ.

Algorithm 6 Adversarial sequence for $r < e$; the adversarial algorithm takes two parameters, $r_2 < 1$ and $b \geq 0$.

```
    ▷ Assume $\hat{a} < \frac{1}{2e+b}$ and $\frac{1}{r_2\hat{a}} \in \mathbb{N}$
1:  $k \leftarrow \left\lfloor \frac{1}{\hat{a}e} \right\rfloor - 1$
2:  while ALG's level $\leq 1 - \frac{1}{k+1} - (b+1)\hat{a}e$ and $\frac{1}{k+1} \geq \hat{a}$ do
3:      $k$++
4:      for $k$ times do
5:          Give an item of size $\frac{1}{k}$
6:          if ALG accepts then
7:              continue (* the while-loop *)
8:      if ALG has accepted fewer than $k - \left\lfloor \frac{1}{\hat{a}e} \right\rfloor - b$ items then
9:          terminate
10: Give $\frac{1}{r_2\hat{a}}$ items of size $r_2\hat{a}$
```

Let k_t be the value of k at the end of the last iteration of the while-loop. If the adversarial algorithm terminates in Line 9, then ALG has accepted at most $k_t - \left\lfloor \frac{1}{\hat{a}e} \right\rfloor - b - 1$ items. For termination in Line 9, ALG has not accepted any of the k_t items in the for-loop immediately preceding this, so k_t items of size $\frac{1}{k_t}$ were given. In this case, OPT accepts exactly these k_t items from the last iteration of the while-loop, and $a = \frac{1}{k_t}$. Let $r_1 = a/\hat{a}$. Since $\frac{1}{k_t} \geq \hat{a}$, $r_1 \geq 1$. Then,

$$\text{ALG}(\sigma) \leq k_t - \left\lfloor \frac{1}{\hat{a}e} \right\rfloor - b - 1 < \text{OPT}(\sigma) - \frac{1}{\hat{a}e} - b = \text{OPT}(\sigma) - \frac{r_1}{ae} - b$$

$$= \text{OPT}(\sigma) - \frac{r_1}{e}\text{OPT}(\sigma) - b = \frac{e - r_1}{e}\text{OPT}(\sigma) - b,$$

contradicting that $\text{ALG}(\sigma) \geq \frac{e-r}{e}\text{OPT}(\sigma) - b$. Thus, the adversarial algorithm cannot terminate in Line 9.

Since the adversarial algorithm does not terminate in Line 9, it must accept its first item no later than in the $(b+2)$nd iteration of the while-loop, and the ith item accepted by ALG has size at least $\frac{1}{\left\lfloor \frac{1}{\hat{a}e} \right\rfloor + b + i} \geq \frac{1}{\frac{1}{\hat{a}e} + b + i}$. Thus, the total size, S_t, of the items accepted by ALG in the while-loop is

$$\begin{aligned} S_t &\geq \sum_{k=\frac{1}{\hat{a}e}+b+1}^{k_t} \frac{1}{k} \\ &= \sum_{k=1}^{k_t} \frac{1}{k} - \sum_{k=1}^{\frac{1}{\hat{a}e}-1} \frac{1}{k} - \sum_{k=\frac{1}{\hat{a}e}}^{\frac{1}{\hat{a}e}+b} \frac{1}{k} \\ &> H_{k_t} - H_{\frac{1}{\hat{a}e}-1} - (b+1)\hat{a}e \\ &\geq \ln(k_t) - \ln\left(\frac{1}{\hat{a}e}\right) - (b+1)\hat{a}e, \qquad \text{by Lemma 1.} \end{aligned}$$

Thus, we have

$$S_t > \ln(k_t) - \ln\left(\frac{1}{\hat{a}e}\right) - (b+1)\hat{a}e\,. \tag{4}$$

By the first condition of the while-loop, and since ALG accepts at most one item per iteration, $S_t \leq 1-(b+1)\hat{a}e$. By Ineq. (4), this means that $\ln(k_t) - \ln\left(\frac{1}{\hat{a}e}\right) - (b+1)\hat{a}e < 1 - \frac{1}{k_t} - (b+1)\hat{a}e$, and we get

$$\begin{aligned} &\ln(k_t) - \ln\left(\frac{1}{\hat{a}e}\right) - (b+1)\hat{a}e < 1 - (b+1)\hat{a}e \\ \Leftrightarrow\ &\ln\left(\frac{k_t}{\frac{1}{\hat{a}e}}\right) < 1 \\ \Leftrightarrow\ &\frac{k_t}{\frac{1}{\hat{a}e}} < e \\ \Leftrightarrow\ &k_t < \frac{1}{\hat{a}}\,. \end{aligned} \tag{5}$$

Furthermore, by the conditions of the while-loop, we have that $S_t > 1 - \frac{1}{k_t+1} - (b+1)\hat{a}e$ or $\frac{1}{k_t+1} < \hat{a}$. If $S_t > 1 - \frac{1}{k_t+1} - (b+1)\hat{a}e$, then, using that $k_t \geq \left\lfloor \frac{1}{\hat{a}e} \right\rfloor$, $S_t > 1 - \frac{1}{\left\lfloor \frac{1}{\hat{a}e} \right\rfloor + 1} - (b+1)\hat{a}e > 1 - \frac{1}{\frac{1}{\hat{a}e}} - (b+1)\hat{a}e = 1 - \hat{a}e - (b+1)\hat{a}e = 1 - (b+2)\hat{a}e$.

Otherwise, we get

$$\begin{aligned}
&\frac{1}{k_t+1} < \hat{a} \\
\Leftrightarrow\ & k_t > \frac{1}{\hat{a}} - 1\,.
\end{aligned} \tag{6}$$

Plugging this into Ineq. (4), we get

$$\begin{aligned}
S_t &> \ln\left(\frac{1}{\hat{a}}-1\right) - \ln\left(\frac{1}{\hat{a}e}\right) - (b+1)\hat{a}e \\
&= \ln\left(\frac{\frac{1}{\hat{a}}-1}{\frac{1}{\hat{a}e}}\right) - (b+1)\hat{a}e \\
&= \ln\left(e-\hat{a}e\right) - (b+1)\hat{a}e \\
&= 1+\ln(1-\hat{a}) - (b+1)\hat{a}e \\
&> 1-(b+2)\hat{a}e,\ \text{since } \hat{a} < \frac{1}{2}\,.
\end{aligned}$$

Thus, in either case, we get $S_t > 1-(b+2)\hat{a}e$. Therefore, the algorithm can fit at most $\frac{(b+2)\hat{a}e}{r_2\hat{a}} = \frac{(b+2)e}{r_2}$ of the items of size $r_2\hat{a}$ into its knapsack, where r_2 is one of the parameters to the adversarial algorithm as defined in Algorithm 6. Since ALG packs at most one item per iteration of the while-loop, this means that

$$\begin{aligned}
\textsc{Alg}(\sigma) &\le k_t - \left\lfloor \frac{1}{\hat{a}e} \right\rfloor + 1 + \frac{(b+2)e}{r_2} \\
&< k_t - \frac{1}{\hat{a}e} + 2 + \frac{(b+2)e}{r_2} \\
&< \frac{1}{\hat{a}} - \frac{1}{\hat{a}e} + 2 + \frac{(b+2)e}{r_2},\ \text{by Ineq. (5)} \\
&= \frac{e-1}{\hat{a}e} + 2 + \frac{(b+2)e}{r_2} \\
&= r_2\frac{e-1}{e}\textsc{Opt}(\sigma) + 2 + \frac{(b+2)e}{r_2}\,.
\end{aligned}$$

For any $r < 1$, this yields an upper bound on the competitive ratio of $r\frac{e-1}{e}$, since for any given r, $2+\frac{(b+2)e}{r}$ is a constant. □

We also note the contrapositive version of this theorem, i.e., that if a deterministic algorithm is better than $r \cdot \frac{e-1}{e}$-competitive for some $r \le 1$, it cannot be $\frac{e-r}{e}$-competitive for all $1 \le r < e$.

5.1.3 The Competitive Ratio of CAT

Combining the positive result from Theorem 6 with the negative result from Theorem 7, we obtain that, if r is guaranteed to be smaller than e, no deterministic algorithm can be better than CAT for both $r < 1$ and $r > 1$. Moreover, we get the following tight result on the performance of CAT.

Theorem 8 CAT *has a competitive ratio of*

$$c_{\text{CAT}}(r) = \begin{cases} r \cdot \dfrac{e-1}{e}, & \text{if } r \leq 1 \\ \dfrac{e-r}{e}, & \text{if } 1 \leq r \leq e \\ 0, & \text{if } r \geq e. \end{cases}$$

Proof For different cases depending on r, we want to determine the exact value of $c_{\text{CAT}}(r)$. We do this by proving upper and lower bounds on those values, case by case. The lower bounds have already been established in Theorem 6, except for the case $r \geq e$. However, for any case, the competitive ratio is non-negative.

Since CAT is $\frac{e-r}{e}$-competitive for $1 \leq r < e$, the upper bound for $r < 1$ follows from Theorem 7.

For $1 \leq r < e$, consider an input sequence, σ, consisting of ℓ items of size $\hat{a}e + \varepsilon_\ell$ followed by s items of size ε_s, where

$$\ell = \frac{1}{\hat{a}e} - 1,\ s = \frac{1}{\hat{a}r} - \frac{1}{\hat{a}e} + 1,\ \varepsilon_s = \frac{\hat{a}e - \ell\varepsilon_\ell}{s},\ \text{and } \varepsilon_\ell < \frac{\hat{a}e}{\ell}.$$

The number of items in σ is $\ell + s = \frac{1}{r\hat{a}}$, and their total size is

$$\ell(\hat{a}e + \varepsilon_\ell) + s\varepsilon_s = \ell\hat{a}e + \ell\varepsilon_\ell + \hat{a}e - \ell\varepsilon_\ell = (\ell + 1)\hat{a}e = 1.$$

Thus, the average size of the items in σ is $r\hat{a} = a$ and $\text{OPT}(\sigma) = \frac{1}{r\hat{a}}$. CAT accepts only the small items, so

$$\text{CAT}(\sigma) = \frac{1}{\hat{a}r} - \frac{1}{\hat{a}e} + 1 = \frac{e-r}{e}\frac{1}{r\hat{a}} + 1 = \frac{e-r}{e}\text{OPT}(\sigma) + 1.$$

For $r > e$, consider the input sequence consisting of $\frac{1}{r\hat{a}}$ items of size $r\hat{a}$. OPT accepts all items and CAT accepts none. □

5.2 Predictions with No Guarantee

In this section, we again consider predictions but now with no restriction on how large r can be.

5.2.1 Positive Result–Lower Bound

When considering the case where the average size of items in OPT_s is estimated to be $\hat{a}$, and the accurate value is $a = r \cdot \hat{a}$, we consider two cases, $r > 1$ and $r < 1$. In either case, we have the problem that we do not even know which case we are in, so, when large items arrive, we have to accept some to be competitive. The algorithm we consider when the value of r is not necessarily one achieves similar competitive ratios in both cases. Algorithm 7, RAT, is ADAPTIVE THRESHOLD with a different threshold function than was used for more accurate predictions in CAT.

Algorithm 7 ROBUST ADAPTIVE THRESHOLD, RAT.

```
1: Define $T(i) = \sqrt{\frac{\hat{a}}{2i}}$ for $i \geq 1$
2: Run ADAPTIVE THRESHOLD, Algorithm 3
```

Since we need to accept larger items than in the case of accurate predictions, we need a threshold function that decreases faster than the threshold function used in Sect. 4, in order not to risk filling up the knapsack before the small items arrive. Therefore, it may seem surprising that we are using a threshold function that decreases as $\frac{1}{\sqrt{i}}$, when the threshold function of Sect. 4 decreases as $\frac{1}{i}$. However, the $\frac{1}{i}$-function of the algorithm for accurate predictions is essentially offset by $\frac{1}{ae}$.

We prove a number of more or less technical results before stating the positive results for $r \leq 1$ (Theorem 10) and $r \geq 1$ (Theorem 9).

Lemma 3 *For any $k \geq 1$, the total size of the k largest items accepted by* RAT *is at most $\sqrt{2k\hat{a}}$.*

Proof RAT calls Algorithm 3, and by the condition in its **if**-statement, as soon as i items of size greater than $T(i+1)$ have been accepted, no more items larger than $T(i+1)$ are accepted after that. Thus, for each $i \geq 0$, at most i items of size greater than $\sqrt{\frac{\hat{a}}{2(i+1)}}$ are accepted. This means that the ith largest item accepted by RAT has size at most $\sqrt{\frac{\hat{a}}{2i}}$. Thus, the total size of the k largest accepted items is bounded by

$$\sum_{i=1}^{k} \sqrt{\frac{\hat{a}}{2i}} \leq \sqrt{\frac{\hat{a}}{2}} \int_0^k \frac{1}{\sqrt{i}} di = \sqrt{\frac{\hat{a}}{2}} \cdot 2\sqrt{k} = \sqrt{2k\hat{a}},$$

since $f(i) = \frac{1}{\sqrt{i}}$ is a decreasing function. □

Corollary 1 *If* RAT *rejects an item based on the level being too high, it has accepted at least $\lfloor \frac{1}{2\hat{a}} \rfloor$ items.*

Proof If RAT has accepted k items when it receives an item with a size no larger than the current bound, $T(i+1)$, that does not fit in the knapsack, then by Lemma 3,

$\sqrt{2(k+1)\hat{a}} > 1$. Now,

$$\sqrt{2(k+1)\hat{a}} > 1 \Leftrightarrow k > \frac{1}{2\hat{a}} - 1 \Rightarrow k \geq \left\lfloor \frac{1}{2\hat{a}} \right\rfloor .$$

□

The following corollary implies that RAT never rejects an item based on the level being too high if $r > 2$. This is because $r > 2$ means that the items in Opt are relatively large compared to $\hat{a}$. Since Opt accepts the smallest items of the sequence, it means that the sequence contains relatively few small items. Thus, the algorithm reserves space for small items that never arrive.

Corollary 2 *If* RAT *rejects an item based on the level being too high,* RAT$(\sigma) > \frac{r}{2}$Opt$(\sigma) - 1$.

Proof By Corollary 1,

$$\text{RAT}(\sigma) > \frac{1}{2\hat{a}} - 1 = \frac{r}{2} \cdot \frac{1}{r\hat{a}} - 1 \geq \frac{r}{2}\text{Opt}(\sigma) - 1 .$$

□

For proving Theorems 9 and 10 below, we need Lemmas 4 and 5. Lemma 4 is a technical lemma, the proof of which can be found in Appendix A.

Lemma 4 *Assume that* $r, \hat{a}, q > 0$, $i \geq 0$, *and* $\ell < \sqrt{\frac{2(i+1)}{\hat{a}}}$. *If*

$$2r\sqrt{2\hat{a}(i+1)} - 2r\hat{a}(i+1) + q - 2 \leq 0,$$

then

$$\frac{(i+1) + \left(\frac{1}{r\sqrt{2\hat{a}(i+1)}} - 1\right)\ell}{\frac{\ell}{r\sqrt{2\hat{a}(i+1)}}} > \frac{q}{2} .$$

Lemma 5 *Assume that* Opt *accepts* ℓ *items larger than* $\sqrt{\frac{\hat{a}}{2(i+1)}}$ *and* s *items of size at most* $\sqrt{\frac{\hat{a}}{2(i+1)}}$, $i \geq 0$. *Then, the following inequalities hold:*

1 $s > \ell\left(\frac{1}{r\sqrt{2\hat{a}(i+1)}} - 1\right)$, *and*
2 $\ell < \sqrt{\frac{2(i+1)}{\hat{a}}}$.

Proof Since Opt's accepted items have average size a, we have that

$$r\hat{a} = a > \frac{\ell \cdot \sqrt{\frac{\hat{a}}{2(i+1)}} + s \cdot 0}{\ell + s},$$

and, equivalently,

$$s > \ell\left(\frac{1}{r\sqrt{2\hat{a}(i+1)}} - 1\right).$$

In addition, since OPT accepts ℓ items larger than $\sqrt{\frac{\hat{a}}{2(i+1)}}$, $\ell\sqrt{\frac{\hat{a}}{2(i+1)}} < 1$, so

$$\ell < \sqrt{\frac{2(i+1)}{\hat{a}}}.$$

□

For the case where the actual average size in OPT$_s$ is at least as large as the predicted average size, we get the following result:

Theorem 9 *For all request sequences σ, such that $r \geq 1$,*

$$\mathrm{RAT}(\sigma) \geq \frac{1}{2r}\mathrm{OPT}(\sigma) - 1\,.$$

Proof By Corollary 2, if RAT rejects an item in σ due to the knapsack not having room for the item, $\mathrm{RAT}(\sigma) \geq \frac{r}{2}\mathrm{OPT}(\sigma) - 1 \geq \frac{1}{2r}\mathrm{OPT}(\sigma) - 1$ for $r \geq 1$.

Now, suppose that RAT does not reject any item due to it not fitting in the knapsack. If RAT is not optimal, it must reject due to the size of the item. Let i_t denote the final value of i when the algorithm is run. This means that RAT has accepted i_t items of size greater than $\sqrt{\frac{\hat{a}}{2(i_t+1)}}$. We perform a case analysis based on whether this value is smaller or larger than $r\hat{a}$.

Case 1: $r \geq \frac{1}{\sqrt{2\hat{a}(i_t+1)}}$.

In this case, $i_t + 1 \geq \frac{1}{2r^2\hat{a}}$ and $\mathrm{OPT}(\sigma) \leq \frac{1}{r\hat{a}} \leq \sqrt{\frac{2(i_t+1)}{\hat{a}}}$. Thus,

$$\frac{\mathrm{RAT}(\sigma)+1}{\mathrm{OPT}(\sigma)} \geq \frac{i_t+1}{\sqrt{\frac{2(i_t+1)}{\hat{a}}}} = \sqrt{\frac{\hat{a}(i_t+1)}{2}} \geq \sqrt{\frac{\hat{a}\frac{1}{2r^2\hat{a}}}{2}} = \frac{1}{2r}.$$

Therefore, $\mathrm{RAT}(\sigma) \geq \frac{1}{2r}\mathrm{OPT}(\sigma) - 1$.

Case 2: $r < \frac{1}{\sqrt{2\hat{a}(i_t+1)}}$.

Suppose OPT accepts ℓ items larger than $\sqrt{\frac{\hat{a}}{2(i_t+1)}}$ and s items of size at most $\sqrt{\frac{\hat{a}}{2(i_t+1)}}$. Note that RAT also accepts the s items of size at most $\sqrt{\frac{\hat{a}}{2(i_t+1)}}$, since we are in the case where it does not reject items because of the knapsack being too full. Given the input sequence σ, we consider the ratio

$$\frac{\mathrm{RAT}(\sigma)+1}{\mathrm{OPT}(\sigma)} \geq \frac{(i_t+1)+s}{\ell+s}\,. \tag{7}$$

The result follows if this ratio is always at least $\frac{1}{2r}$.

Subcase 2a: $i_t + 1 \geq \frac{1}{2\hat{a}}$.

In this case, $\text{RAT}(\sigma) \geq i_t \geq \frac{1}{2\hat{a}} - 1$, while $\text{OPT}(\sigma) \leq \frac{1}{r\hat{a}}$. Thus, $\text{RAT}(\sigma) \geq \frac{r}{2}\text{OPT}(\sigma) - 1$.

Subcase 2b: $i_t + 1 < \frac{1}{2\hat{a}}$.

By Ineq. (7) and Item 1 of Lemma 5, and since $\frac{\text{RAT}(\sigma)+1}{\text{OPT}(\sigma)} \leq 1$,

$$\frac{\text{RAT}(\sigma) + 1}{\text{OPT}(\sigma)} \geq \frac{(i_t + 1) + s}{\ell + s} \geq \frac{(i_t + 1) + \left(\frac{1}{r\sqrt{2\hat{a}(i_t+1)}} - 1\right)\ell}{\frac{\ell}{r\sqrt{2\hat{a}(i_t+1)}}}.$$

We now show that this is at least $\frac{1}{2r}$. From our case conditions, $i_t + 1 < \frac{1}{2\hat{a}}$ and $1 \leq r < \frac{1}{\sqrt{2\hat{a}(i_t+1)}}$, we get that $\frac{1}{r^2} > 2\hat{a}(i_t + 1)$ and $0 < 2\hat{a}(i_t + 1) < 1$. Consider the function

$$f(r) = 2r\sqrt{2\hat{a}(i_t + 1)} - 2r\hat{a}(i_t + 1) + \frac{1}{r} - 2.$$

Taking the derivative with respect to r gives

$$f'(r) = 2\sqrt{2\hat{a}(i_t + 1)} - 2\hat{a}(i_t + 1) - \frac{1}{r^2}.$$

Setting this equal to zero and solving for r, we find

$$r^* = \frac{1}{\sqrt{2\sqrt{2\hat{a}(i_t + 1)} - 2\hat{a}(i_t + 1)}}.$$

The possible maximum value for $f(r)$ in the range for r is then at 1, r^*, or $\frac{1}{\sqrt{2\hat{a}(i_t+1)}}$. For all three values, $f(r) \leq 0$. The hardest (but still simple) case is for $r = r^*$, where

$$f(r^*) = \frac{2\sqrt{v} - v}{\sqrt{2\sqrt{v} - v}} + \sqrt{2\sqrt{v} - v} - 2,$$

where we let v denote $2\hat{a}(i_t + 1)$. Note that due to the subcase we are in, $0 < v < 1$. Now,

$$\frac{2\sqrt{v} - v}{\sqrt{2\sqrt{v} - v}} + \sqrt{2\sqrt{v} - v} - 2 \leq 0 \;\Leftrightarrow\; 2\sqrt{v} \leq v + 1 \;\Leftrightarrow\; 0 \leq (v - 1)^2,$$

which clearly holds. By Item 2 of Lemma 5, the result now follows from Lemma 4 with $q = \frac{1}{r}$. □

For the case where the actual average size in OPT_s is no larger than the predicted average size, we get the following result:

Theorem 10 *For all request sequences σ, such that $r < 1$,*

$$\mathrm{RAT}(\sigma) \geq \frac{r}{2}\mathrm{OPT}(\sigma) - 1.$$

Proof The proof follows that of the previous theorem.
Case 1. $i_t + 1 \geq \frac{1}{2\hat{a}}$.

Since $\ell \geq i_t + 1$ (otherwise RAT is optimal), RAT has accepted at least $\frac{1}{2\hat{a}} - 1$ items, while OPT can accept at most $\frac{1}{r\hat{a}}$. Thus, $\mathrm{RAT}(\sigma) \geq \frac{r}{2} \cdot \frac{1}{r\hat{a}} - 1 \geq \frac{r}{2}\mathrm{OPT}(\sigma) - 1$.
Case 2. $i_t + 1 < \frac{1}{2\hat{a}}$.

By Item 1 of Lemma 5 and since $\frac{\mathrm{RAT}(\sigma)+1}{\mathrm{OPT}(\sigma)} \leq 1$,

$$\frac{\mathrm{RAT}(\sigma)+1}{\mathrm{OPT}(\sigma)} \geq \frac{(i_t+1)+s}{\ell+s} \geq \frac{(i_t+1) + \left(\frac{1}{r\sqrt{2\hat{a}(i_t+1)}} - 1\right)\ell}{\frac{\ell}{r\sqrt{2\hat{a}(i_t+1)}}}.$$

We will show that this is at least $\frac{r}{2}$. Consider the function

$$f(r) = 2r\sqrt{2\hat{a}(i_t+1)} - 2r\hat{a}(i_t+1) + r - 2.$$

Taking the derivative with respect to r gives

$$f'(r) = 2\sqrt{2\hat{a}(i_t+1)} - 2\hat{a}(i_t+1) + 1,$$

which is positive, since by the case condition, $0 < 2\hat{a}(i_t+1) < 1$. Thus, $f(r)$ is an increasing function for the values of $\hat{a}$, i_t+1, and r considered in this case, so the maximum value is at the maximum value of r, $r = 1$, giving that

$$f(r) \leq 2r\sqrt{2\hat{a}(i_t+1)} - 2r\hat{a}(i_t+1) + r - 2 < 0.$$

By Item 2 of Lemma 5, the result now follows from Lemma 4 with $q = r$. □

5.2.2 General Negative Result–Upper Bound

In Sect. 4, we showed that, even with accurate predictions, no deterministic algorithm can be better than $\frac{e-1}{e}$-competitive. In this section, we give a trade-off in the competitive ratio attainable by any algorithm for different values of r. Theorem 11 below leads to Corollary 3, which shows that if an algorithm, ALG, performs as well as RAT for all $1 \leq r \leq \frac{1}{2\hat{a}}$, then ALG cannot perform better than RAT for any $r \leq 1$. The theorem gives a more general result using a parameter z, which, if set equal to 2, gives the trade-off just described.

Theorem 11 *Let $0 < z \leq 2$ and consider a deterministic algorithm,* ALG. *If* ALG *is $\frac{1}{zr}$-competitive for every r between $\frac{2}{z}$ and $\frac{1}{\sqrt{z\hat{a}}}$, its competitive ratio is at most $\frac{zr}{4}$, for every $r \leq \frac{2}{z}$.*

Proof We consider the adversary that gives the input sequence σ_z defined by Algorithm 8.

Algorithm 8 Adversarial sequence establishing trade-off in performance with respect to r; the adversarial algorithm takes parameters, z, q, and b, such that $0 < z \leq 2$, $0 < q < \frac{1}{\sqrt{z\hat{a}}}$, and $b \geq 0$.

```
   ▷ Assume $\frac{1}{q\hat{a}} \in \mathbb{N}$
1: $p \leftarrow \lfloor \frac{z}{4\hat{a}} \rfloor$
2: $k \leftarrow 0$
3: while $k \leq p - 1$ do
4:     $k$++
5:     Give $\left\lfloor \sqrt{\frac{zk}{\hat{a}}} \right\rfloor$ items of size $\sqrt{\frac{\hat{a}}{zk}}$
6:     if ALG has accepted fewer than $k - b$ items then terminate
7: Give $\frac{1}{q\hat{a}}$ items of size $q\hat{a}$
```

Consider an online algorithm, ALG, and assume that there exists a constant, b, such that $\text{ALG}(\sigma) \geq \frac{1}{zr}\text{OPT}(\sigma) - b$, for any sequence σ and any r such that $\frac{2}{z} \leq r \leq \frac{1}{\sqrt{z\hat{a}}}$. Now, consider the adversary that gives the input sequence σ_z defined by Algorithm 8.

If the adversarial algorithm terminates in Line 6, then, ALG has accepted at most $k - b - 1$ items. In this case, $a = \sqrt{\frac{\hat{a}}{zk}}$, and OPT accepts exactly the $\left\lfloor \sqrt{\frac{zk}{\hat{a}}} \right\rfloor$ items from the last iteration of the while-loop. Since $a = r\hat{a}$, $r = \sqrt{\frac{1}{zk\hat{a}}}$, which lies between $\sqrt{\frac{1}{zp\hat{a}}} \geq \sqrt{\frac{1}{z\hat{a}} \cdot \frac{4\hat{a}}{z}} = \frac{2}{z}$ and $\frac{1}{\sqrt{z\hat{a}}}$. Thus,

$$\begin{aligned}
\text{ALG}(\sigma_z) \;&\leq\; k - b - 1 \;\leq\; \frac{k-1}{\left\lfloor \sqrt{\frac{zk}{\hat{a}}} \right\rfloor}\text{OPT}(\sigma_z) - b \;<\; \frac{k-1}{\sqrt{\frac{zk}{\hat{a}}} - 1}\text{OPT}(\sigma_z) - b \\
&<\; \frac{k}{\sqrt{\frac{zk}{\hat{a}}}}\text{OPT}(\sigma_z) - b \;=\; \sqrt{\frac{k\hat{a}}{z}}\text{OPT}(\sigma_z) - b \;=\; \frac{1}{zr}\text{OPT}(\sigma_z) - b\,,
\end{aligned}$$

where the second strict inequality holds because 1 is added to the numerator and denominator of a positive fraction less than 1. This contradicts the assumption that for each r between $\frac{2}{z}$ and $\frac{1}{\sqrt{z\hat{a}}}$, $\text{ALG}(\sigma) \geq \frac{1}{zr}\text{OPT}(\sigma) - b$, for any sequence σ, when the adversarial algorithm terminates in Line 6. Thus, the adversarial algorithm does not terminate there.

If the adversarial algorithm does not terminate in Line 6, $r = q$ and $\text{OPT}(\sigma_z) = \frac{1}{q\hat{a}} = \frac{1}{r\hat{a}}$. Moreover, for ALG, the ith accepted item must have size at least $\sqrt{\frac{\hat{a}}{z(i+b)}}$, for $1 \le i \le p-b$. Thus, these first $p-b$ items fill the knapsack to at least

$$\sum_{i=b+1}^{p} \sqrt{\frac{\hat{a}}{zi}} \ge \sqrt{\frac{\hat{a}}{z}} \int_{b+1}^{p+1} \frac{1}{\sqrt{i}} di = \sqrt{\frac{\hat{a}}{z}} (2\sqrt{p+1} - 2\sqrt{b+1}),$$

where we use that $\frac{1}{\sqrt{i}}$ is a decreasing function.

Since the items of size $r\hat{a}$ are the smallest items of the sequence, this means that

$$\begin{aligned}
\textsc{Alg}(\sigma_z) &\le p + \frac{1 - \sqrt{\frac{\hat{a}}{z}}(2\sqrt{p+1} - 2\sqrt{b+1})}{r\hat{a}} \\
&\le \frac{z}{4\hat{a}} + \frac{1 - \sqrt{\frac{\hat{a}}{z}}\left(2\sqrt{\frac{z}{4\hat{a}}} - 2\sqrt{b+1}\right)}{r\hat{a}} \\
&= \frac{z}{4\hat{a}} + \frac{1 - 1 + 2\sqrt{\frac{\hat{a}(b+1)}{z}}}{r\hat{a}} \\
&= \frac{1}{r\hat{a}}\left(\frac{zr}{4} + 2\sqrt{\frac{\hat{a}(b+1)}{z}}\right) \\
&= \left(\frac{zr}{4} + 2\sqrt{\frac{\hat{a}(b+1)}{z}}\right)\textsc{Opt}(\sigma_z)\,.
\end{aligned}$$

As a function of $\hat{a}$, the upper bound is $\frac{zr}{4} + 2\sqrt{\frac{\hat{a}(b+1)}{z}}$, but the second term becomes insignificant as $\hat{a}$ approaches zero. □

We also note the contrapositive version of this theorem, i.e., if Alg is better than $\frac{zr}{4}$-competitive for some $r \le \frac{2}{z}$, it cannot be $\frac{1}{\sqrt{z\hat{a}}}$-competitive for all r between $\frac{2}{z}$ and $\frac{1}{\sqrt{z\hat{a}}}$.

Setting $z = 2$ in Theorem 11 demonstrates a Pareto-like optimality for RAT:

Corollary 3 *Consider a deterministic algorithm,* Alg.

If Alg *is $\frac{1}{2r}$-competitive for every r between 1 and $\frac{1}{\sqrt{2\hat{a}}}$, it has a competitive ratio of at most $\frac{r}{2}$, for every positive $r \le 1$.*

Moreover, if Alg *is better than $\frac{r}{2}$-competitive for some $r \le 1$, it cannot be $\frac{1}{2r}$-competitive for all r between 1 and $\frac{1}{\sqrt{2\hat{a}}}$.*

6 Advice Complexity

In this section, we briefly consider the Online Unit Profit Knapsack Problem in terms of advice complexity, concentrating on upper bounds on the number of bits, following the techniques in [6], used for the General Knapsack Problem with advice, and many other articles on advice complexity including [51, 52]: When representing (an approximation of) a number, x, the k most significant bits of x are given along with a number, z, representing the number of bits between these k bits and the binary point in x. Two different possible algorithms are explained, the first demonstrating how algorithms with predictions may be useful in defining algorithms with good advice complexity, since approximating values with few bits is related to having some limited error in a prediction. In addition, a non-constant lower bound on the number of bits needed for optimality is presented.

An algorithm based on CAT

The prediction given in the algorithms CAT and RAT is the value a, representing the average size of an item in $\textsc{Opt}_s$, and it could have some error. One could use CAT in the advice complexity setting, assuming that an oracle gives two values: z, the number of zeros between the binary point and the most significant bit in the binary representation of a, followed by s, the k most significant bits of a. In this case, the prediction $\hat{a}$ given for a is $\frac{s}{2^{z+k}}$. Since the high order bit of s is 1, this value is at least $\frac{1}{2^{z+1}}$. The error in the prediction, $\hat{a}$, is only due to the missing low order bits (assumed, possibly incorrectly, to be zero). The missing bits represent a number less than $\frac{1}{2^{z+k}}$. Thus, the ratio, r, in $a = r\hat{a}$ is in the range $1 \leq r \leq 1 + \frac{1}{2^{k-1}}$.

We denote by CATA the advice based version of CAT described above. Since $1 \leq r \leq 1 + \frac{1}{2^{k-1}} < e$, Theorem 6 gives the following guarantee.

$$\text{CAT}\textsc{a}(\sigma) \geq \frac{e - 1 - \frac{1}{2^{k-1}}}{e}\textsc{Opt}(\sigma), \text{ for all } \sigma.$$

Note that the length of the advice is independent of the length of the request sequence, though dependent on the values in that sequence. The value, z, and the bitstring, s, must be specified using self-delimiting encoding, since we do not know how many bits are used for them. For example, $\lceil \log(z+1) \rceil$ could be written in unary ($\lceil \log(z+1) \rceil$ ones, followed by a zero) before writing z itself in binary. Treating s similarly, at most $2(k + \lceil \log(z+1) \rceil + 1)$ bits are used. This gives us:

Proposition 1 CATA *has a competitive ratio of at least* $\frac{e-1-\frac{1}{2^{k-1}}}{e}$, *using at most* $2(k + \lceil \log(z+1) \rceil + 1)$ *bits of advice, when* $\frac{1}{2^{z+1}} \leq a < \frac{1}{2^z}$.

An algorithm based on techniques in [6]

As mentioned at the end of Sect. 1.3, since $\textsc{Opt}$ can be viewed as accepting a prefix of the sequence of items sorted in non-decreasing order of size, there is another obvious type of advice to give. Let the advice be k-bit approximations to both the size, S, of the largest item that $\textsc{Opt}_s$ accepts and the fraction, t, of the knapsack not filled with items of size strictly smaller than S. The approximation to S can be given using the technique above, specifying the number of leading zeros first and then k significant bits. For t, we simply use the first k bits after the binary point, letting t' be the number represented by those k bits. The reason that it is necessary to give the fraction of the knapsack not filled with items of at most this size is the following. Even if the exact value of S was given, it is unknown if OPT_s accepts one or many items of that size, and these "large" items could come before any smaller ones.

The algorithm, called ALG_S, will accept all items that are smaller than S, which is the optimal behavior on those items (so in the worst case for the performance ratio, no such items arrive). Using the notation s and z as above for approximating S, we are only interested in items of sizes between $\frac{s}{2^{z+k}}$ and $\frac{s+1}{2^{z+k}}$ and can calculate a bound on the competitive ratio just from the algorithm's and OPT_s's performance on items in that range. Since the algorithm does not accept all items in the worst case, we may assume that there are enough items in this size range that it rejects some. Under this assumption, the algorithm accepts at least $\left\lfloor \frac{t'/2^k}{(s+1)/2^{z+k}} \right\rfloor$ and OPT_s accepts at most $\left\lfloor \frac{(t'+1)/2^k}{s/2^{z+k}} \right\rfloor$. For an asymptotic result, ignoring the rounding down on the algorithm's performance, this gives a performance ratio of at least

$$\frac{t' \cdot s}{(s+1)(t'+1)} \geq \frac{2^k \cdot 2^{z+k}}{(2^k+1)(2^{z+k}+1)} \geq \frac{2^{2k}}{(2^k+1)^2}.$$

Since we approximate both S (using $2(k + \lceil \log(z+1) \rceil + 1)$ bits) and t (using $2k+1$ bits), we need a total of $4k + 2(\lceil \log(z+1) \rceil + 1)$ advice bits. This gives us:

Proposition 2 ALG_S *has a competitive ratio of at least* $\frac{2^{2k}}{(2^k+1)^2}$, *using at most* $4k + 2(\lceil \log(z+1) \rceil + 1)$ *bits of advice, when* $\frac{1}{2^{z+1}} \leq S < \frac{1}{2^z}$.

The competitive ratio with this approach is better than that of the first approach, but it also uses more advice.

Advice complexity for optimality

With respect to optimality, we note that the lower bound of $\log n$ from [6] for the general Knapsack Problem cannot be used directly here, since the items used in their sequences all have size 1, so the weights are very important. In contrast to the upper bounds proven above, we prove that for optimality, the number of advice bits needed is a function of the length, n, of the input sequence:

Theorem 12 *Any algorithm with advice, solving the Unit Profit Knapsack Problem to optimality, requires at least* $\log(n/3)$ *bits of advice.*

Proof For any algorithm, ALG, consider the set of input sequences defined to have length n as follows. Let $k \in \mathbb{N}$, $n = 3k$, and $0 \leq \ell \leq k$. Then I_ℓ consists of (in the order listed)

- k items of size $\frac{1}{k}$,
- $2(k-\ell)$ items of size $\frac{1}{2k}$,
- 2ℓ items of size 1.

Suppose for the sake of contradiction that ALG is optimal on all of these sequences and never reads $\log(n/3)$ bits of advice. OPT accepts ℓ items of size $\frac{1}{k}$ and then $2(k-\ell)$ items of size $\frac{1}{2k}$, completely filling up the knapsack with $2k - \ell$ items. Intuitively, the advice needs to say how many of the first k items to accept. Since there are $n/3$ sequences in all and fewer than $\log(n/3)$ bits of advice, there are at least two of the sequences, I_j and $I_{j'}$, for which ALG receives the same advice. Thus, ALG accepts the same number, j^*, of items of size $\frac{1}{k}$ on both I_j and $I_{j'}$. Without loss of generality, assume that $j^* \neq j'$. If $j^* > j'$, then ALG can accept only $2(k - j^*)$ items of size $\frac{1}{2k}$. In all, $\text{ALG}(I_{j'}) \leq j^* + 2(k - j^*) < 2k - j' = \text{OPT}(I_{j'})$. If $j^* < j'$, then $\text{ALG}(I_{j'}) \leq j^* + 2(k - j') < 2k - j' = \text{OPT}(I_{j'})$. Thus, ALG is not optimal on $I_{j'}$, giving a contradiction. □

A Proofs of Minor Technical Lemmas

Proof of Lemma 1 Define $\Delta_k = H_k - \ln k$. First, we argue that $\Delta_k > \Delta_{k+1}$.

Observe that

$$\ln(k+1) - \ln k = \int_k^{k+1} \frac{1}{x} dx > \frac{1}{k+1},$$

since $\frac{1}{k+1}$ is the smallest value we are integrating over. So, $\frac{1}{k+1} - \ln(k+1) < -\ln(k)$.

Using this,

$$\Delta_{k+1} = H_{k+1} - \ln(k+1) = H_k + \frac{1}{k+1} - \ln(k+1) < H_k - \ln k = \Delta_k.$$

By the definition of Δ_k,

$$\begin{aligned} H_k - H_p &= \ln k + \Delta_k - (\ln p + \Delta_p) \\ &\leq \ln k - \ln p, \text{ since, by induction,} \Delta_k \leq \Delta_p. \end{aligned}$$

From the integral, it follows similarly that $\ln(p+1) - \ln p < \frac{1}{p}$. Thus,

$$\begin{aligned} & \ln(p+1) - \ln p < \tfrac{1}{p} \\ \Leftrightarrow\ & H_{p-1} - H_p = -\tfrac{1}{p} < \ln p - \ln(p+1) \\ \Leftrightarrow\ & H_{p-1} - \ln p < H_p - \ln(p+1). \end{aligned}$$

Now, $H_k - \ln k > H_p - \ln(p+1)$ clearly holds for $p = k$, since ln is increasing. So, by induction, using the above in the induction step, it holds for smaller p as well. Thus, $\ln k - \ln(p+1) \leq H_k - H_p$ for $k \geq p$.

Proof of Lemma 2 We prove that $\frac{e - e^{1-ae}}{a}$ is bounded from above by e^2.

The derivative of the term is $\frac{ae^{2-ea} - (e - e^{1-ae})}{a^2} = \frac{e^{1-ae}(ea - e^{ea} + 1)}{a^2}$.

The terms a^2 and e^{1-ae} are positive. Consider the remaining term, $ea - e^{ea} + 1$. For $a = 0$, this term is zero. The derivative of ea is e and the derivative of e^{ea} is e^{ea+1}. For any $a > 0$, $e^{ea+1} > e$, so $ea - e^{ea} + 1$ is negative. Thus, for $a > 0$, the derivative of $\frac{e - e^{1-ae}}{a}$ is negative, and the term decreases with increasing a. Thus, the limit for a going towards zero is an upper bound.

Using L'Hôpital's rule, $\lim_{a\to 0^+} \frac{e-e^{1-ae}}{a} = \lim_{a\to 0^+} \frac{e^{2-ae}}{1} = e^2$.

Proof of Lemma 4

$$\begin{aligned}
& 2r\sqrt{2\hat{a}(i+1)} - 2r\hat{a}(i+1) + q - 2 \le 0 \\
\Leftrightarrow\ & 2r\sqrt{2\hat{a}(i+1)} - 2 + q \le 2r(i+1)\hat{a} \\
\Leftrightarrow\ & \sqrt{\tfrac{2(i+1)}{\hat{a}}}(2r\sqrt{2\hat{a}(i+1)} - 2 + q) \le 2r(i+1)\sqrt{2\hat{a}(i+1)} \\
\Leftrightarrow\ & \ell(2r\sqrt{2\hat{a}(i+1)} - 2 + q) < 2r(i+1)\sqrt{2\hat{a}(i+1)},\ \text{since } \ell < \sqrt{\tfrac{2(i+1)}{\hat{a}}} \\
\Leftrightarrow\ & \ell q < 2r(i+1)\sqrt{2\hat{a}(i+1)} + 2\ell - 2\ell r\sqrt{2\hat{a}(i+1)} \\
\Leftrightarrow\ & \frac{\ell q}{\sqrt{2\hat{a}(i+1)}} < 2r(i+1) + \frac{2\ell}{\sqrt{2\hat{a}(i+1)}} - 2\ell r \\
\Leftrightarrow\ & \frac{\ell q}{\sqrt{2\hat{a}(i+1)}} < 2r((i+1) + \frac{\ell}{r\sqrt{2\hat{a}(i+1)}} - \ell) \\
\Leftrightarrow\ & \frac{q}{2} < \frac{(i+1) + \left(\frac{1}{r\sqrt{2\hat{a}(i+1)}} - 1\right)\ell}{\frac{\ell}{r\sqrt{2\hat{a}(i+1)}}}.
\end{aligned}$$

Acknowledgements This work was supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B. A preliminary extended abstract of some of these results was published by the same authors as "Online Unit Profit Knapsack with Untrusted Predictions" in the 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT), Leibniz International Proceedings in Informatics (LIPIcs) 227, Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, 20:1–20:17, 2022. The authors would like to thank the anonymous referees for their careful reading of a long manuscript and for valuable suggestions.

Author Contributions The authors contributed equally to this work in all of its phases.

Funding Open access funding provided by University of Southern Denmark All authors were supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B.

Data Availability No datasets or data repositories were used in this work.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. J. ACM **68**(4), 24–12425 (2021)

2. Purohit, M., Svitkina, Z., Kumar, R.: Improving online algorithms via ML predictions. In: 31st Annual Conference on Neural Information Processing Systems (NeurIPS), pp. 9661–9670. Curran Associates, Inc., Red Hook, New York (2018)
3. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin, Heidelberg (2004)
4. Cygan, M., Jeż, Ł, Sgall, J.: Online knapsack revisited. Theory Comput. Syst. **58**, 153–160 (2016)
5. Marchetti-Spaccamela, A., Vercellis, C.: Stochastic on-line knapsack problems. Math. Program. **68**, 73–104 (1995)
6. Böckenhauer, H.-J., Komm, D., Královič, R., Rossmanith, P.: The online knapsack problem: Advice and randomization. Theoret. Comput. Sci. **527**, 61–72 (2014)
7. Boyar, J., Favrholdt, L.M., Kudahl, C., Larsen, K.S., Mikkelsen, J.W.: Online algorithms with advice: a survey. ACM Comput. Surv. **50**(2), 19–11934 (2017)
8. Komm, D.: An Introduction to Online Computation. Springer, Berlin, Heidelberg (2016)
9. Zeynali, A., Sun, B., Hajiesmaili, M.H., Wierman, A.: Data-driven competitive algorithms for online knapsack and set cover. In: 35th AAAI conference on artificial intelligence (AAAI), 10833–10841. AAAI Press, Palo Alto, California (2021)
10. Zhou, Y., Chakrabarty, D., Lukose, R.M.: Budget constrained bidding in keyword auctions and online knapsack problems. In: 4th international workshop on internet and network economics (WINE). Lecture Notes in Computer Science, vol. 5385, pp. 566–576. Springer, Berlin, Heidelberg (2008)
11. Im, S., Kumar, R., Qaem, M.M., Purohit, M.: Online knapsack with frequency predictions. In: Pre-Proceedings of the 34th annual conference on neural information processing systems (NeurIPS), 2733–2743. Curran Associates, Inc., Red Hook, New York (2021)
12. Boyar, J., Favrholdt, L.M., Larsen, K.S., Nielsen, M.N.: The competitive ratio for on-line dual bin packing with restricted input sequences. Nordic J. Comput. **8**, 463–472 (2001)
13. Angelopoulos, S., Dürr, C., Jin, S., Kamali, S., Renault, M.P.: Online computation with untrusted advice. In: 11th Innovations in Theoretical Computer Science Conference (ITCS). LIPIcs, vol. 151, pp. 52–15215. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Saarbrücken/Wadern (2020)
14. Angelopoulos, S., Kamali, S., Shadkami, K.: Online bin packing with predictions. J. Artif. Intell. Res. **78**, 1111–1141 (2023)
15. Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. In: 35th international conference on machine learning (ICML), 80, 3302–3311. PMLR, London (2018)
16. Angelopoulos, S.: Online search with a hint. Inf. Comput. **295**, 105091 (2023)
17. Angelopoulos, S., Kamali, S., Zhang, D.: Online search with best-price and query-based predictions. In: 36th AAAI conference on artificial intelligence, 36, 9652–9660 (2022)
18. Bhaskara, A., Cutkosky, A., Kumar, R., Purohit, M.: Online learning with imperfect hints. In: 37th International Conference on Machine Learning (ICML). Proceedings of machine learning research, vol. 119, pp. 822–831. PMLR, London (2020)
19. Lee, R., Maghakian, J., Hajiesmaili, M.H., Li, J., Sitaraman, R.K., Liu, Z.: Online peak-aware energy scheduling with untrusted advice. ACM SIGEnergy Inf. Rev. **1**(1), 59–77 (2021)
20. Medina, A.M., Vassilvitskii, S.: Revenue optimization with approximate bid predictions. In: 30th annual conference on neural information processing systems (NIPS), pp. 1858–1866. Curran Associates, Inc., Red Hook, New York (2017)
21. Ahmadian, S., Esfandiari, H., Mirrokni, V., Peng, B.: Robust load balancing with machine learned advice. J. Mach. Learn. Res. **24**, 44–14446 (2023)
22. Angelopoulos, S., Kamali, S.: Contract scheduling with predictions. J. Artif. Intell. Res. **77**, 396–426 (2023)
23. Azar, Y., Leonardi, S., Touitou, N.: Flow time scheduling with uncertain processing time. In: 53rd Annual ACM SIGACT symposium on theory of computing (STOC), 1070–1080. ACM, New York (2021)
24. Azar, Y., Leonardi, S., Touitou, N.: Distortion-oblivious algorithms for minimizing flow time. In: 33rd ACM-SIAM symposium on discrete algorithms (SODA), 252–274. SIAM, Philadelphia (2022)
25. Balkanski, E., Gkatzelis, V., Tan, X.: Strategyproof scheduling with predictions. In: Kalai, Y.T. (ed.) 14th Innovations in Theoretical Computer Science Conference (ITCS) 2023. LIPIcs, vol. 251, pp. 11–11122. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Saarbrücken/Wadern (2023)
26. Boyar, J., Favrholdt, L.M., Kamali, S., Larsen, K.S.: Online interval scheduling with predictions. In: 18th international symposium on algorithms and data structures (WADS). Lecture Notes in Computer Science, vol. 14079, pp. 193–207. Springer, Berlin, Heidelberg (2023)

27. Bamas, E., Maggiori, A., Rohwedder, L., Svensson, O.: Learning augmented energy minimization via speed scaling. In: 33rd annual conference on neural information processing systems (NeurIPS), 15350–15359. Curran Associates, Inc., Red Hook, New York (2020)
28. Im, S., Kumar, R., Qaem, M.M., Purohit, M.: Non-clairvoyant scheduling with predictions. ACM Trans. Parallel Comput. **10**(4), 19:1–19:26 (2022)
29. Kumar, A., Alam, B.: Task scheduling in real time systems with energy harvesting and energy minimization. J. Comput. Sci. **14**(8), 1126–1133 (2018)
30. Lattanzi, S., Lavastida, T., Moseley, B., Vassilvitskii, S.: Online scheduling via learned weights. In: 31st ACM-SIAM symposium on discrete algorithms (SODA), 1859–1877. SIAM, Philadelphia (2020)
31. Li, S., Xian, J.: Online unrelated machine load balancing with predictions revisited. In: 38th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 139, pp. 6523–6532. PMLR, London (2021)
32. Mitzenmacher, M.: Scheduling with Predictions and the Price of Misprediction. In: 11th Innovations in Theoretical Computer Science Conference (ITCS). LIPIcs, vol. 151, pp. 14–11418. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Saarbrücken/Wadern (2020)
33. Gollapudi, S., Panigrahi, D.: Online algorithms for rent-or-buy with expert advice. In: 36th international conference on machine learning (ICML). Proceedings of Machine Learning Research, vol. 97, pp. 2319–2327. PMLR, London (2019)
34. Kodialam, R.: Optimal algorithms for ski rental with soft machine-learned predictions. ArXiv (2019). arXiv:1903.00092 [cs.DS]
35. Wang, S., Li, J.: Online algorithms for multi-shop ski rental with machine learned predictions. In: 19th international conference on autonomous agents and multiagent systems (AAMAS), 2035–2037. International Foundation for Autonomous Agents and Multiagent Systems, Liverpool (2020)
36. Bansal, N., Coester, C., Kumar, R., Purohit, M., Vee, E.: Learning-augmented weighted paging. In: 33rd ACM-SIAM Symposium on Discrete Algorithms (SODA), 67–89. SIAM, Philadelphia (2022)
37. Indyk, P., Mallmann-Trenn, F., Mitrovic, S., Rubinfeld, R.: Online page migration with ML advice. In: 25th international conference on artificial intelligence and statistics (AISTATS). Proceedings of Machine Learning Research, vol. 151, pp. 1655–1670. PMLR, London (2022)
38. Jiang, Z., Panigrahi, D., Sun, K.: Online algorithms for weighted paging with predictions. ACM Trans. Algorithms **18**(4), 39:1–39:27 (2022)
39. Rohatgi, D.: Near-optimal bounds for online caching with machine learned advice. In: 31st ACM-SIAM symposium on discrete algorithms (SODA), 1834–1845. SIAM, Philadelphia (2020)
40. Wei, A.: Better and simpler learning-augmented online caching. In: approximation, randomization, and combinatorial optimization. algorithms and techniques (APPROX/RANDOM). LIPIcs, vol. 176, pp. 60–16017. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Saarbrücken/Wadern (2020)
41. Antoniadis, A., Coester, C., Eliás, M., Polak, A., Simon, B.: Online metric algorithms with untrusted predictions. ACM Trans. Algorithms **19**(2), 19–11934 (2023)
42. Antoniadis, A., Gouleakis, T., Kleer, P., Kolev, P.: Secretary and online matching problems with machine learned advice. Discrete Optim. **48**, 100778 (2023)
43. Banerjee, S., Gkatzelis, V., Gorokh, A., Jin, B.: Online Nash social welfare maximization with predictions. In: 33rd ACM-SIAM symposium on discrete algorithms (SODA),1–19. SIAM, Philadelphia (2022)
44. Mitzenmacher, M.: Queues with small advice. In: SIAM conference on applied and computational discrete algorithms (ACDA), 1–12. SIAM, Philadelphia (2021)
45. Rutten, D., Mukherjee, D.: Capacity scaling augmented with unreliable machine learning predictions. SIGMETRICS Perform. Eval. Rev. **49**(2), 24–26 (2022)
46. Azar, Y., Panigrahi, D., Touitou, N.: Online graph algorithms with predictions. In: 33rd ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 35–66. SIAM, Philadelphia (2022)
47. Bamas, E., Maggiori, A., Svensson, O.: The primal-dual method for learning augmented algorithms. In: 33rd annual conference on neural information processing systems (NeurIPS), pp. 20083–20094. Curran Associates, Inc., Red Hook, New York (2020)
48. Lavastida, T., Moseley, B., Ravi, R., Xu, C.: Learnable and instance-robust predictions for online matching, flows and load balancing. In: 29th Annual European Symposium on Algorithms (ESA). LIPIcs, vol. 204, pp. 59–15917. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Saarbrücken/Wadern (2021)
49. Wei, A., Zhang, F.: Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In: 33rd annual conference on neural information processing systems (NeurIPS) (2020)

50. Mitzenmacher, M., Vassilvitskii, S.: Algorithms with predictions. In: Roughgarden, T. (ed.) Beyond the Worst-Case Analysis of Algorithms, pp. 646–662. Cambridge University Press, Cambridge (2021)
51. Angelopoulos, S., Dürr, C., Kamali, S., Renault, M.P., Rosén, A.: Online bin packing with advice of small size. Theory Comput. Syst. **62**(8), 2006–2034 (2018)
52. Christ, M.G., Favrholdt, L.M., Larsen, K.S.: Online Multi-Coloring with Advice. Theoret. Comput. Sci. **596**, 79–91 (2015)