Online Unit Profit Knapsack with Untrusted Predictions

Joan Boyar
University of Southern Denmark, Odense, Denmark

Lene M. Favrholdt
University of Southern Denmark, Odense, Denmark

Kim S. Larsen
University of Southern Denmark, Odense, Denmark

Abstract

A variant of the online knapsack problem is considered in the settings of trusted and untrusted predictions. In Unit Profit Knapsack, the items have unit profit, and it is easy to find an optimal solution offline: Pack as many of the smallest items as possible into the knapsack. For Online Unit Profit Knapsack, the competitive ratio is unbounded. In contrast, previous work on online algorithms with untrusted predictions generally studied problems where an online algorithm with a constant competitive ratio is known. The prediction, possibly obtained from a machine learning source, that our algorithm uses 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. The algorithm presented achieves a competitive ratio of $\frac{1}{2r}$ for $r \geq 1$ and $\frac{r}{2}$ for $r \leq 1$. Using an adversary technique, we show that this is optimal in some sense, giving a trade-off in the competitive ratio attainable for different values of r. Note that the result for accurate advice, $r = 1$, is only $\frac{1}{2}$, but we show that no deterministic algorithm knowing the value a can achieve a competitive ratio better than $\frac{e-1}{e} \approx 0.6321$ and present an algorithm with a matching upper bound. We also show that this latter algorithm attains a competitive ratio of $r\frac{e-1}{e}$ for $r \leq 1$ and $\frac{e-r}{e}$ for $1 \leq r < e$, and no deterministic algorithm can be better for both $r < 1$ and $1 < r < e$.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, untrusted predictions, knapsack problem, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.SWAT.2022.20

Related Version *Full Version*: https://arxiv.org/abs/2203.00285

Funding Supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B.

1 Introduction

In this paper, we consider the Online Unit Profit Knapsack Problem: The request sequence consists of n item with sizes in $(0, 1]$. An online algorithm receives them 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 any 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 that fit in the knapsack is an optimal solution.

Even for this special case of the Knapsack Problem, no competitive online algorithms can exist. Thus, we study the problem under the assumption that (an approximation of) the average item size, a, in an optimal solution is known to the algorithm. We study the

18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022).
Editors: Artur Czumaj and Qin Xin; Article No. 20; pp. 20:1–20:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

case, where the exact value of a is given to the algorithm as advice by an oracle, as well as the case where a is untrusted, e.g., estimated using machine learning. For instance, the characteristics of the input may be different depending on the time of day the input is produced, which source produced the input, etc. This could be learned to some extent and result in a prediction, which could be provided to the algorithm.

When considering machine-learned advice, the concepts of consistency and robustness are often considered, describing the balance between performing well on accurate advice and not doing too poorly when the advice is completely wrong. Our setting is different from most work on online algorithms with machine-learned advice, where there is generally a known online algorithm with a constant competitive ratio for the problem without advice. For this problem, if the advice is completely wrong, the algorithm cannot be competitive, since the problem without advice does not allow for competitive algorithms. Despite this hardness for the standard online version of the problem, we obtain results with untrusted predictions that are surprisingly consistent and robust.

1.1 Previous Work

The Knapsack Problem is well studied and comes in many variants; see Kellerer et al. [24]. Cygan et al. [18] 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. To verify this result, consider, for instance, the family of input sequences σ_j consisting of items of sizes $\frac{1}{i}$, $i = 1, 2, 3, \ldots, j$.

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*[1] of an item.

The Online Knapsack Problem was first studied by Marchetti-Spaccamela and Vercellis [34]; 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.

The Online Unweighted (or Simple) Knapsack Problem with advice was studied in [14]. This is also called the proportional or uniform case. In this version, the importance of each item is equal to 1. They show 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. (As mentioned later, they also considered the General Knapsack Problem in the advice model.) 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 [15], though many newer results for specific problems have been published since.

In [45], a knapsack problem is considered in a setting with machine-learned advice, with results incomparable to ours. In their setting, the General Knapsack Problem is considered, and results depend on upper and lower bounds on the importance of the items. The authors define limited classes of algorithms, based on a parameter, leading to some controlled degradation compared to an optimal competitive ratio. Within the defined classes, focus is then on tuning compared with historical data. Decisions to accept or reject an item are based on a threshold function based on the item's importance. Though the definition of this function is ad hoc, in the sense that it is not derived from some direct optimality criterion, it is well-motivated, aiming to coincide with the behavior found in optimal algorithms for the standard online algorithms setting.

[1] This is sometimes called *value*, but we want to avoid confusion with other uses of the word.

Recently, in [21], the General Knapsack Problem is revisited, again with upper and lower bounds on the possible importance of items. Machine-learned advice is given for each importance v, both an upper and a lower bound for the sum of the sizes of the items with importance v. 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 results can be extended to the case where the predictions are off by a small amount, the lower bounds can be divided by $1 + \varepsilon$, and the upper bounds can be multiplied by $1 + \varepsilon$. This is in contrast to ours, where robustness results are proven for arbitrarily large errors in the predictions, but only a is predicted. Since we have no bounds on the ratio of the largest to smallest size, those values do not enter into our results. Their algorithm obtains what they prove to be the optimal competitive ratio (for the given predictions), up to an additive factor that goes to zero as the size of the largest item goes to zero; this result has some of the flavor of our negative result. The authors also consider two related problems.

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 [17]; the Unit Price Knapsack Problem is Dual Bin Packing with one bin. The standard Bin Packing Problem was considered with machine learning in [3], considering a model of machine learning where, for a given algorithm, ALG, they consider a pair of values, $(r_{\text{ALG}}, w_{\text{ALG}})$, representing worst case ratios compared to the optimal offline algorithm, OPT. The value r_{ALG} gives the ratio for the best (trusted) advice and w_{ALG} gives the ratio for the worst possible (untrusted) advice. They use a parameter α in their algorithm, and show that their algorithm achieves values $(r, f(r))$ with $1.5 < r \leq 1.75$ and $f(r) = \max\{33 - 18r, 7/4\}$.

Bin Packing is also studied in [5] in the standard setting for online algorithms with machine learning, giving a trade-off between consistency and robustness, with the performance degrading as a function of the prediction error. They also have experimental results. Since the problem is so difficult, they have restricted their consideration to integer item sizes.

Much additional work has been done for other online problems, studying variants with untrusted predictions (machine-learned advice, for instance), initiated by the work of Lykouris and Vassilvitskii [32, 33] and Purohit et al. [39] in 2018, with further work in the directions of search-like problems [2, 6, 13, 29, 30, 35], scheduling [1, 4, 9, 20, 26, 27, 31, 36], rental problems [19, 25, 42], caching/paging [12, 22, 23, 40, 43], and other problems [5, 7, 8, 11, 37, 41], while some papers attack multiple problems [3, 10, 28, 44]. For a survey, see [38].

1.2 Preliminaries

We let a denote the average size of items accepted by the offline, optimal algorithm, OPT, that accepts as many of the smallest items as possible. Moreover, we let $\hat{a}$ denote the "guessed" or predicted value of a. In the case of accurate advice (received from an oracle), $\hat{a} = a$. If $\hat{a}$ may not be accurate, possibly determined via machine learning, and therefore not necessarily exactly a, we define a ratio r such that $a = r \cdot \hat{a}$. This particular advice is considered as a value that might be available or predictable, and the competitive ratios we present are a function of r.

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, $\text{ALG}(\sigma) \geq c\,\text{OPT}(\sigma) - b$, where $\text{ALG}(\sigma)$ denotes ALG's profit on σ. ALG's competitive ratio is then $\sup\{c \mid \text{ALG is } c\text{-competitive}\}$. Note that this is a maximization problem and all competitive ratios are in the interval $[0, 1]$.

We use the notation $\mathbb{N} = \{0, 1, 2, \ldots\}$. Define $\sum_{i=x}^{y} f(k)$ for some function f and real-valued x and y such that $y - x \in \mathbb{N}$ as $f(x) + f(x+1) + \cdots + f(y)$. We generalize the Harmonic numbers by defining $H_k = \sum_{i=1+k-\lfloor k \rfloor}^{k} \frac{1}{k}$, for any real-valued $k \geq 1$. The following easy result is proven in the full paper [16].

▶ **Lemma 1.** *If $k \geq p \geq 1$ and $k - p \in \mathbb{N}$, then $\ln k - \ln(p+1) \leq H_k - H_p \leq \ln k - \ln p$.*

At any given time during the processing of the input sequence, the *level* of the knapsack denotes the total size of the items accepted.

1.3 Our Results

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

For our algorithm Adaptive Threshold (AT) where the advice 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 trusted predictions can do; the competitive ratio cannot be better than $\frac{e-1}{e} \approx 0.6321$ for $r = 1$.

If AT is used for untrusted predictions, it obtains a competititve 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 our algorithm, Adaptive Threshold for Untrusted Predictions (ATup) we have 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 advice, the competitive ratio of ATup is $\frac{1}{2}$, slightly less good 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 $r = 1$.

Exact, oracle-based advice is not our focus point, though it is a crucial step in our work towards an algorithm for untrusted predictions. Thus, we do not emphasize the direction of advice complexity, where the focus is on the number of bits of oracle advice used to obtain given competitive ratios (or optimality), but we include a brief discussion in the full paper [16]. Instead, we focus on advice that may be easy to obtain. It seems believable that the average size of requests in an optimal solution would be information easily obtainable. The average size is probably a crucial component with regards to the profit secured by a process and quite possibly crucial with regards to supplying resources (knapsacks) over time. It is a single number (or two numbers: number of items and total size) to collect and store, as opposed to more detailed information about a distribution. So little storage is required that one could keep multiple copies if, for instance, the expected average changes during the day.

Given the simple optimal algorithm for the offline version of unit price knapsack, it seems obvious to consider another possibility for advice, the maximum size, s, for items to accept. However, this is insufficient, as there might be many items of that size, but the optimal solution may contain very few of them. Thus, one also needs further advice, including, for example, the fraction of the knapsack filled by items of size s. With these parameters given as advice, there would be two possibilities for the error. An extension of this idea is presented in [14], where the minimum importance is used, instead of the maximum size, for the General Knapsack Problem, giving k-bit approximations to the advice.

The full paper [16] contains the proofs missing from this paper. We treat the trusted as well as the untrusted case, and they build on similar ideas. To make this exposition accessible, we have emphasized giving a full account of the simpler, the trusted case, here, and refer the reader to the full paper for the more technical elements from the untrusted case.

2 The Adaptive Threshold Algorithm

In Algorithm 1, we introduce an algorithm template, which can be used to establish an oracle-based advice algorithm as well as an algorithm for untrusted predictions. The template omits the definition of a threshold function, T, since it is different for the two algorithms. In both algorithms, the threshold functions have the property that $T(i) > T(i+1)$ for $i \geq 1$. We use the notation n_x to denote the number of accepted items strictly larger than x.

Algorithm 1 Algorithm Adaptive Threshold.

1: $\hat{a} \leftarrow$ predicted average size of Opt's accepted items
2: level $\leftarrow 0$
3: **for** each input item x **do**
4:     $i = \max_{j \geq 0} \left\{ n_{T(j+1)} = j \right\}$
5:     **if** $\text{size}(x) \leq T(i+1)$ **and** $\text{level} + \text{size}(x) \leq 1$ **then**
6:         Accept x
7:         level += size(x)
8:     **else**
9:         Reject x

Intuitively, Adaptive Threshold accepts items that fit as long as it has not accepted too many items larger than the current item. The threshold functions are used to determine how many larger items is too many; no more than i items of size larger than $T(i+1)$ are accepted. For smaller item sizes, this number of larger items is larger, since we need to accept more items if there are many small items.

Note that using $\max_{j \geq 0} \left\{ n_{T(j+1)} \geq j \right\}$ instead of $\max_{j \geq 0} \left\{ n_{T(j+1)} = j \right\}$ in Line 4 would result in the same algorithm. 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.

3 Accurate Predictions

In this section, we give an $\frac{e-1}{e}$-competitive algorithm which receives a, the average size of the items in Opt, as advice and prove that it is optimal among algorithms that get only a as advice.

3.1 Positive Result

To define an advice-based algorithm, we define a threshold function; see Algorithm 2. Throughout this section, we assume that $\hat{a} = a$, but the algorithm is also be used for untrusted predictions in Subsection 4.1.

First, we prove that AT with $\hat{a} = a$ has competitive ratio at least $\frac{e-1}{e} \approx 0.6321$.

▶ **Theorem 2.** *For $\hat{a} = a$, AT, as defined in Algorithm 2, is $\frac{e-1}{e}$-competitive.*

Algorithm 2 Adaptive Threshold with advice, AT.

1: Define $T(i) = \dfrac{\hat{a}e}{\hat{a}e(i-1)+1}$ for $i \geq 1$
2: Run Adaptive Threshold, Algorithm 1

Proof. If AT never rejects an item, it performs optimally. So assume it rejects an item at some point in the request sequence σ. Considering the conditional statement in the algorithm, if AT 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, AT rejects an item, x, because $\text{level} + \text{size}(x) > 1$.

The value of $T(k)$ from Algorithm 1 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 definitions of sums over non-integer values (see Section 1.2), this gives an upper bound on the total size of items accepted by AT of

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

Simple calculations (detailed in Lemma 1) give,

$$H_{\text{AT}(\sigma)+\frac{1}{ae}-1} - H_{\frac{1}{ae}-1} < \ln\left(\text{AT}(\sigma) + \frac{1}{ae} - 1\right) - \ln\left(\frac{1}{ae} - 1\right) = \ln\left(\frac{\text{AT}(\sigma)+\frac{1}{ae}-1}{\frac{1}{ae}-1}\right)$$

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

$$\ln\left(\frac{\text{AT}(\sigma)+\frac{1}{ae}-1}{\frac{1}{ae}-1}\right) > 1 - \text{size}(x)$$

$$\Updownarrow$$

$$\frac{\text{AT}(\sigma)+\frac{1}{ae}-1}{\frac{1}{ae}-1} > e^{1-\text{size}(x)}$$

$$\Updownarrow$$

$$\text{AT}(\sigma) > \left(\frac{1}{ae} - 1\right) e^{1-\text{size}(x)} - \frac{1}{ae} + 1$$

$$\Updownarrow$$

$$\text{AT}(\sigma) > \frac{e^{1-\text{size}(x)} - 1}{ae} + 1 - e^{1-\text{size}(x)}.$$

In the algorithm, i is at least zero, so we cannot accept items larger than $T(1) = ae$.

$$\begin{aligned}
\mathrm{AT}(\sigma) &> \frac{e^{1-\mathrm{size}(x)}-1}{ae} + 1 - e, && \text{since } -e^{1-\mathrm{size}(x)} > -e \\
&> \frac{e^{1-ae}-1}{ae} + 1 - e, && \text{by the observation above} \\
&\geq \frac{e-e^2a-1}{ae} + 1 - e, && \text{by simple calculcations; see the full paper [16]} \\
&= \frac{e-1}{ae} - 2e + 1 \\
&\geq \frac{e-1}{e}\,\textsc{Opt}(\sigma) - 2e + 1, \text{ since } \textsc{Opt}(\sigma) \leq \tfrac{1}{a}
\end{aligned}$$

So, $\lim_{\textsc{Opt}\to\infty} \frac{\mathrm{AT}(\sigma)}{\textsc{Opt}(\sigma)} \geq \frac{e-1}{e}$.

Case 2. This is the case where AT never rejects any item, x, when $\mathrm{size}(x) \leq T(i+1)$. 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, AT 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, AT has accepted all of the s items no larger than $T(i_t+1)$. To see this, note that the is 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 Opt's average size, a.

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

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

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 $\frac{e-1}{e}$:

$$\frac{\left(\frac{1}{T(i_t+1)} - \frac{1}{ae}\right) + \left(\frac{T(i_t+1)}{a} - 1\right)\ell}{\frac{T(i_t+1)}{a}\ell} \geq \frac{e-1}{e}$$

$$\Updownarrow$$

$$\frac{e}{T(i_t+1)} - \frac{1}{a} + e\left(\frac{T(i_t+1)}{a} - 1\right)\ell \geq e\frac{T(i_t+1)}{a}\ell - \frac{T(i_t+1)}{a}\ell$$

$$\Updownarrow$$

$$\frac{e}{T(i_t+1)} - \frac{1}{a} \geq \left(e - \frac{T(i_t+1)}{a}\right)\ell$$

$$\Updownarrow$$

$$\frac{ea - T(i_t+1)}{aT(i_t+1)} \geq \frac{ea - T(i_t+1)}{a}\ell$$

$$\Updownarrow$$

$$\frac{1}{T(i_t+1)} \geq \ell$$

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

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

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

3.2 Negative Result

Now, we show that AT is optimal among online algorithms knowing a and nothing else.

▶ **Theorem 3.** *Any deterministic algorithm getting only a as advice has a competitive ratio of at most $\frac{e-1}{e}$.*

Algorithm 3 Adversarial sequence establishing optimality with advice.

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

Proof. Let ALG denote the online algorithm with advice, and let σ be the adversarial sequence defined by Algorithm 3, which explains how the adversary defines its sequence based on ALG's actions.

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 advice.

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

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

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

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.

$$\sum_{k=\lfloor \frac{1}{ae} \rfloor}^{k_t} \left(\frac{1}{k} - \varepsilon\right) \leq 1 - (k_t+1)\varepsilon$$
$$\Downarrow$$
$$H_{k_t} - H_{\lfloor \frac{1}{ae} \rfloor - 1} - k_t\varepsilon < 1 - k_t\varepsilon$$
$$\Updownarrow$$
$$H_{k_t} - H_{\lfloor \frac{1}{ae} \rfloor - 1} < 1$$
$$\Downarrow$$
$$\ln(k_t) - \ln\left(\left\lfloor \frac{1}{ae} \right\rfloor\right) < 1, \text{ by Lemma 1}$$
$$\Updownarrow$$
$$k_t < e\left\lfloor \frac{1}{ae} \right\rfloor$$

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, $k_t \geq \lfloor \frac{1}{ae} \rfloor$.

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

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 $k_t - \lfloor \frac{1}{ae} \rfloor + 1$. Using $k_t < e\lfloor \frac{1}{ae} \rfloor$, we can now bound ALG's profit:

$$\text{ALG}(\sigma) \leq k_t - \left\lfloor \frac{1}{ae} \right\rfloor + 1 + 2 < (e-1)\left\lfloor \frac{1}{ae} \right\rfloor + 3 \leq \frac{e-1}{e}\frac{1}{a} + 3 = \frac{e-1}{e}\text{OPT}(\sigma) + 3$$

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 we have established that $k_t < e\lfloor \frac{1}{ae} \rfloor < \frac{1}{a}$. Additionally, from the start value of k, we know that $\lfloor \frac{1}{ae} \rfloor \leq k$. Using these facts, together with the assumption from the algorithm that $a < \frac{1}{2e}$, we get the following bounds on the various terms.

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

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

For the last relevant value, $1 - ka \geq 1 - k_t a > 1 - \frac{1}{a}a = 0$ 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. ◀

4 Untrusted Predictions

Some of our main results are found in this section, but the proofs build on ideas from the trusted case. To make the exposition accessible, we have emphasized explaining the ideas in the simpler setting. The reader is referred to the full paper [16] for the missing details.

For the case where the predictions may be inaccurate, the algorithm AT can be used with $\hat{a}$ possibly not being a as long as $r < e$, see Subsection 4.1. In Subsection 4.2, we give an adaptive threshold algorithm, ATUP, that works for all r.

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

4.1 Semi-Trusted Predictions

In this section, we consider the algorithm AT with a semi-trusted (being guaranteed that $r < e$) prediction, $\hat{a}$, instead of a. See the full paper [16] for the proofs.

Using a proof analogous to the proof of Theorem 2, we get the following result.

▶ **Theorem 4.** *For untrusted advice, AT has a competitive ratio of at least*

$$c_{\mathrm{AT}}(r) \geq \begin{cases} \dfrac{e-1}{e} \cdot r, & \text{if } r \leq 1 \\ \dfrac{e-r}{e}, & \text{if } r \geq 1 \end{cases}$$

▶ **Theorem 5.** *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 $r \leq 1$.*

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

Combining Theorems 4 and 5, we obtain that, if r is guaranteed to be smaller than e, no deterministic algorithm can be better than AT for both $r < 1$ and $r > 1$. Moreover, we get the following tight result on the performance of AT.

▶ **Theorem 6.** *AT has a competitive ratio of*

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

4.2 Untrusted Predictions

4.2.1 Positive Result

When considering the case where the average item size 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 4, ATUP, is ADAPTIVE THRESHOLD with a different threshold function than was used for accurate advice (and in AT).

Since we need to accept larger items than in the case of accurate advice, we need a threshold function that decreases faster than the threshold function used in Section 3, 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 Section 3 decreases as $\frac{1}{i}$. However, the $\frac{1}{i}$-function of the algorithm for accurate advice is essentially offset by $\frac{1}{ae}$.

Algorithm 4 Adaptive Threshold for Untrusted Predictions, ATup.

1: Define $T(i) = \sqrt{\frac{\hat{a}}{2i}}$ for $i \geq 1$
2: Run Adaptive Threshold, Algorithm 1

We prove a number of more or less technical results before stating the positive results for $r \leq 1$ (Theorem 11) and $r \geq 1$ (Theorem 10). See the full paper [16] for those missing here, along with the missing proofs.

▶ **Lemma 7.** *For any $k \geq 1$, the total size of the k largest items accepted by ATup is at most $\sqrt{2k\hat{a}}$.*

The following corollary implies that ATup 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 8.** *If ATup rejects an item based on the level being too high, ATup$(\sigma) > \frac{r}{2}$ Opt$(\sigma) - 1$.*

▶ **Lemma 9.** *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. $s > \ell \left(\dfrac{1}{r\sqrt{2\hat{a}(i+1)}} - 1 \right)$
2. $\ell < \sqrt{\dfrac{2(i+1)}{\hat{a}}}$

For the case where the actual average size in Opt's packing is at least as large as the predicted average size, we get the following result.

▶ **Theorem 10.** *For all request sequences σ, such that $r \geq 1$,*

$$\text{ATup}(\sigma) \geq \frac{1}{2r}\, \text{Opt}(\sigma) - 1\,.$$

Proof. By Corollary 8, if ATup rejects an item in σ due to the knapsack not having room for the item, ATup$(\sigma) \geq \frac{r}{2}$ Opt$(\sigma) - 1 \geq \frac{1}{2r}$ Opt$(\sigma) - 1$ for $r \geq 1$.

Now, suppose that ATup does not reject any item due to it not fitting in the knapsack. If it 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 ATup 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 Opt$(\sigma) \leq \frac{1}{r\hat{a}} \leq \sqrt{\frac{2(i_t+1)}{\hat{a}}}$. Thus,

$$\frac{\text{ATup}(\sigma) + 1}{\text{Opt}(\sigma)} \geq \frac{i_t + 1}{\sqrt{\frac{2(i_t+1)}{\hat{a}}}} = \sqrt{\frac{\hat{a}(i_t+1)}{2}} \geq \sqrt{\frac{\hat{a}\frac{1}{2r^2\hat{a}}}{2}} = \frac{1}{2r}\,.$$

Therefore, ATup$(\sigma) \geq \frac{1}{2r}$ 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 ATUP 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

$$\frac{\text{ATUP}(\sigma)+1}{\text{OPT}(\sigma)} \geq \frac{(i_t+1)+s}{\ell+s}\,.$$

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

Subcase 2b: $i_t+1 < \frac{1}{2\hat{a}}$. By Ineq. 1 of Lemma 9, and since $\frac{\text{ATUP}(\sigma)+1}{\text{OPT}(\sigma)} \leq 1$,

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

In the full paper, we show that this is at least $\frac{1}{2r}$. ◀

For the case where the actual average size in OPT's packing is no larger than the predicted average size, we get the following result.

▶ **Theorem 11.** *For all request sequences σ, such that $r < 1$,*

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

4.2.2 Negative Result

In Section 3, we showed that, even with accurate advice, no deterministic algorithm can be better than $\frac{e-1}{e}$-competitive. In this section, we give a trade-off in the competitive ratio attained for different values of r.

▶ **Theorem 12.** *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 $\frac{1}{\sqrt{z\hat{a}}}$, it cannot be better than $\frac{zr}{4}$-competitive, for any $r \leq \frac{2}{z}$.

Moreover, if ALG is better than $\frac{zr}{4}$-competitive for some $r \leq \frac{2}{z}$, it cannot be $\frac{1}{\sqrt{z\hat{a}}}$-competitive for all r between $\frac{2}{z}$ and $\frac{1}{\sqrt{z\hat{a}}}$.

Proof. We consider the adversary that gives the input sequence σ_z defined by Algorithm 5. 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 5.

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

Algorithm 5 Adversarial sequence establishing trade-off on robustness versus consistency. 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 1/(qâ) ∈ ℕ
1: p ← ⌊z/(4â)⌋
2: k ← 0
3: while k ≤ p − 1 do
4:     k++
5:     Give ⌊√(zk/â)⌋ items of size √(â/(zk))
6:     if ALG has accepted fewer than k − b items then terminate
7: Give 1/(qâ) items of size qâ
```

$\sqrt{\frac{1}{zp\hat{a}}} \geq \sqrt{\frac{1}{z\hat{a}} \cdot \frac{4\hat{a}}{z}} = \frac{2}{z}$ and $\frac{1}{\sqrt{z\hat{a}}}$. Thus,

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

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 $\frac{1}{\sqrt{z\hat{a}}}$, $\textsc{Alg}(\sigma) \geq \frac{1}{zr}\,\textsc{Opt}(\sigma) - b$, 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 $\textsc{Opt}(\sigma_z) = \frac{1}{q\hat{a}} = \frac{1}{r\hat{a}}$. Moreover, for $\textsc{Alg}$, the ith accepted item must have size at least $\sqrt{\frac{\hat{a}}{z(i+b)}}$, for $1 \leq i \leq p-b$. Thus, these first $p-b$ items fill the knapsack to at least

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

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

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

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. This proves the first part of the theorem.

The second part of the theorem is the contrapositive of the first part. ◀

Setting $z = 2$ in Theorem 12 demonstrates a Pareto-like trade-off between consistency and robustness for ATup:

▶ **Corollary 13.** *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 any positive $r \leq 1$.

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

References

1 Sara Ahmadian, Hossein Esfandiari, Vahab Mirrokni, and Binghui Peng. Robust load balancing with machine learned advice. In *2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 20–34. SIAM, 2022.

2 Spyros Angelopoulos. Online search with a hint. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 185 of *LIPIcs*, pages 51:1–51:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

3 Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc P. Renault. Online computation with untrusted advice. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151 of *LIPIcs*, pages 52:1–52:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

4 Spyros Angelopoulos and Shahin Kamali. Contract scheduling with predictions. In *35th AAAI Conference on Artificial Intelligence (AAAI), 33rd Conference on Innovative Applications of Artificial Intelligence (IAAI), 11th Symposium on Educational Advances in Artificial Intelligence (EAAI)*, pages 11726–11733. AAAI Press, 2021.

5 Spyros Angelopoulos, Shahin Kamali, and Kimia Shadkami. Online bin packing with predictions. *ArXiv*, 2021. `arXiv:2102.03311`.

6 Spyros Angelopoulos, Shahin Kamali, and Dehou Zhang. Online search with best-price and query-based predictions. *ArXiv*, 2021. `arXiv:2112.01592`.

7 Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 345–355. PMLR, 2020.

8 Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In *33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 7933–7944. Curran Associates, Inc., 2020.

9 Etienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. Learning augmented energy minimization via speed scaling. In *33rd Annual conference on Neural Information Processing Systems (NeurIPS)*, pages 15350–15359. Curran Associates, Inc., 2020.

10 Etienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In *33rd Annual conference on Neural Information Processing Systems (NeurIPS)*, pages 20083–20094. Curran Associates, Inc., 2020.

11 Siddhartha Banerjee, Vasilis Gkatzelis, Artur Gorokh, and Billy Jin. Online nash social welfare maximization with predictions. In *2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–19. SIAM, 2022.

12 Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. Learning-augmented weighted paging. In *2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67–89. SIAM, 2022.

13 Aditya Bhaskara, Ashok Cutkosky, Ravi Kumar, and Manish Purohit. Online learning with imperfect hints. In *37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 822–831. PMLR, 2020.

14 Hans-Joachim Böckenhauer, Dennis Komm, Richard Královič, and Peter Rossmanith. The online knapsack problem: Advice and randomization. *Theoretical Computer Science*, 527:61–72, 2014.

15 Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online Algorithms with Advice: A Survey. *ACM Computing Surveys*, 50(2):1–34, 2017. Article No. 19.

16 Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Online unit profit knapsack with untrusted predictions. *ArXiv*, 2022. `arXiv:2203.00285`.

17 Joan Boyar, Lene M. Favrholdt, Kim S. Larsen, and Morten N. Nielsen. The competitive ratio for on-line dual bin packing with restricted input sequences. *Nordic Journal of Computing*, 8:463–472, 2001.

18 Marek Cygan, Łukasz Jeż, and Jirí Sgall. Online knapsack revisited. *Theory of Computing Systems*, 58, 2016.

19 Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In *36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2319–2327. PMLR, 2019.

20 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. In *33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 285–294. ACM, 2021.

21 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Online knapsack with frequency predictions. In *Pre-Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

22 Piotr Indyk, Frederik Mallmann-Trenn, Slobodan Mitrović, and Ronitt Rubinfeld. Online page migration with ML advice. *ArXiv*, 2020. `arXiv:2006.05028`.

23 Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. Online algorithms for weighted paging with predictions. In *47th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 168 of *LIPIcs*, pages 69:1–69:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

24 Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

25 Rohan Kodialam. Optimal algorithms for ski rental with soft machine-learned predictions. *ArXiv*, 2019. `arXiv:1903.00092`.

26 Arvind Kumar and Bashir Alam. Task scheduling in real time systems with energy harvesting and energy minimization. *Journal of Computational Science*, 14(8):1126–1133, 2018.

27 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1859–1877. SIAM, 2020.

28 Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. Learnable and Instance-Robust Predictions for Online Matching, Flows and Load Balancing. In *29th Annual European Symposium on Algorithms (ESA)*, volume 204 of *LIPIcs*, pages 59:1–59:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

29 Russell Lee, Mohammad H. Hajiesmaili, and Jian Li. Learning-assisted competitive algorithms for peak-aware energy scheduling. *ArXiv*, 2020. `arXiv:1911.07972`.

30 Russell Lee, Jessica Maghakian, Mohammad H. Hajiesmaili, Jian Li, Ramesh K. Sitaraman, and Zhenhua Liu. Online peak-aware energy scheduling with untrusted advice. In *12th ACM International Conference on Future Energy Systems (e-Energy)*, pages 107–123. ACM, 2021.

31 Shi Li and Jiayi Xian. Online unrelated machine load balancing with predictions revisited. In *38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 6523–6532. PMLR, 2021.

32 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *35th International Conference on Machine Learning (ICML)*, volume 80, pages 3302–3311. PMLR, 2018.

33 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM*, 68(4):24:1–24:25, 2021.

34 Alberto Marchetti-Spaccamela and Carlo Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995.

35 Andres Muñoz Medina and Sergei Vassilvitskii. Revenue optimization with approximate bid predictions. In *30th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1858–1866. Curran Associates, Inc., 2017.

36 Michael Mitzenmacher. Scheduling with Predictions and the Price of Misprediction. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151 of *LIPIcs*, pages 14:1–14:18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020.

37 Michael Mitzenmacher. Queues with small advice. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA)*, pages 1–12, 2021.

38 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *ArXiv*, 2020. `arXiv:2006.09123`.

39 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *31st Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 9661–9670. Curran Associates, Inc., 2018.

40 Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1834–1845. SIAM, 2020.

41 Daan Rutten and Debankur Mukherjee. A new approach to capacity scaling augmented with unreliable machine learning predictions. *ArXiv*, 2021. `arXiv:2101.12160`.

42 Shufan Wang and Jian Li. Online algorithms for multi-shop ski rental with machine learned predictions. In *19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 2035–2037. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

43 Alexander Wei. Better and simpler learning-augmented online caching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 176 of *LIPIcs*, pages 60:1–60:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

44 Alexander Wei and Fred Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. *ArXiv*, 2020. `arXiv:2010.11443`.

45 Ali Zeynali, Bo Sun Mohammad Hajiesmaili, and Adam Wierman. Data-driven competitive algorithms for online knapsack and set cover. In *35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.