Online Knapsack with Frequency Predictions

Sungjin Im
Electrical Engineering and Computer Science
University of California, Merced
sim3@ucmerced.edu

Ravi Kumar
Google Research
Mountain View, CA
ravi.k53@gmail.com

Mahshid Montazer Qaem
Electrical Engineering and Computer Science
University of California, Merced
mmontazerqaem@ucmerced.edu

Manish Purohit
Google Research
Mountain View, CA
mpurohit@google.com

Abstract

There has been recent interest in using machine-learned predictions to improve the worst-case guarantees of online algorithms. In this paper we continue this line of work by studying the online knapsack problem, but with very weak predictions: in the form of knowing an upper and lower bound for the number of items of each value. We systematically derive online algorithms that attain the best possible competitive ratio for any fixed prediction; we also extend the results to more general settings such as generalized one-way trading and two-stage online knapsack. Our work shows that even seemingly weak predictions can be utilized effectively to provably improve the performance of online algorithms.

1 Introduction

An algorithm designer's ultimate goal is to develop algorithms that are reliable and work well in practice. High reliability has been the major focus of discrete algorithmic research that primarily seeks to give strong worst-case guarantees for *all* inputs. Unfortunately, such algorithms treat all input instances on an equal footing—even pathological ones that rarely occur in practice—and do not necessarily exploit the latent structure that can be present in realistic instances. In contrast, machine learning has demonstrated amazing successes in many real-world applications, yet occasionally exhibits unacceptable failures on specific (worst-case) instances.

Learning-augmented algorithm design has recently emerged as a model to achieve both reliability and worst-case guarantees, particularly in the online setting. In this model, an online algorithm is given certain predictions on the input it will face in the future, and the algorithm then makes irrevocable decisions as the actual input is revealed one by one. Ideally, the algorithm should perform close to the optimum when the predictions are good, and yet possess worst-case guarantees even when the predictions are malicious. Recent work has shown that this is achievable for online problems such as caching [17, 26, 29], ski-rental [2, 14, 22], scheduling [6, 15, 22, 28], load balancing [23], secretary problem [5], metrical task systems [4], set cover [7], flow/matching [24], bin packing [3], etc.

Most prior work on learning-augmented online algorithms assumes that we are given as prediction a specific value of key parameters of the online instance, e.g., the number of days to ski in the ski-rental problem or the next request time of a page in the caching problem. In this paper, we take a relaxed approach and assume that the prediction only gives a *range* of values for key parameters. This is a natural way to model weak predictions where the predictor has high uncertainty and can only provide a reliable range in which the value must lie. Given such weak predictions, we seek to obtain beyond worst-case guarantees for all inputs *respecting* the prediction, by possibly exploiting the structure

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

provided by these predictions. In fact, we seek algorithms that give worst-case guarantees for all inputs while promising sharper bounds for instances that indeed respect the predictions. In this paper, we design such learning-augmented algorithms with weak predictions for the online knapsack problem, which is a basic problem and also of practical importance [31, 30].

Knapsack Problem and the Prediction Model. In the classic *online knapsack problem*, we have a knapsack of unit capacity[i] and n items that arrive online. Each item i has a *profit* $p_i \geq 0$ and a *size* $0 \leq s_i \leq 1$. For convenience, we define the *value* of an item, $v_i := p_i/s_i$, to be the profit of the item per unit size. We assume that the value of each item lies in a range known a priori, i.e., we have $v_{\min} \leq v_i \leq v_{\max}$ for some known constants $v_{\min}$ and $v_{\max}$.[ii] When an item arrives, an online algorithm must make an irrevocable choice of whether to accept or reject the item. The goal is to select a subset of items of the highest total profit whose total size is at most the knapsack capacity.

Recall that an online algorithm is said to have a *competitive ratio* c, or equivalently, be *c-competitive* if the algorithm's objective (profit) is at least c times the total profit obtained by the optimal solution for all inputs. Zhou, Chakrabarty, and Lukose [31] gave a $1/(1 + \ln \frac{v_{\max}}{v_{\min}})$-competitive algorithm (that we call *ZCL*) under the assumption that all items have infinitesimal sizes (or can be accepted fractionally) and further showed that no algorithm can obtain a better competitive ratio.

Online knapsack is a fundamental problem in online algorithm design and finds applications in many real-world settings, e.g., in the context of online auctions [9, 31] and charging electric vehicles [30]. For these practical applications however, the worst-case guarantees offered by standard competitive analysis are too pessimistic to be useful. Indeed, practical instances are not typically worst-case and further one often possesses some foreknowledge of the kind of items that are expected to arrive in the future. In an online ad-auction, for instance, the advertiser can reliably predict a range such that the number of queries in a day for a particular keyword lies within the range. In this work, we explore formal models to utilize such predictions to obtain provably good algorithms for online knapsack beyond what is possible via classical competitive analysis. In particular, we introduce and study the online knapsack problem with frequency predictions (KNAPSACK-FP), described next.

Let Vdenote the set of all possible values that items may take, and $v_{\min} = \min(V)$ and $v_{\max} = \max(V)$ denote minimum and maximum values respectively. We may assume that the items have discrete values by rounding them to the closest power of $(1 + \epsilon)$ for some $\epsilon > 0$, yielding $|V| = O(\log(v_{\max}/v_{\min}))$ and making the competitive ratio only marginally worse. For any value $v \in V$, let $s_v = \sum_{i|v_i=v} s_i$ denote the total size of items with value v. In the *online knapsack with frequency predictions* problem, for each value $v \in V$, the *frequency predictions* $P = \{\ell_v; u_v\}_{v \in V}$ provided to the algorithm are a lower bound ℓ_v and an upper bound u_v such that $\ell_v \leq s_v \leq u_v$. We say that the algorithm has a competitive ratio of α for KNAPSACK-FP with prediction P if the algorithm's profit is at least α times the optimum for all inputs respecting the given frequency prediction P. Our goal is to design an online algorithm with the best competitive ratio for this prediction model.

1.1 Our Results and Techniques

Our main result is a nearly optimal algorithm for KNAPSACK-FP.

Theorem 1. *For* KNAPSACK-FP*, given any frequency predictions, we can find an algorithm of competitive ratio $\alpha^* - \frac{v_{\max}}{v_{\min}} \cdot (|V| + 1) \cdot \max_i s_i$, where α^* is the best possible competitive ratio for the problem (with predictions).*

Our algorithm, called SENTINEL, pre-computes a *budget* b_v for each value v and accepts items that maintain the invariant that the total size of accepted items of value at most v is at most $B_v := \sum_{v_{\min} \leq x \leq v} b_x$ for all $v \in V$. In some sense, B_v can be viewed as a *sentinel* to guard against accepting too many low-valued items. The key intuition underlying our algorithm is that the knowledge regarding the arrival of future higher-valued items enables it to not over-allocate capacity to earlier lower-valued items. Our algorithm can also be implemented using only $O(\log |V|)$ time per item. We note that the additive loss of $\frac{v_{\max}}{v_{\min}} \cdot (|V| + 1) \cdot \max_i s_i$ in the competitive ratio in Theorem 1 is

[i] This is without loss of generality by scaling all sizes.

[ii] Such an assumption is necessary since otherwise we cannot hope for a bounded competitive ratio in the adversarial model [27, 31].

due to a reduction to a continuous version of the problem (as discussed in Section 2) and this quantity approaches zero as $\max_i s_i \to 0$.

Interestingly our algorithm has a distinguishing feature from the ZCL [31] algorithm for the standard online knapsack. The ZCL algorithm constructs a *threshold* function $\Psi(t) = (e \cdot v_{\max}/v_{\min})^t \cdot (v_{\min}/e)$ a priori and accepts an item online if its value is no smaller than $\Psi(t)$ when t is the knapsack capacity that has been consumed. In contrast, we show that *no* such algorithm that maintains a threshold function based on the capacity used can give the optimum algorithm (Section 3.2). We note that by combining our algorithm and ZCL (say by using a γ fraction of the capacity to run our algorithm and the other $1-\gamma$ fraction to run ZCL), we can simultaneously obtain a worst-case guarantee of $\frac{1-\gamma}{1+\log(\frac{v_{\max}}{v_{\min}})}$ as well as our improved guarantees for instances that respect the prediction. Further, we can easily extend our result to show that our algorithm has a small loss in the competitive ratio when the predictions are slightly incorrect, for example, when we have $\ell_v/(1+\epsilon) \leq s_v \leq u_v(1+\epsilon)$.

Extensions. We extend our model and results to two generalizations of the knapsack problem: the *generalized one-way trading* problem and the *two-stage online knapsack* problem. In generalized one-way trading [13, 30], each item of type t is associated with a certain concave function f_t with $f_t(0) = 0$. If an algorithm accepts s amount of items having type t, then it obtains profit $f_t(s)$. The goal, as always, is to accept a set of items subject to the knapsack capacity to maximize the total profit. This is a well-studied problem, motivated by converting a given amount of money from one currency to another, in the presence of time-varying conversion rates.

The two-stage online knapsack is a generalization that we introduce in this paper. The key difference from the standard setting is that after seeing all the items, the online algorithm is given another chance in a second stage to accept any items that it rejected before. But, an item accepted in the second stage gives only a $\lambda \leq 1$ fraction of its original profit. The two-stage knapsack effectively interpolates between the online and offline problems: it becomes the standard online knapsack problem if $\lambda = 0$, and the offline problem if $\lambda = 1$. The two-stage setting has natural applications. For example, while clients typically prefer to know if their requests will be accepted instantly, some might choose to wait further for a discount, albeit with the possibility of getting no service if the service provider accepts other competing requests arriving later.

For both extensions we generalize our results and obtain the best competitive algorithms given frequency predictions (Sections 4 and 5). This shows that our prediction model and analysis could find more applications, providing beyond-worst-case guarantees even with weak predictions.

Experiments. In Section 6, we conduct experiments on synthetic data sets, and show our algorithm augmented with frequency predictions considerably outperforms the classic worst-case ZCL algorithm.

1.2 Other Related Work

The knapsack problem has been extensively studied in the literature. For a comprehensive survey of the results on the offline knapsack problem, see [18]. The offline version of the standard knapsack problem and most of its variants are well-known to be NP-hard.

The classic online knapsack problem falls into a general class of problems called online packing problems and one can obtain an $\Omega(1/\ln(v_{\max}/v_{\min}))$-competitive algorithm via the online primal-dual technique [10, 11]. For a nice survey of online primal-dual technique and its applications for various packing and covering problems, see [12]. Sun et al. [30] generalize the threshold algorithm in [31] by introducing a control parameter that adjusts how fast the threshold grows. They empirically learn the best parameter within a certain range to retain some worst-case guarantees, but do not provide optimality. Kong et al. [21] empirically show that reinforcement learning can find a good threshold for the online knapsack.

Another line of work considers the stochastic online knapsack problem where items' profits and sizes are drawn from a distribution (e.g., see [27, 25, 20] and their follow-up works). In the random arrival model, the items are assumed to be chosen adversarially but arrive in a uniformly random order. In this model, one can obtain constant competitive algorithms [1, 19] and further if all items have an infinitesimal size, one can obtain nearly-optimal online algorithms [19]. There are many other

variants studied and we only mention a few here, e.g., removable online knapsack problems [16] and online knapsack with reservation costs [8].

2 Preliminaries

To ease the presentation, we will consider the following relaxed model, which differs from KNAPSACK-FP in two aspects: first, we allow an item to be accepted fractionally and second, we allow an item to have any value in $[v_{\min}, v_{\max}]$, not from a discrete set. The second relaxation is purely for ease of presentation and it preserves the competitive ratio. The first relaxation results in a small additive loss in the competitive ratio since at most one item is accepted fractionally among those of the same value and all items are assumed to be small. We now formalize the relaxed model.

Continuous-Valued Online Knapsack with Frequency Predictions (C-KNAPSACK-FP). In the continuous version, we assume that there is a continuum of items, each having *infinitesimal* size and some value $v \in [v_{\min}, v_{\max}]$. In a particular problem instance, for any value $v \in [v_{\min}, v_{\max}]$, let $s(v)\mathtt{d}v$ be the total size of items having value in the range $[v, v + \mathtt{d}v]$ for some sufficiently small $\mathtt{d}v$. For each value $v \in [v_{\min}, v_{\max}]$, we are given a lower bound[iii] $\ell(v)$ and an upper bound $u(v)$ such that $s(v) \in [\ell(v), u(v)]$. We assume that $s(v), \ell(v), u(v)$ are all continuous in v. Define $S(v) := \int_{x=v_{\min}}^{v} s(x)\mathtt{d}x$, $L(v) := \int_{x=v_{\min}}^{v} \ell(x)\mathtt{d}x$, and $U(v) := \int_{x=v_{\min}}^{v} u(x)\mathtt{d}x$, which are differentiable at all $v \in [v_{\min}, v_{\max}]$ by definition. As in the standard model, we have a knapsack of unit capacity, and the goal is to select a subset of items of total size at most one that maximizes total value. The following relates the competitive ratio of the standard and the continuous-valued versions.

Lemma 2. *Given a prediction $P = \{\ell_v; u_v\}_{v \in V}$ for* KNAPSACK-FP*, we can create a prediction $P' = \{\ell(v); u(v)\}_{v \in [v_{\min}, v_{\max}]}$ for* C-KNAPSACK-FP *in polynomial time such that*

1. *An α-competitive algorithm for* C-KNAPSACK-FP *with prediction P' implies an $(\alpha - \frac{v_{\max}}{v_{\min}} \cdot |V| \cdot \max_i s_i)$-competitive algorithm for* KNAPSACK-FP *with prediction P.*
2. *If no algorithm has a competitive ratio better than α for* C-KNAPSACK-FP *with prediction P', then no algorithm has a competitive ratio better than $\alpha + \max_i s_i$ for* KNAPSACK-FP *with prediction P.*

Thus, if we find an algorithm with the optimum competitive ratio for C-KNAPSACK-FP, then we will have Theorem 1. In the rest of the paper we only consider the continuous version for convenience.

3 The SENTINEL algorithm

Let $\{\ell(v); u(v)\}_{v \in [v_{\min}, v_{\max}]}$ be a fixed set of predictions; this defines a family of knapsack instances, where each instance is a sequence of items, each with a profit and size. Let α denote the desired competitive ratio of our algorithm. We first derive an online algorithm that guarantees a competitive ratio of α assuming one exists and then discuss how one can find the optimal such competitive ratio α^*. For any instance $\mathcal{I}$, let $\text{OPT}(\mathcal{I})$ denote the total value obtained by the optimal solution for instance $\mathcal{I}$. To systematically derive our algorithm, it is instructive to consider the following family of instances. For any value $v \in [v_{\min}, v_{\max}]$, let $\mathcal{M}(v)$ denote an instance containing the maximum possible amount of items of value at most v and the minimum possible amount of items of value larger than v. Formally, $\mathcal{M}(v)$ is an instance such that $s(x) = u(x)$ for all $x \leq v$ and $s(x) = \ell(x)$ for all $x > v$. Further, items in $\mathcal{M}(v)$ arrive in the following order: first $\ell(x)$ items of value $x \in [v_{\min}, v_{\max}]$ arrive in non-decreasing order of value, followed by $u(x) - \ell(x)$ items of value $x \leq v$ arriving in non-decreasing order. Note that we consider $\mathcal{M}(v)$ only to derive our algorithm.

Step 1. Let v^* denote the largest value such that

$$\int_{x=v^*}^{v_{\max}} x \cdot \ell(x)\mathtt{d}x = \alpha \cdot \text{OPT}(\mathcal{M}(v^*)). \tag{1}$$

[iii] We use the functional notation (e.g., $\ell(v)$) for the continuous case and the subscript notation (e.g., ℓ_v) for the discrete case.

Intuitively v^* denotes the largest value such that an algorithm is guaranteed to be α-competitive even if it does not accept any items of value less than v^*. Note that since we assume that the functions $\ell(\cdot)$ and $u(\cdot)$ are continuous, such a value v^* is guaranteed to exist by the intermediate value theorem.

Step 2. Consider an adversary that has constructed an instance of the form $\mathcal{M}(x)$ for some unknown value x. By instead constructing an instance of the form $\mathcal{M}(x + \mathtt{d}x)$, the adversary can gain an additional profit of $\frac{\mathtt{d}}{\mathtt{d}x}\textsc{opt}(\mathcal{M}(x))$. In order to maintain a competitive ratio of α, our algorithm must accept enough additional items of value x so that it receives a profit of at least $\alpha \cdot \frac{\mathtt{d}}{\mathtt{d}x}\textsc{opt}(\mathcal{M}(x))$. Motivated by this discussion, we define a function $\tau : [v^*, v_{\max}] \to [0, \infty)$ to determine the additional amount of items of a particular value that our algorithm accepts.

$$\tau(x) = \frac{\alpha}{x}\frac{\mathtt{d}}{\mathtt{d}x}\textsc{opt}(\mathcal{M}(x)), \qquad \forall x \in [v^*, v_{\max}]. \tag{2}$$

Step 3. For any value $x \in [v_{\min}, v_{\max}]$, we define a budget function as follows.

$$b(x) = \begin{cases} 0, & \forall x < v^* \\ \ell(x) + \tau(x), & \forall x \geq v^* \end{cases}; \text{ and } B(v) = \int_{x=v_{\min}}^{v} b(x)\mathtt{d}x \quad \forall v \in [v_{\min}, v_{\max}]. \tag{3}$$

For any value v, the budget function $B(v)$ defined above prescribes the maximum total amount of items of value at most v that must be accepted. Formally, let $A(v)$ denote the total amount of items of value at most v that has been accepted by the online algorithm so far. Then, when an item of value v arrives, the algorithm accepts the item if and only if $A(x) < B(x)$ for all $x \geq v$.

Step 4. Finally, we discuss how to find the optimal competitive ratio. The functions $\tau(\cdot)$, $b(\cdot)$, and $B(\cdot)$ defined above are all (implicitly) functions of the desired competitive ratio α. To make this dependence explicit, let $B_\alpha(\cdot)$ denote the budget function for some fixed α. It can be easily verified that $B_\alpha(v_{\max})$ is a continuous and monotonically increasing function of α. Assume $U(v_{\max}) > 1$ since otherwise we can accept all items by setting $b(x) = u(x)$ and we obtain a competitive ratio of 1. We set $\alpha^* \in (0, 1)$ to be such that $B_{\alpha^*}(v_{\max}) = 1$. Again, by the intermediate value theorem, such an α^* is guaranteed to exist and further one can compute it up to arbitrary precision by binary search. For clarity, we include a formal description of the algorithm in the Supplementary Material.

3.1 Analysis

We begin by proving that our algorithm is indeed α^*-competitive for any instance of C-KNAPSACK-FP that respects the given frequency predictions. Let $\tilde{v}$ be the largest value such that $A(\tilde{v}) = B(\tilde{v})$; if no such a value exists, set $\tilde{v} = v_{\min}$. Let ALG and OPT denote the total value of items chosen by our online algorithm and the optimal solution respectively. In the following two claims, we first bound ALG and OPT respectively.

Claim 3. $\textsc{alg} \geq \int_{x=v_{\min}}^{\tilde{v}} x \cdot b(x)\,\mathtt{d}x + \int_{x=\tilde{v}}^{v_{\max}} x \cdot s(x)\,\mathtt{d}x$.

Proof. For convenience of notation, let $a(v) = \frac{\mathtt{d}}{\mathtt{d}x}A(x)|_{x=v}$ denote the marginal amount of items of value v accepted by the algorithm. Since we have $A(v) < B(v)$ for all $v > \tilde{v}$, the algorithm accepts all items having value larger than $\tilde{v}$ and thus $a(x) = s(x), \forall x > \tilde{v}$. Also, by definition of $\tilde{v}$, we have $A(\tilde{v}) = B(\tilde{v})$. Thus, we have the following.

$$\textsc{alg} = \int_{x=v_{\min}}^{v_{\max}} x \cdot a(x)\mathtt{d}x = \int_{x=v_{\min}}^{\tilde{v}} x \cdot a(x)\mathtt{d}x + \int_{x=\tilde{v}}^{v_{\max}} x \cdot s(x)\mathtt{d}x$$

integrating the first term by parts yields

$$\textsc{alg} = \tilde{v} \cdot A(\tilde{v}) - v_{\min} \cdot A(v_{\min}) - \int_{v_{\min}}^{\tilde{v}} A(x)\mathtt{d}x + \int_{x=\tilde{v}}^{v_{\max}} x \cdot s(x)\mathtt{d}x$$

since $A(\tilde{v}) = B(\tilde{v})$ and $A(x) \leq B(x),\ \forall x$, by definition of our algorithm

$$\geq \tilde{v} \cdot B(\tilde{v}) - v_{\min} \cdot B(v_{\min}) - \int_{v_{\min}}^{\tilde{v}} B(x)\mathtt{d}x + \int_{x=\tilde{v}}^{v_{\max}} x \cdot s(x)\mathtt{d}x$$

$$= \int_{x=v_{\min}}^{\tilde{v}} x \cdot b(x)\mathtt{d}x + \int_{x=\tilde{v}}^{v_{\max}} x \cdot s(x)\mathtt{d}x. \qquad \square$$

Claim 4. $\text{OPT} \leq \text{OPT}(\mathcal{M}(\tilde{v})) + \int_{x=\tilde{v}}^{v_{\max}} x \cdot (s(x) - \ell(x))\,\mathtt{d}x$.

Proof. Let the optimum solution accept $s(x) - \ell(x)$ items of value x for free without consuming the knapsack capacity for all $x \in [\tilde{v}, v_{\max}]$. The remaining items form an instance $\mathcal{I}'$ that is dominated by $\mathcal{M}(\tilde{v})$, i.e., for any value $x \in [v_{\min}, v_{\max}]$, the amount of items having value x in the instance $\mathcal{I}'$ is at most the amount of items with value x in $\mathcal{M}(\tilde{v})$. However, by definition, $\text{OPT}(\mathcal{M}(\tilde{v}))$ is the maximum profit that one can obtain from $\mathcal{M}(\tilde{v})$ and the claim follows. $\square$

The following is from the definition of SENTINEL and is useful to relate the value of the optimal solution for any instance of the form $\mathcal{M}(v)$ to the budget function $b(\cdot)$ utilized by our algorithm.

Claim 5. *For any value* $v \in [v^*, v_{\max}]$, $\alpha^* \cdot \text{OPT}(\mathcal{M}(v)) = \int_{x=v}^{v_{\max}} x \cdot \ell(x)\,\mathtt{d}x + \int_{x=v_{\min}}^{v} x \cdot b(x)\,\mathtt{d}x$.

Proof. For all $v \geq v^*$, we have the following.

$$\alpha^* \cdot \text{OPT}(\mathcal{M}(v)) = \alpha^* \cdot \left(\text{OPT}(\mathcal{M}(v^*)) + \int_{x=v^*}^{v} \frac{\mathtt{d}}{\mathtt{d}x} \text{OPT}(\mathcal{M}(x))\mathtt{d}x \right)$$

substituting (1) and (2),

$$= \int_{x=v^*}^{v_{\max}} x \cdot \ell(x)\mathtt{d}x + \int_{x=v^*}^{v} x \cdot \tau(x)\mathtt{d}x.$$

The claim follows by the definition of $b(x) = \ell(x) + \tau(x)$ for all $x \geq v^*$, and $b(x) = 0,\ \forall x < v^*$. $\square$

Finally, we are ready to show that the algorithm is indeed α^*-competitive. Claims 3 and 5 together yield that:

$$\text{ALG} \geq \alpha^* \text{OPT}(\mathcal{M}(\tilde{v})) + \int_{x=\tilde{v}}^{v_{\max}} x \cdot (s(x) - \ell(x))\mathtt{d}x$$

$$\geq \alpha^* \cdot \left(\text{OPT}(\mathcal{M}(\tilde{v})) + \int_{x=\tilde{v}}^{v_{\max}} x \cdot (s(x) - \ell(x))\mathtt{d}x \right) \geq \alpha^* \cdot \text{OPT},$$

where the last inequality follows from Claim 4. Thus, we have the following theorem. As discussed, the value of α^* depends on the predictions.

Theorem 6. *For any input instance that respects the frequency predictions* $\{\ell(v), u(v)\}_{v \in [v_{\min}, v_{\max}]}$, *the* SENTINEL *algorithm is* α^**-competitive for* C-KNAPSACK-FP.

3.1.1 Optimality of Competitive Ratio

In this section we show that the competitive ratio obtained by SENTINEL is optimal. In other words, no algorithm, even randomized, can obtain a competitive ratio of better than α^* for C-KNAPSACK-FP.

We will consider the family of instances $\{\mathcal{M}(v)\}_{v \in [v_{\min}, v_{\max}]}$. Recall that in the instance $\mathcal{M}(v)$, first $\ell(x)$ amount of items of all values $x \in [v_{\min}, v_{\max}]$ arrive in non-decreasing order of value, followed by $u(x) - \ell(x)$ amount of items of value $x \in [v_{\min}, v]$, which also arrive in non-decreasing order of value. Consequently, until an algorithm sees items of value higher than v in the second stage, it cannot distinguish between instances in $\{\mathcal{M}(x)\}_{x \in [v, v_{\max}]}$. In other words, the instance $\mathcal{M}(v)$ is a *prefix* of each of the instances in $\{\mathcal{M}(x)\}_{x \in [v, v_{\max}]}$. Thus for any value v, the adversary can either strategically stop the instance at $\mathcal{M}(v)$ or continue to release higher-valued items.

Lemma 7. *For any given set* $\{\ell(v), u(v)\}_{v \in [v_{\min}, v_{\max}]}$ *of frequency predictions, let* α^* *denote the competitive ratio of the* SENTINEL *algorithm. Then no deterministic algorithm can obtain a competitive ratio larger than* α^* *on all instances in the family* $\{\mathcal{M}(v)\}_{v \in [v_{\min}, v_{\max}]}$.

To see why randomization does not help, consider the deterministic algorithm that accepts each item by the expected amount that the optimum randomized algorithm accepts. This is well-defined as we are allowed to choose items fractionally. It is easy to see that the algorithm never exceeds the knapsack capacity and obtains as much value as the randomized algorithm. Thus, we have:

Theorem 8. *For any given set $\{\ell(v), u(v)\}_{v\in[v_{\min},v_{\max}]}$ of frequency predictions, there exists no randomized algorithm that attains a competitive ratio better than that of the* SENTINEL *algorithm.*

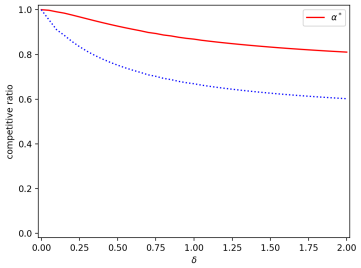

Figure 1: Illustration of the optimal competitive ratio α^* obtained by SENTINEL (in red) for predictions $\ell_v = 1/100$ and $u_v = (1+\delta)\cdot\ell_v$ for integers $v \in [v_{\min} = 1, v_{\max} = 100]$. The competitive ratio is shown for $\delta \in [0, 2]$. For comparison, the competitive ratio resulting from accepting ℓ_v items of value v is shown in blue, i.e., $\text{OPT}(\mathcal{M}(0))/\text{OPT}(\mathcal{M}(v_{\max}))$.

Thus SENTINEL yields the *optimal* competitive ratio for any given set of predictions. It can be easily verified that the classical adversarial online knapsack problem is equivalent to the special case with predictions $\ell(v) = 0$ and $u(v) = \infty$ for all $v \in [v_{\min}, v_{\max}]$. For this special case, we recover the competitive ratio $\frac{1}{1+\ln(\frac{v_{\max}}{v_{\min}})}$ of ZCL. For more informative predictions, we obtain much tighter competitive ratios. Indeed, if $\ell(v) = u(v), \forall v \in [v_{\min}, v_{\max}]$, then $\alpha^* = 1$ and SENTINEL obtains the optimum solution. Figure 1 illustrates the optimum competitive ratio for a family of predictions.

3.2 Need for Sentinel

Prior algorithms for (variants of) the online knapsack problem [31, 30] have all been *threshold* algorithms, i.e., the algorithm chooses a threshold Ψ that is a function of the amount of residual capacity left in the knapsack and the algorithm accepts an item if and only if its value is higher than the threshold. On the other hand, the SENTINEL algorithm does not fit into this framework since the threshold for choosing an item depends on the values of items already chosen by the algorithm. In this section we show that there are predictions $\{l_v; u_v\}_{v\in V}$ such that no pure *threshold-based* algorithm can achieve the best competitive ratio. In fact, these predictions use only two values.

Let $\mathcal{I}$ denote a family of instances that respect the following predictions. There are two distinct values, $V = \{1, 2\}$. The predicted bounds are given by $l_1 = l_2 = 0$ and $u_1 = u_2 = 2/3$. As discussed in Section 3.1.1, since randomization has no benefits, we only consider deterministic algorithms.

Claim 9. *No deterministic online algorithm that maintains a non-decreasing threshold function $\Psi(z)$, where z is the knapsack capacity used, and accepts an item if and only if its value is higher than $\Psi(z)$ has a competitive ratio better than $4/5$ for the family of instances $\mathcal{I}$.*

On the other hand, with these predictions there is a simple algorithm that obtains a competitive ratio of $6/7$. Let $\mathcal{A}_{\text{sentinel}}$ denote the following algorithm: accept all items of value 2 and accept at most $4/7$ items of value 1. The proof of the following claim follows from a simple case analysis.

Claim 10. *For the instances in $\mathcal{I}$, the algorithm $\mathcal{A}_{\text{sentinel}}$ yields a competitive ratio of $6/7$.*

4 Generalized One-Way Trading

In the classical one-way trading problem [13], a trader needs to convert a unit of money from one currency to another. Faced with a sequence of (unknown) exchange rates that are presented online, the goal is to maximize the amount of money obtained in the target currency. In this section we consider the following generalized one-way trading problem, also considered by [30]: There are T different types of items and each type t is associated with a *concave* value function f_t. For convenience, we consider the setting where each item i has the same infinitesimal size λ. Items arrive online and the total value earned by the algorithm is $\sum_t f_t(x_t)$, where x_t denotes the total size of items of type t

accepted by the algorithm. The goal is to design an algorithm that maximizes the total value such that the total size of accepted items is at most one. As in the online knapsack problem, we assume that the algorithm has access to a lower bound ℓ_t and an upper bound u_t on the total size of items of type t.

We now show that we can reduce this problem to the online knapsack problem with predictions. For each type t, we create ℓ_t/λ *mandatory* items with values $f_t(\lambda), f_t(2\lambda)-f_t(\lambda), \ldots, f_t(\ell_t)-f_t(\ell_t-\lambda)$ respectively. Additionally, we create u_t/λ *optional* items with values $f_t(\ell_t+\lambda)-f_t(\ell_t), \ldots, f_t(u_t)-f_t(u_t-\lambda)$ respectively. Note that many created items belonging to different types may have the same value. For each value v, let $\ell(v)$ be the total size of *mandatory* items of value v and similarly let $u(v)$ be the total size of all *mandatory* and *optional* items of value v. Let $\mathcal{A}$ be an online algorithm for the knapsack problem using predictions $\{\ell(v), u(v)\}$. For the generalized one-way trading problem, as items arrive online, we construct an instance of the knapsack problem online such that the kth item of type t has value $f_t(k\lambda) - f_t((k-1)\lambda)$. Finally, we accept an item (in the generalized one-way trading problem) iff the corresponding item is accepted by algorithm $\mathcal{A}$ in the knapsack instance.

It is easy to verify that the optimum solution attains the same value, say OPT, in both the problem instances. Let α^* denote the competitive ratio attained by $\mathcal{A}$, and hence the total value obtained by $\mathcal{A}$ on the knapsack instance is at least $\alpha^*\text{OPT}$. On the other hand, suppose $\mathcal{A}$ accepts k items of type t, then due to concavity of f_t, it attains a value of at most $\sum_{j=1}^{k} f_t(j\lambda) - f_t((j-1)\lambda) = f_t(k\lambda)$. Thus, the total value obtained by the online algorithm for the generalized one-way trading problem is at least that obtained by $\mathcal{A}$ and hence it is also α^*-competitive.

5 Two-Stage Knapsack

Consider the following two-stage version of the typical fractional online knapsack problem. An instance here is parameterized by a discount factor $0 \leq \lambda \leq 1$. Once again, for convenience, we work in the continuous model, so there is a continuum of items each having infinitesimal size[iv]. We assume that all items have a value between $[v_{\min}, v_{\max}]$. When an item of value v arrives, an algorithm can either pick the item and receive a value of v or it may choose to reject the item. After all items have arrived, the second stage starts and the algorithm can go back to previously rejected items and choose some of them, subject to the knapsack capacity. However, an item originally of value v is only worth λv in the second stage. Note that an item cannot be rejected once it gets accepted in either stage.

We note that this two-stage knapsack formulation naturally interpolates between the online and offline versions of the standard knapsack problem; when $\lambda = 0$, an algorithm cannot obtain any value from the second stage and hence reduces to the regular online knapsack problem, whereas with $\lambda = 1$, an algorithm can commit to only accepting items in the second stage after all items have been revealed and thus the problem reduces to the offline version.

To the best of our knowledge this problem has not been considered in the literature. Thus we first show how to obtain the optimal competitive ratio for the problem in the absence of predictions. We then continue to show how to incorporate predictions on item frequencies for each value. We present the algorithm without predictions and defer the one with predictions to the Supplementary Material.

5.1 Algorithm for Two-Stage Knapsack without Frequency Predictions

As before let α^* be the optimal competitive ratio for the problem, which will be decided later. In the absence of predictions, the optimal competitive ratio can be attained by a *threshold* algorithm. In the first stage of the problem, the algorithm will maintain a threshold function depending on the knapsack capacity used and accept an item if and only if its value is higher than the current threshold. Finally, in the second stage, it is easy to observe that any reasonable algorithm accepts the highest value items that can still be accommodated in the knapsack. We now show how to systematically derive such an algorithm.

For any value v, let $z(v)$ be the fraction of the knapsack capacity for which the algorithm maintains the threshold at value v. To derive $z(v)$ we will consider the following family of adversarial instances: (i) items arrive in increasing order of value; (ii) there are enough items of each value to fill the entire knapsack; and (iii) the adversary can stop releasing items at any point in time.

[iv] As in Lemma 2, when items have sufficiently small sizes, this assumption only leads to a small additive loss in the competitive ratio.

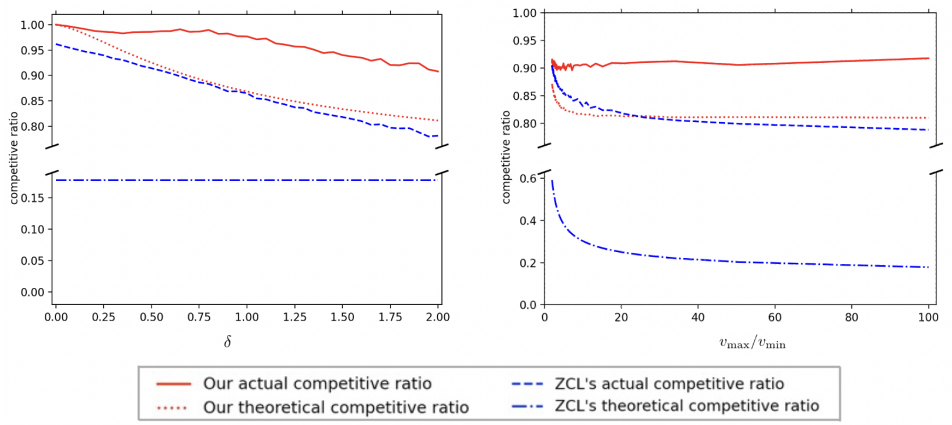

Figure 2: Illustration of the empirical and theoretical competitive ratios of our algorithm SENTINEL and ZCL. In the left, $v_{\min} = 1$, $v_{\max} = 100$, and $\delta \in [0, 2]$; and in the right, $v_{min} \in [1, 100]$, $v_{\max} - v_{\min} = 99$ and $\delta = 2$. In both $u_v = \lceil (1+\delta) \rceil \ell_v$. All items have size 1/10000.

We construct $z(v)$ that is continuous at all $v \in (v_{\min}, v_{\max}]$ as follows. First we set, $z(v_{\min}) = \frac{\alpha^* - \lambda}{1-\lambda}$.This is because if the adversary only gives items of value $v_{\min}$ and they are enough to fill up the knapsack, then by accepting $z(v_{\min})$ of them in the first stage and more to the full capacity in the second stage we obtain profit $(z(v_{\min}) + \lambda(1 - z(v_{\min})))v_{\min} = \alpha^* v_{\min}$.

Now consider any value $v > v_{\min}$. If the adversary stops the instance at value v, the profit obtained by the algorithm is:

$$\text{ALG}(v) := v_{\min} z(v_{\min}) + \int_{x \in (v_{\min}, v]} x \cdot z(x) \mathtt{d}x + \lambda v \Big(1 - z(v_{\min}) - \int_{x \in (v_{\min}, v]} z(x) \mathtt{d}x\Big).$$

Here, the first two terms denote the value earned by the algorithm in the first stage while the last term denotes the profit gained in the second stage. In particular, $(1 - z(v_{\min}) - \int_{x=v_{\min}}^{v} z(x)\mathtt{d}x)$ is the amount of capacity left in the knapsack after the first stage and the algorithm will fill it with the best items—of value v—and receive value of λv per each unit.

To maintain a competitive ratio of α^*, we need to maintain for all $v \in (v_{\min}, v_{\max}]$, $\mathcal{M}(v) = \alpha^* v$. By solving this, together with $z(v_{\min}) + \int_{x \in (v_{\min}, v_{\max}]} z(x)\mathtt{d}x = 1$, we can find α^* and $z(v)$. Then, set the threshold function $\Psi(t)$ as follows:

$$\Psi(t) = \begin{cases} v_{\min} & \text{if } 0 \le t \le \frac{\alpha^* - \lambda}{1-\lambda} \\ Z^{-1}(t) & \text{otherwise} \end{cases},$$

where $Z(v) := \int_{x=v_{\min}}^{v} z(x)\mathtt{d}x$; this integral includes $z(v_{\min})$. And if we see an item of value v in the first phase when the knapsack is t-full, we accept it if $v \ge \Psi(t)$. In other words, we keep the threshold at $v_{\min}$ until the knapsack gets $\frac{\alpha^* - \lambda}{1-\lambda}$-full; afterwards, keep the threshold at v for $z(v)$ units. In the second phase, we accept the highest valued remaining items to the full capacity. Further details are in the Supplementary Material.

6 Experiments

We design experiments to show that our algorithm, SENTINEL, achieves a considerably larger profit than ZCL [31] even when we are given loose frequency bounds as predictions. Specifically, we consider the following setup. The knapsack capacity is set to 1. Each item is assumed to have a size 10^{-4} and an integer value in the range of $[v_{\min} = 1, v_{\max} = 100]$. For each value v, its frequency lower bound ℓ_v is sampled from $[50, 150]$ independently and uniformly at random; therefore, in expectation, at least 10^4 items arrive in an instance. We use a control parameter δ to derive the upper bounds u_v from ℓ_v and set u_v to $\lceil (1+\delta)\ell_v \rceil$. For each instance, s_v is sampled from $[\ell_v, u_v]$

independently and uniformly at random and items arrive in random order. The results are the geometric average over 10 instances.

In the first experiment we observe how the competitive ratios of SENTINEL and ZCL change as we vary δ from 0 to 2. Intuitively, smaller δ value means better predictions. We confirm that SENTINEL outperforms ZCL for all values of δ in terms of empirical as well as theoretical competitive ratios. The actual competitive ratios degrade as δ grows, but SENTINEL continues to dominate ZCL.

In the second experiment, we consider different values of $v_{\min}$ in the range of $\{1, \ldots, 100\}$, fixing the difference $v_{\max} - v_{\min} = 99$ and $\delta = 2$. The goal of this experiment is to see how the competitive ratios change as the ratio $v_{\max}/v_{\min}$ changes. Note that the ratio decreases from 100 to nearly 2. Recall that ZCL has competitive ratio $1/(1 + \ln(v_{\max}/v_{\min}))$. Again, SENTINEL consistently outperforms ZCL, although the difference becomes smaller as $v_{\max}/v_{\min}$ decreases.

7 Conclusions

In this paper we studied the online knapsack problem and its extensions when given the range of the number of items of each value as predictions. We showed that even such weak predictions can be exploited to give provably better competitive ratios. It would be interesting to revisit other problems with such weak predictions. Another fruitful direction is to explore other types of weak predictions.

Acknowledgements

Im and Montazer Qaem are supported in part by NSF grants CCF-1617653 and CCF-1844939.

References

[1] S. Albers, A. Khan, and L. Ladewig. Improved online algorithms for knapsack and GAP in the random order model. *Algorithmica*, pages 1–36, 2021.

[2] K. Anand, R. Ge, and D. Panigrahi. Customizing ML predictions for online algorithms. In *ICML*, pages 303–313, 2020.

[3] S. Angelopoulos, S. Kamali, and K. Shadkami. Online bin packing with predictions, 2021. arXiv:2102.03311.

[4] A. Antoniadis, C. Coester, M. Elias, A. Polak, and B. Simon. Online metric algorithms with untrusted predictions. In *ICML*, pages 345–355, 2020.

[5] A. Antoniadis, T. Gouleakis, P. Kleer, and P. Kolev. Secretary and online matching problems with machine learned advice. In *NeurIPS*, 2020.

[6] É. Bamas, A. Maggiori, L. Rohwedder, and O. Svensson. Learning augmented energy minimization via speed scaling. In *NeurIPS*, 2020.

[7] É. Bamas, A. Maggiori, and O. Svensson. The primal-dual method for learning augmented algorithms. In *NeurIPS*, 2020.

[8] H. Böckenhauer, E. Burjons, J. Hromkovic, H. Lotze, and P. Rossmanith. Online simple knapsack with reservation costs. In *STACS*, 2021.

[9] C. Borgs, J. Chayes, N. Immorlica, K. Jain, O. Etesami, and M. Mahdian. Dynamics of bid optimization in online advertisement auctions. In *WWW*, pages 531–540, 2007.

[10] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *ESA*, pages 689–701. Springer, 2005.

[11] N. Buchbinder and J. Naor. Improved bounds for online routing and packing via a primal-dual approach. In *FOCS*, pages 293–304, 2006.

[12] N. Buchbinder and J. Naor. The design of competitive online algorithms via a primal-dual approach. *Found. Trends Theor. Comput. Sci.*, 3(2-3):93–263, 2009.

[13] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 30(1):101–139, 2001.

[14] S. Gollapudi and D. Panigrahi. Online algorithms for rent-or-buy with expert advice. In *ICML*, pages 2319–2327, 2019.

[15] S. Im, R. Kumar, M. Montazer Qaem, and M. Purohit. Non-clairvoyant scheduling with predictions. In *SPAA*, pages 285–294, 2021.

[16] K. Iwama and S. Taketomi. Removable online knapsack problems. In *ICALP*, pages 293–305, 2002.

[17] Z. Jiang, D. Panigrahi, and K. Sun. Online algorithms for weighted paging with predictions. *ICALP*, pages 69:1–69:18, 2020.

[18] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, Germany, 2004.

[19] T. Kesselheim, K. Radke, A. Tonnis, and B. Vocking. Primal beats dual on online packing LPs in the random-order model. *SIAM Journal on Computing*, 47(5):1939–1964, 2018.

[20] A. J. Kleywegt and J. D. Papastavrou. The dynamic and stochastic knapsack problem. *Operations Research*, 46(1):17–35, 1998.

[21] W. Kong, C. Liaw, A. Mehta, and D. Sivakumar. A new dog learns old tricks: RL finds classic optimization algorithms. In *ICLR*, 2018.

[22] R. Kumar, M. Purohit, and Z. Svitkina. Improving online algorithms using ML predictions. In *NeurIPS*, pages 9661–9670, 2018.

[23] S. Lattanzi, T. Lavastida, B. Moseley, and S. Vassilvitskii. Online scheduling via learned weights. In *SODA*, pages 1859–1877, 2020.

[24] T. Lavastida, B. Moseley, R. Ravi, and C. Xu. Learnable and instance-robust predictions for online matching, flows and load balancing. In *ESA*, pages 59:1–59:17, 2021.

[25] G. S. Lueker. Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms*, 29(2):277–305, 1998.

[26] T. Lykouris and S. Vassilvtiskii. Competitive caching with machine learned advice. In *ICML*, pages 3296–3305, 2018.

[27] A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68(1):73–104, 1995.

[28] M. Mitzenmacher. Scheduling with predictions and the price of misprediction. In *ITCS*, pages 14:1–14:18, 2020.

[29] D. Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *SODA*, pages 1834–1845, 2020.

[30] B. Sun, A. Zeynali, T. Li, M. H. Hajiesmaili, A. Wierman, and D. H. K. Tsang. Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(3):51:1–51:32, 2020.

[31] Y. Zhou, D. Chakrabarty, and R. M. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *WINE*, pages 566–576, 2008.

Supplementary Material

A Missing Proofs

Lemma 2. *Given a prediction $P = \{\ell_v; u_v\}_{v \in V}$ for* KNAPSACK-FP*, we can create a prediction $P' = \{\ell(v); u(v)\}_{v \in [v_{\min}, v_{\max}]}$ for* C-KNAPSACK-FP *in polynomial time such that*

1. *An α-competitive algorithm for* C-KNAPSACK-FP *with prediction P' implies an $(\alpha - \frac{v_{\max}}{v_{\min}} \cdot |V| \cdot \max_i s_i)$-competitive algorithm for* KNAPSACK-FP *with prediction P.*
2. *If no algorithm has a competitive ratio better than α for* C-KNAPSACK-FP *with prediction P', then no algorithm has a competitive ratio better than $\alpha + \max_i s_i$ for* KNAPSACK-FP *with prediction P.*

Proof. We first describe how to create P' from P. We define a continuous piecewise- linear function $\ell(x)$ as follows. Consider each $v \in V$. Let δ be infinitesimally small. For $x \in [v - \delta, v + \delta]$, let $\ell(v - \delta) = \ell(v + \delta) = 0$, $\ell(v) = \frac{\ell_v}{\delta}$ and interpolate the function values at $v - \delta, v, v + \delta$ by linear functions. It is an easy exercise to show that $\int_{x=v-\delta}^{v+\delta} \ell(x) \mathtt{d}x = \ell_v$ and $\lim_{\delta \to 0} \int_{x=v-\delta}^{v+\delta} x\ell(x) \mathtt{d}x = v\ell_v$. We analogously define $u(x)$ such that $u(v - \delta) = u(v + \delta) = 0$, $u(v) = \frac{u_v}{\delta}$. By construction, we have that $u(x)/\ell(x)$ has the same value for all $x \in (v - \delta, v + \delta)$; thus $\frac{\ell(x)}{\ell_v} = \frac{u(x)}{u_v}$ for all $x \in [v - \delta, v + \delta]$. For all the other x values, let $\ell(x) = 0$. Note that $\ell(x) = 0$ for all $x \notin [v_{\min} - \delta, v_{\max} + \delta]$. Define $u(x)$ similarly. It is easy to see that $\ell(x) \leq u(x)$ by definition.

To show the first claim, consider any α-competitive algorithm A' for C-KNAPSACK-FP with prediction P'. Consider any instance I respecting P. An item i of size s_i and profit p_i in I is converted into s_i items of value approximately equal to $v = p_i/s_i$ in I'. More precisely, we create $\frac{s_i}{\ell_v}\ell(x)$ items of value x for each $x \in [v - \delta, v + \delta]$. Then, the resulting instance I' respects P'. This is because $\frac{s_v}{\ell_v}\ell(x)$ items of value x are created and $\frac{s_v}{\ell_v}\ell(x) \geq \ell(x)$, meaning that the lower bound frequency predictions are respected. Similarly, as observed before, $\frac{s_v}{\ell_v}\ell(x) = \frac{s_v}{u_v}u(x) \leq u(x)$.

Suppose A' has accepted $a'(v)$ items of value approximately equal to $v \in V$—more precisely, in $[v - \delta, v + \delta]$ for some $v \in V$. We define an algorithm A for KNAPSACK-FP trying to keep up with A'. When an item i of value v arrives in I, corresponding items are created in I' and algorithm A' updates a'. Then, A accepts item i if it will have accepted at most $a'(v)$ items of value v after accepting it.

How much can A be behind A'? Let $a(v)$ be the total size of items of value v accepted by A. It is an easy exercise to see that $a'(v) - s_{\max} \leq a(v) \leq a'(v)$, where $s_{\max} := \max_j s_j$ is the maximum item size in I. When $\delta \to 0$, A' gets $a'(v)v$ profit while as A gets $a(v)v$ profit. Thus, A can be behind A' by at most $\sum_{v \in V} vs_{\max} \leq v_{\max} \cdot |V| \cdot s_{\max}$ in terms of profit. Clearly, the optimum is at least $v_{\min}$, which implies an additive loss of at most $\frac{v_{\max}}{v_{\min}} \cdot |V| \cdot s_{\max}$ in the competitive ratio.

We shift to showing the second claim. We use the same P' and I' as above. For the sake of contradiction, suppose there exists an algorithm A for KNAPSACK-FP that has a competitive ratio of $\beta > \alpha + s_{\max}$. Consider the following algorithm A' for C-KNAPSACK-FP: If A accepts an item of value v and size s, we let A' accept the corresponding items. It is easy to see that A and A' achieve the same profit Q at the end as $\delta \to 0$. But the optimum solution for I' has more profit than the optimum solution for I because it can accept items fractionally. The optimum solution for I' accepts items from the highest value until it saturates the knapsack capacity. Thus, the corresponding solution for I accepts items integrally, possibly except one item of some value v'. Therefore, $\text{OPT}' - v's_{\max} \leq \text{OPT} \leq \text{OPT}'$ and $v' \leq \text{OPT}'$ (when $\delta \to 0$), where OPT and OPT$'$ denote the optimum for I and I', respectively. By definition, $\beta = \frac{Q}{\text{OPT}}$. Since $\frac{Q}{\text{OPT}} - \frac{Q}{\text{OPT}'} \leq \frac{Q \cdot v's_{\max}}{\text{OPT} \cdot \text{OPT}'} \leq s_{\max}$, A's competitive ratio is at least $\beta - s_{\max}$, a contradiction. □

Lemma 7. *For any given set $\{\ell(v), u(v)\}_{v \in [v_{\min}, v_{\max}]}$ of frequency predictions, let α^* denote the competitive ratio of the* SENTINEL *algorithm. Then no deterministic algorithm can obtain a competitive ratio larger than α^* on all instances in the family $\{\mathcal{M}(v)\}_{v \in [v_{\min}, v_{\max}]}$.*

Proof. Let $\mathcal{A}$ denote an arbitrary fixed deterministic algorithm. Abusing notation slightly, let $A(v)$ denote the total amount of items of value at most v that are accepted by the algorithm $\mathcal{A}$ and let $a(v) = \frac{\mathtt{d}}{\mathtt{d}x} A(x)|_{x=v}$. Let v_1 be the smallest value v such that $B(v) \neq A(v)$.[v] If there is no such value, then consider the adversarial instance $\mathcal{M}(v_{\max})$. By Claim 5, it follows that the algorithm $\mathcal{A}$ has competitive ratio exactly α^*. Thus assuming that the value v_1 exists, we now consider two cases.

Case 1. $A(v_1) < B(v_1)$. In this case, the adversary stops the instance at $\mathcal{M}(v_1)$. Note that $v_1 > v^*$ since $B(v^*) = 0$ and $B(v)$ is non-decreasing in v. Since the algorithm accepted $a(x) = b(x)$ amount of items for all values less than v_1, less than $b(v_1)$ items of value v_1, and at most $l(v)$ items of each value $v > v_1$, the algorithm's profit is upper bounded by

$$\int_{x=v_{\min}}^{v_1} x \cdot a(x)\mathtt{d}x + \int_{x=v_1}^{v_{\max}} x \cdot \ell(x)\mathtt{d}x < \int_{x=v_{\min}}^{v_1} x \cdot b(x)\mathtt{d}x + \int_{x=v_1}^{v_{\max}} x \cdot \ell(x)\mathtt{d}x = \alpha^* \cdot \textsc{opt}(\mathcal{M}(v_1)),$$

where the last equality follows from Claim 5. Thus, the algorithm fails to achieve the competitive ratio α^*.

Case 2. $A(v_1) > B(v_1)$. Recall that by definition of α^*, we have $B(v_{\max}) = 1$[vi]. As the knapsack capacity is 1, we must have $A(v_{\max}) \leq 1$. Thus, there must exist $v > v_1$ such that $A(v) = B(v)$. Let v_2 be the minimum among all such values v. Note that $v_2 > v^*$ since $B(v^*) = 0$ and $B(v_2) = A(v_2) \geq A(v_1) > B(v_1) \geq 0$. In this case, the adversary stops at v_2 declaring $\mathcal{M}(v_2)$ as the final instance. By definition of v_2, we have $A(v_2) = B(v_2)$ and $A(x) \geq B(x)$ for all $x \leq v_2$. Using integration by parts we have:

$$\begin{aligned}
\int_{x=v_{\min}}^{v_2} x \cdot a(x)\mathtt{d}x &= v_2 \cdot A(v_2) - v_{\min} \cdot A(v_{\min}) - \int_{x=v_{\min}}^{v_2} A(x)\mathtt{d}x \\
&\leq v_2 \cdot B(v_2) - v_{\min} \cdot B(v_{\min}) - \int_{x=v_{\min}}^{v_2} B(x)\mathtt{d}x \\
&= \int_{x=v_{\min}}^{v_2} x \cdot b(x)\mathtt{d}x.
\end{aligned}$$

As before, the total profit of algorithm $\mathcal{A}$ is at most:

$$\int_{x=v_{\min}}^{v_2} x \cdot a(x)\mathtt{d}x + \int_{x=v_2}^{v_{\max}} x \cdot \ell(x)\mathtt{d}x \leq \int_{x=v_{\min}}^{v_2} x \cdot b(x)\mathtt{d}x + \int_{x=v_1}^{v_{\max}} x \cdot \ell(x)\mathtt{d}x = \alpha^* \cdot \textsc{opt}(\mathcal{M}(v_2)).$$

Thus, in all cases, we have shown that no deterministic algorithm can have a competitive ratio better than α. □

Claim 9. *No deterministic online algorithm that maintains a non-decreasing threshold function $\Psi(z)$, where z is the knapsack capacity used, and accepts an item if and only if its value is higher than $\Psi(z)$ has a competitive ratio better than $4/5$ for the family of instances $\mathcal{I}$.*

Proof. Since there are only two distinct item values, assume w.l.o.g. that we have the following threshold function for some parameter λ:

$$\Psi(t) = \begin{cases} 1 & \text{if } t \in [0, \lambda] \\ 2 & \text{if } t \in [\lambda, 1]. \end{cases}$$

We begin by considering the most interesting case when $\lambda \in [1/3, 2/3]$. Suppose λ items of value 2 first arrive, and then, $2/3$ items of value 1 arrive. Then, the algorithm gets profit 2λ as it accepts no item of value 1. On the other hand, the optimum solution can obtain profit $2\lambda + 1 \cdot (1 - \lambda) = 1 + \lambda$. The competitive ratio is upper bounded by $\frac{2\lambda}{1+\lambda} \leq 4/5$ where the equality holds when $\lambda = 2/3$.

[v] Strictly speaking, this may not exist in general in the continuous setting, but we assume wlog that values are sufficiently discretized to avoid such issues.

[vi] The only case that this it not true is that the knapsack is big enough to accept all the items, in which case our algorithm has competitive ratio 1.

Next, suppose $\lambda \in [0, 1/3]$. If only items of value 1 arrive, by 2/3 amount, the algorithm gets profit 1/3, whereas the optimum profit is 2/3. Thus, in this case, the competitive ratio is only 1/2.

Finally, consider the case when $\lambda \in [2/3, 1]$. In this case, $2/3$ amount of items of each value arrive in non-decreasing order of their value. It is an easy exercise to see that the algorithm obtains profit $1 \cdot (2/3) + 2 \cdot (1/3) = 4/3$, but the optimum is $5/3$. Again, we only obtain a competitive ratio of $4/5$. □

Claim 10. *For the instances in $\mathcal{I}$, the algorithm $\mathcal{A}_{\textit{sentinel}}$ yields a competitive ratio of $6/7$.*

Proof. We consider the following algorithm: if the item's value is 2, always accept it; otherwise, we accept up to 4/7 items of value 1. Let A and OPT denote the profit achieved by the algorithm and the optimum, respectively. Suppose x items of value 1 and y items of value 2 arrive. Then, we have,

$$\text{OPT} = 2 \cdot y + 1 \cdot \min\{1 - y, x\}.$$

Clearly, the algorithm can suffer the most when given items of value 1 first. Then, we have

$$A = 2 \cdot \min\{y, 1 - \min\{x, 4/7\}\} + 1 \cdot \min\{x, 4/7\}.$$

Note that $x, y \in [0, 2/3]$ in the instances that respect the given predictions.

We consider the following cases:

1. $x + y \geq 1$: In this case note that $x, y \in [1/3, 2/3]$.
   (a) $x \geq 4/7$:
      i. $y \geq 3/7$:
      $$\frac{A}{\text{OPT}} = \frac{1 \cdot 4/7 + 2 \cdot 3/7}{1 + y} \geq \frac{10/7}{5/3} = 6/7, \text{ where the equality holds when } y = 2/3.$$
      ii. $y < 3/7$:
      $$\frac{A}{\text{OPT}} = \frac{1 \cdot 4/7 + 2 \cdot y}{1 + y} \geq \frac{4/7 + 2/3}{4/3} = 13/14,$$
      where the last inequality follows from $y \geq 1/3$, which is the case because $x \in [0, 2/3]$ due to the given prediction.
   (b) $x < 4/7$:
      i. $y \geq 1 - x$:
      $$\frac{A}{\text{OPT}} = \frac{x + 2(1 - x)}{1 + y} \geq \frac{2 - x}{1 + 2/3} \geq \frac{2 - 4/7}{5/3} = 6/7.$$
      ii. $y < 1 - x$: The inequalities are contradictory.
2. $x + y < 1$:
   (a) $x \geq 4/7$:
      i. $y \geq 3/7$: The inequalities are contradictory.
      ii. $y < 3/7$:
      $$\frac{A}{\text{OPT}} = \frac{1 \cdot 4/7 + 2y}{x + 2y} \geq \frac{4/7}{x} \geq \frac{4/7}{2/3} \geq 6/7$$
   (b) $x < 4/7$:
      i. $y \geq 1 - x$: The inequalities are contradictory.
      ii. $y < 1 - x$:
      $$\frac{A}{\text{OPT}} = \frac{x + 2y}{x + 2y} = 1.$$

In all cases, we have $A/\text{OPT} \geq 6/7$, which establishes that the algorithm A is $(6/7)$-competitive for the instance family $\mathcal{I}$. □

B SENTINEL Algorithm Description

We give a short summary of the SENTINEL algorithm here for clarity.

Algorithm 1 Algorithm SENTINEL for C-KNAPSACK-FP

Input: $P = \{\ell(v); u(v)\}_{v \in [v_{\min}, v_{\max}]}$ as predictions

Compute v^*_α, $\tau_\alpha(\cdot)$, $b_\alpha(\cdot)$, and $B_\alpha(\cdot)$, $\forall \alpha \in [0,1]$ using Equations (1), (2), and (3)
Let α^* be such that $B_{\alpha^*}(v_{\max}) = 1$
Initialize $A(v) = 0,\ \forall v \in [v_{\min}, v_{\max}]$
for each item that arrives online **do**
    Let v denote value of the item
    Let s denote size of the item (infinitesimal)
    if $A(x) < B_{\alpha^*}(x), \forall x \geq v$ **then**
        Accept item
        Update $A(x) \leftarrow A(x) + s, \forall x \geq v$

C Algorithm for Items with Discrete Values

In this section we describe the discrete version of our algorithm SENTINEL for KNAPSACK-FP, while discussing some implementation issues. Recall that an item is assumed to have a value in $V = \{v_{\min} = v_0, v_1, \ldots, v_{k-1} = v_{\max}\}$. For simplicity, we will assume for a while that items can be accepted fractionally; this assumption will be removed at the end of this section. If an item is accepted fractionally, a partial profit is obtained in proportion to the fraction of the item accepted. As before, we are given $l_v \leq u_v$ for each $v \in V$ as prediction such that $s_v \in [l_v, u_v]$. For notational convenience, we will let ℓ_i, u_i, s_i denote $\ell_{v_i}, u_{v_i}, s_{v_i}$, respectively.

As the following steps directly correspond to those of SENTINEL in Section 3, we do not repeat the intuition behind them. Let $\mathcal{M}(v_i, \beta)$ be an instance such that $s_{i'} = \ell_{i'}$ for all $i' > i$, $s_i = \ell_i + \beta(u_i - \ell_i)$, and $s_{i'} = u_{i'}$ for all $i' < i$, where $\ell_{i'}$ items of each value $v_{i'}$ first arrive, and then $s_{i'} - \ell_{i'}$ items of each value $v_{i'}$ arrive in non-decreasing order of their value.

Step 1. Let v_{i^*} denote the largest value in V such that for some $\beta \in [0,1]$, $(1-\beta)\ell_{v_{i^*}} v_{i^*} + \sum_{i>i^*}^{k-1} \ell_{v_i} v_i = \alpha \cdot \text{OPT}(\mathcal{M}(v_{i^*}, \beta))$. Let β_{i^*} denote the value of β.

Step 2. Set $\tau_{i^*} v_{i^*} = \alpha(\text{OPT}(\mathcal{M}(v_{i^*}, 1)) - \text{OPT}(\mathcal{M}(v_{i^*}, \beta_{i^*}))$ for $i = i^*$ and $\tau_i v_i = \alpha(\text{OPT}(\mathcal{M}(v_i, 1)) - \text{OPT}(\mathcal{M}(v_{i-1}, 1))$ for all $i > i^*$.

Step 3. For any value $i \in 0, 1, \ldots, k-1$, we define a budget function as follows: $b_i = 0\ \forall i < i^*$, $b_{i^*} = (1 - \beta_{i^*})\ell_{v_{i^*}} + \tau_{i^*}$, and $b_i = \ell_i + \tau_i\ \forall i > i^*$.

Step 4. Find the value of $\alpha^* \in (0,1]$ by binary search such that $\sum_{k'=0}^{k-1} b_{k'} = 1$ to an arbitrary precision.

We accept an item of value v if $A_i \leq B_i$ for all $i \in \{0, 1, \ldots, k-1\}$ after accepting it, where A_x is the total size of items accepted so far. If items cannot be accepted fractionally, we may lose at most $s_{\max} v_i$ profit for each value $v_i \in V$ where $s_{\max}$ is the maximum item size. It is easy to see that the resulting competitive ratio is at least $\alpha^* - |V| \cdot s_{\max} \cdot v_{\max}/v_{\min}$ following the proof of Lemma 2.

Finally, we discuss how to maintain the invariant $A_i \leq B_i$ efficiently. Consider the following equivalent way of enforcing the invariant: Initially, set $b'_j = b_j$ for all $j \in \{0, 1, 2, ..., k-1\}$. When an item of value v_i and size s arrives, let $j = \arg\max_{i' \in [0,i]}(s \leq b'_i)$. If j does not exist, then we reject the item. Otherwise, we accept it and update b'_j to be $b'_j - s$. It is left as an exercise to verify the equivalence if items have infinitesimal sizes. Using a (balanced) binary search tree, we can handle each arriving item with $O(\log |V|)$ update time.

D Two-stage Knapsack without Predictions

D.1 Algorithm

As before let α^* be the optimal competitive ratio for the problem, which will be decided later. In the absence of predictions, the optimal competitive ratio can be attained by a *threshold* algorithm. In the first stage of the problem, the algorithm will maintain a threshold function depending on the knapsack capacity used and accept an item if and only if its value is higher than the current threshold. Finally, in the second stage, it is easy to observe that any reasonable algorithm accepts the highest value items that can still be accommodated in the knapsack. We now show how to systematically derive such an algorithm.

For any value v, let $z(v)$ be the fraction of the knapsack capacity for which the algorithm maintains the threshold at value v. To derive $z(v)$ we will consider the following family of adversarial instances: (i) items arrive in increasing order of value; (ii) there are enough items of each value to fill the entire knapsack; and (iii) the adversary can stop releasing items at any point in time.

We construct $z(v)$ that is continuous at all $v \in (v_{\min}, v_{\max}]$ as follows. First we set:

$$z(v_{\min}) = \frac{\alpha^* - \lambda}{1 - \lambda}. \tag{4}$$

This is because if the adversary only gives items of value $v_{\min}$ and they are enough to fill up the knapsack, then by accepting $z(v_{\min})$ of them in the first stage and more to the full capacity in the second stage we obtain profit $(z(v_{\min}) + \lambda(1 - z(v_{\min})))v_{\min} = \alpha^* v_{\min}$.

Now consider any value $v > v_{\min}$. If the adversary stops the instance at value v, the profit obtained by the algorithm is:

$$\textsc{alg}(v) := v_{\min} z(v_{\min}) + \int_{x \in (v_{\min}, v]} x \cdot z(x) \mathtt{d}x + \lambda v \Big(1 - z(v_{\min}) - \int_{x \in (v_{\min}, v]} z(x) \mathtt{d}x\Big).$$

Here, the first two terms denote the value earned by the algorithm in the first stage while the last term denotes the profit gained in the second stage. In particular, $(1 - z(v_{\min}) - \int_{x=v_{\min}}^{v} z(x)\mathtt{d}x)$ is the amount of capacity left in the knapsack after the first stage and the algorithm will fill it with the best items—of value v—and receive value of λv per each unit.

To maintain a competitive ratio of α^*, we need to maintain for all $v \in (v_{\min}, v_{\max}]$,

$$v_{\min} z(v_{\min}) + \int_{x \in (v_{\min}, v]} x z(x) \mathtt{d}x + \lambda v \left(1 - z(v_{\min}) - \int_{x \in (v_{\min}, v]} z(x) \mathtt{d}x\right) = \alpha^* v \tag{5}$$

differentiating w.r.t. v,

$$v z(v) + \lambda(1 - z(v_{\min})) - \lambda(v z(v) + \int_{x \in (v_{\min}, v]} z(x) \mathtt{d}x) = \alpha^*$$

rearranging,

$$v z(v) - (\frac{\lambda}{1 - \lambda}) \cdot \int_{x \in (v_{\min}, v]} z(x) \mathtt{d}x = \frac{\alpha^* - \lambda}{1 - \lambda} + \frac{\lambda z(v_{\min})}{1 - \lambda} \tag{6}$$

solving this ODE, for some c, we get

$$z(v) = c v^{\frac{2\lambda - 1}{1 - \lambda}} n.$$

To determine α^* and c, by taking $\lim_{v \to v_{\min}+}$ on (6), we have

$$c\, v_{\min}^{\frac{\lambda}{1-\lambda}} = \frac{\alpha^* - \lambda}{1 - \lambda} + \frac{\lambda z(v_{\min})}{1 - \lambda} = \frac{\alpha^* - \lambda}{1 - \lambda} + \frac{\alpha^* - \lambda}{(1 - \lambda)^2}, \tag{7}$$

where the last equality follows from (4)

Further, we want to use the whole capacity. Thus, from $z(v_{\min}) + \int_{x \in (v_{\min}, v_{\max}]} z(x) \mathtt{d}x = 1$ and (4), we have

$$\frac{\alpha^* - \lambda}{1 - \lambda} + c \frac{1 - \lambda}{\lambda} (v_{\max}^{\frac{\lambda}{1-\lambda}} - v_{\min}^{\frac{\lambda}{1-\lambda}}) = 1 \tag{8}$$

We can determine the value of α^* and c from (7) and (8). Here, we do not explicitly show their closed form as they are not very simple.

We are now ready to describe the algorithm. Set the threshold function $\Psi(t)$ as follows:

$$\Psi(t) = \begin{cases} v_{\min} & \text{if } 0 \leq t \leq \frac{\alpha^* - \lambda}{1-\lambda} \\ Z^{-1}(t) & \text{otherwise,} \end{cases},$$

where $Z(v) := \int_{x=v_{\min}}^{v} z(x)\mathtt{d}x$; this integral includes $z(v_{\min})$. And if we see an item of value v in the first phase when the knapsack is t-full, we accept it if $v \geq \Psi(t)$. In other words, we keep the threshold at $v_{\min}$ until the knapsack gets $\frac{\alpha^* - \lambda}{1-\lambda}$-full; afterwards, keep the threshold at v for $z(v)$ units. In the second phase, we accept the highest valued remaining items to the full capacity.

D.2 Analysis

Suppose the last threshold value used by the algorithm was y. Let A_1 and A_2 denote the items we accepted in the first and second phases, respectively. Let O be the items accepted by the optimum solution. As the algorithm accepts the highest-valued remaining items in the second phase, it must be that $A_2 \subseteq O \setminus A_1$. Let $v(P)$ denote the total profit of items in P and $s(P)$ the total size of items in the same set. Then, the ratio of our algorithm's objective to the optimum is

$$\rho := \frac{v(A_1 \cap O) + v(A_1 \setminus O) + \lambda \cdot v(A_2)}{v(A_1 \cap O) + v(O \setminus A_1 \setminus A_2) + v(A_2)}.$$

For the sake of contradiction suppose $\rho < \alpha^*$. Observe that all items of value greater than y are accepted by the algorithm in the first phase, and also by the optimum solution. To draw a contradiction we change the item values ensuring $\rho < \alpha^*$. First consider the items in A_2. By definition of A_2, all items in A_2 have value at most y. Due to the multiplier λ of $v(A_2)$ in the numerator and the fact that $\lambda \leq \alpha^*$, we have

$$\frac{v(A_1 \cap O) + v(A_1 \setminus O) + \lambda y \cdot s(A_2)}{v(A_1 \cap O) + v(O \setminus A_1 \setminus A_2) + y \cdot s(A_2)} < \alpha^*,$$

which follows from the thought process of increasing the value of items in A_2 to y.

As no items in $O \setminus A_1$ have value greater than y, by increasing the value of the items in $O \setminus A_1$ to y, we can only decrease the ratio. Thus, we have,

$$\frac{v(A_1 \cap O) + v(A_1 \setminus O) + \lambda y \cdot s(A_2)}{v(A_1 \cap O) + y \cdot s(O \setminus A_1 \setminus A_2) + y \cdot s(A_2)} < \alpha^*,$$

Finally, we decrease the value of each item in A_1 to the threshold of our algorithm when it was accepted. As the items in $A_1 \cap O$ appear in both the numerator and denominator and the ratio is less than 1, decreasing the value of an item in $A_1 \cap O$ can only decrease the ratio. Further, it is clear that doing so for items in $A_1 \setminus O$ decreases the ratio as they only appear in the numerator. Finally, knowing that the threshold was always no greater than y, by increasing the value of the items in $A_1 \cap O$ only in the denominator, we have,

$$\frac{v_{\min} z(v_{\min}) + \int_{x \in (v_{\min}, v]} y \cdot z(y)\mathtt{d}x + \lambda y \cdot s(A_2)}{y \cdot s(A_1 \cap O) + y \cdot s(O \setminus A_1 \setminus A_2) + y \cdot s(A_2)} < \alpha^*.$$

The numerator and denominator are $\text{ALG}(y)$ and y respectively. Therefore, the ratio is exactly α^* due to (5), which is a contradiction.

D.3 Optimality

In this section we show that no algorithm can have a competitive ratio better than α^*. We closely follow the optimality proof in Section 3.1.1. We use the same adversarial instances as we used to compute $z(v)$ in Section 5.1. For easy reference, we restate them here: (i) all items arrive in increasing order of their value; (ii) there are enough items of each value to fill up the entire knapsack; and (iii) the adversary can stop releasing items at any point in time.

Fix any deterministic algorithm. Suppose it accepts $a(v)$ items of value v. Let $A(v)$ denote the total size of items of value at most v accepted by the algorithm. If $A(v) = Z(v)$ for all v, then it coincides with our algorithm whose competitive ratio is α^*. So, let $v_1 := \arg\min_{x \in [v_{\min}, v_{\max}]} A(x) \neq Z(x)$.

If $A(v_1) < Z(v_1)$, then the adversary stops at value v_1. Then, the algorithm's profit is less than $\text{ALG}(v_1)$. This is because what the algorithm accepts is different from $\text{ALG}(v_1)$ only in that it accepts more items of value v_1 in the second stage. As $\text{ALG}(v_1) = \alpha^* v_1$ (from Eqn. (5)) and the optimum is v_1, the algorithm has a competitive less than α^*.

In the other case that $A(v_1) > Z(v_1)$, consider $v_2 := \arg\min_{x \in (v_1, v_{\max}]} A(x) = Z(x)$. Note that v_2 must exist as $A(v_{\max}) \leq 1$ and $Z(v_{\max}) = 1$. By definition of v_2, we have $A(v_2) = Z(v_2)$ and $A(x) \geq Z(x)$ for all $x \leq v_2$. Using integration by parts we have:

$$\begin{aligned}\int_{x=v_{\min}}^{v_2} x \cdot a(x) \mathtt{d}x &= v_2 \cdot A(v_2) - v_{\min} \cdot A(v_{\min}) - \int_{x=v_{\min}}^{v_2} A(x) \mathtt{d}x \\ &\leq v_2 \cdot Z(v_2) - v_{\min} \cdot Z(v_{\min}) - \int_{x=v_{\min}}^{v_2} Z(x) \mathtt{d}x \\ &= \int_{x=v_{\min}}^{v_2} x \cdot z(x) \mathtt{d}x.\end{aligned}$$

What this means is that the algorithm has obtained less profit in the first stage as oppose to $\text{ALG}(v_2)$ using the same capacity $A(v_2)$. Thus, the algorithm's value is at most $\text{ALG}(v_2) = \alpha^* v_2$. As the optimum is v_2, the competitive ratio is no better than α^*.

Using the same reasoning as we used before, we know that randomization does not help. Thus, we have shown that no randomized online algorithm has a competitive ratio better than α^*.

E Two-Stage Knapsack with Predictions

E.1 Algorithm

We now extend our algorithm for the two-stage knapsack problem to the setting where we are given frequency predictions. As before, we will focus on the fully continuous version of the problem, which is exactly C-KNAPSACK-FP except with two stages. We adopt the same notation as we used in Section 3, with minor changes. Let $\text{OPT}(\mathcal{I})$ be either the optimum solution or its profit for the two-stage knapsack problem when $\mathcal{I}$ is the given input. Define $\text{OPT}(\mathcal{I}, \kappa)$ analogously but assuming that the knapsack has a reduced capacity, κ.

Because we derive our algorithm closely following the steps we took in Sections 3 and 5.1, we will omit details that are repeated. As before, we will find a competitive ratio that is close to the best competitive ratio α^* to an arbitrary precision using a binary search over $[\lambda, 1]$. We use the same instance $\mathcal{M}(v)$ as defined in Section 3.

Step 1. Find v^* such that

$$\int_{x=v^*}^{v_{\max}} x \cdot \ell(x) \mathtt{d}x = \frac{\alpha^* - \lambda}{1 - \lambda} \cdot \text{OPT}(\mathcal{M}(v^*)).$$

Under the assumptions we made for C-KNAPSACK-FP, it can be shown that v^* exists. Note that this generalizes Step 1 for the single-stage knapsack with frequency predictions in Section 3 and we can recover it when $\lambda = 0$. The difference is that in addition to the value $\int_{x=v^*}^{v_{\max}} x \cdot \ell(x) \mathtt{d}x$, the algorithm can accept $\lambda(\text{OPT}(\mathcal{M}(v^*)) - \int_{x=v^*}^{v_{\max}} x \cdot \ell(x) \mathtt{d}x)$ extra profit in the second stage. Here, we used an easy observation that $\text{OPT}(\mathcal{M}(v^*))$ includes all items of value greater than v^* in $\mathcal{M}(v^*)$.

Step 2. In this step we define a function $\tau : [v^*, v_{\max}] \to [0, \infty)$ to determine the additional amount of items of a particular value that our algorithm would like to accept. Let $\mathcal{R}(v)$ denote an instance consisting of $u(x) - \ell(x) - \tau(x)$ items of each value $x \in [v^*, v]$ and $u(x)$ items of each value $x \in [v_{\min}, v^*]$. Let $B(v) := \int_{x=v^*}^{v} \ell(x) + \tau(x) \mathtt{d}x$ for $v \in [v^*, v_{\max}]$; and $B(v) = 0$ for $v \in [v_{\min}, v^*]$. For each $v \in [v^*, v_{\max}]$, let

$$\int_{x=v^*}^{v_{\max}} x \cdot \ell(x) \mathtt{d}x + \int_{x=v^*}^{v} x \cdot \tau(x) \mathtt{d}x + \lambda \text{OPT}(\mathcal{R}(v), 1 - B(v)) = \alpha^* \text{OPT}(\mathcal{M}(v)). \tag{9}$$

Here, the first two terms in the left-hand-side are our algorithm's profit in the first stage and the third is that in the second stage. Taking derivatives w.r.t. v, we solve the equations.

Step 3. The previous steps are performed using the predictions $\{\ell(x); u(x)\}_{x\in[v_{\min},v_{\max}]}$ before seeing the actual items. Then, when an item of value v arrives in the first stage, the algorithm accepts it if and only if $A_1(x) < B(x)$, $\forall x \geq v$, where $A_1(x)$ denotes the total amount of items of value at most v that has been accepted by the online algorithm so far (in the first stage). In the second stage, the algorithm accepts the highest-valued remaining items to the full residual capacity.

Step 4. As before, we can find the desired α^* such that $B(v_{\max}) = 1$ by binary search over $[\lambda, 1]$. A minor difference from the previous binary search is that if $B(v) \geq 1$ for some v in deriving $B(\cdot)$, we stop and decrease the value of α. This is to avoid the residual capacity, i.e., $1 - B(v)$, becoming negative in Eqn. (9). However, for a fixed value v, we can still view $B(v)$ as a function of α^* and easily verify that it is continuous and mononotically increasing in α^* as long as $B(v) < 1$. Thus, this modified binary search is well-defined.

E.2 Analysis

We show that the algorithm presented in the previous section indeed has a competitive ratio of α^* and that it is the best possible one can hope for.

At the end of the algorithm's execution, let y be the largest value v such that $A_1(v) = B(v)$. This implies $A_1(v) < B(v)$ for all $v > y$. Let H denote all the items of value greater than y. Let G be a set of items consisting of arbitrary $\ell(v)$ items for each $v \in [v_{\min}, v_{\max}]$ among those that actually arrive. It is an easy observation that all items in H are accepted by the algorithm in the first phase, and also by the optimum solution. Let A_1 and A_2 denote the items we accepted in the first and second phases, respectively, with the items in H excluded. Recall that $A_1(v)$ denotes the total size of items of value at most v accepted by our algorithm in the first stage, and it is distinguished from A_1. Let O be the items not in H that are accepted by the optimum solution. As the algorithm accepts the highest-valued remaining items in the second phase, it must be that $A_2 \subseteq O \setminus A_1$. Let $v(P)$ denote the total profit of items in P and $s(P)$ the total size of items in the same set. Then, the ratio of our algorithm's profit to the optimum is the following:

$$\rho := \frac{v(H \setminus G) + v(H \cap G) + v(A_1) + \lambda \cdot v(A_2)}{v(H \setminus G) + v(H \cap G) + v(O)}.$$

For the sake of contradiction suppose $\rho < \alpha^*$. Recall that $\lambda \leq \alpha^*$. Say our algorithm accepts $a_1(v)$ and $a_2(v)$ items of value v in the first and second stages, respectively, and the optimum accepts $o(v)$ items of value v. Let $a(v) = a_1(v) + a_2(v)$.

Our proof strategy is to make a sequence of changes to modify our algorithm's solution and/or some items' value, ensuring (i) ρ never increases; (ii) the instance continues respecting the prediction; (iii) the algorithm makes the optimal choices in the second stage regarding the remaining items; and (iv) the optimum solution remains optimum. After all the changes, we will show that $\rho = \alpha^*$ to draw a contradiction.

First Change. We first show that we can assume wlog that $a_1(x) = b(x)$ for all $x \in [v^*, y]$ for analysis. Observe that there exists y_1 such that $a(x) = s(x)$ for all $x \in (y_1, v^*)$ and $a_2(x) = 0$ for all $x \in [v_{\min}, y_1)$ due to the greedy behavior of our algorithm in the second stage and the fact that all items in H are accepted in the first stage. Here, we only change a_1, a_2, therefore, the denominator of ρ, which is the optimum profit, will remain unchanged. Thus, (iv) will hold. We will consider a sequence of changes that only decreases the numerator.

Assume $a_1(x) \neq b(x)$ for some $x \in [v^*, y]$ since otherwise there is nothing to prove. Let v_3 denote the largest $x \in (v^*, y)$ such that $a_1(x) > b(x)$; it is an easy exercise to show v_3 exists from the definition of y. Similarly, let v_1 denote the smallest $x \in (v^*, y)$ such that $a_1(x) < b(x)$. Note that $v_1 < v_3$ from the fact that $A_1(y) = B(y)$ and $A_1(x) \leq B(x)$ for all $x \in [v^*, y]$; and $A_1(v^*) = 0$.

We consider two cases. First consider the case that $v_3 \geq y_1$. Let $v_2 := \max\{v_1, \min(A_2)\}$, where $\min(A_2)$ denotes the min value of items in A_2. By definition, $v_1 \leq v_2 \leq v_3$. We make the following changes: decrease $a_1(v_3)$ by an infinitesimally small $\delta > 0$; increase $a_2(v_3)$ by δ; increase

$a_1(v_1)$ by δ; and decrease $a_2(v_2)$ by δ. Then, the numerator, the algorithm's profit, changes by $\delta(-v_3 + \lambda v_3 - \lambda v_2 + v_1) \leq \delta(-v_3 + \lambda v_3 - \lambda v_2 + v_2) = \delta(v_3 - v_2)(\lambda - 1) \leq 0$.

Now consider the other case that $v_3 \leq y_1$. Then, this case is easier to track the changes as we do not have to consider items in A_2. In this case, we decrease $a_1(v_3)$ by δ and increase $a_1(v_1)$ by δ. It is trivial to see that this change decreases the numerator, i.e., our algorithm's profit.

We have shown (i). It is easy to see that (ii) and (iii) are never violated under the changes we have made. As a result of the changes, we have $a_1(x) = b(x)$ for all $x \in [v^*, y]$ and $a_2(x) = s(x) - b(x)$ for all $x \in [y_1, y]$ and $a_2(x) = 0$ for all $x \in [v_{\min}, y_1]$.

Second Change. In this change, we increase the value of some items in A_2 to make $a(x) = s(x) = u(x)$ if $a_2(x) > 0$, and possibly change $o(\cdot)$ to keep satisfying (iv). To do this, we first increase the value of items in A_2 maximally up to value y ensuring (ii). Since $A_2 \subseteq O$, such items appear in both the numerator and denominator. But, $v(A_2)$ has a multiplier λ in the numerator. As discussed, $\lambda < \alpha^*$. Thus, if we make these changes, we will continue to have (i). It is trivial to see that (iii) holds true. To satisfy (iv), we let the optimum solution accepts $1 - s(H)$ highest valued items from the set of items consisting of $u(x)$ items $x \in [v_{\min}, v^*]$; here, we let $s(x) = u(x)$ for all $x \in [v_{\min}, v^*]$, and this doesn't change (iii). As this increases only the denominator, we still have (i). It is an exercise to see that we made changes respecting the prediction, so we have (ii) as well.

As a result, we have $s(x) = u(x) \ \ \forall x \in [v_{\min}, v^*]$, $a_1(x) = b(x) \ \ \forall x \in [v^*, y]$, $a_2(x) = u(x) - b(x)$ $\forall x \in [y_1, y]$ and $a_2(x) = 0 \ \ \forall x \in [v_{\min}, y_1]$.

Third Change. Here we decrease the value of some items in $H \setminus G$. As both our algorithm and the optimum solution accepts such items and get the full face value from them, it will only lower ρ. So, we will have (i). Specifically, we let $s(x) = u(x)$ for all $x \in [v^*, y_2]$, and $\int_{v^*}^{y_2} (s(x) - \ell(x))\mathtt{d}x = s(H \setminus G)$. It is easy to check all (i)–(iv) hold true.

Fourth Change. Now we only change $a_1(x)$ and $a_2(x)$ for $x \in [y, y_2]$. Thus, this will only affect the numerator. After the third change $a_1(x) = u(x)$ for all $x \in [y, y_2]$. For each $x \in [y, y_2]$, we set $a_1(x) = b(x)$ and $a_2(x) = u(x) - b(x)$. Clearly this can only decrease the numerator. And all (i)–(iv) hold true.

After all the changes, the resulting instance is $\mathcal{M}(y_2)$ and the algorithm's solution has the following form:

$$a_1(x) = \begin{cases} \ell(x) & x \in [y_2, v_{\max}] \\ b(x) & x \in [v^*, y_2] \\ 0 & \text{otherwise,} \end{cases}$$

$$a_2(x) = \begin{cases} 0 & x \in [y_2, v_{\max}] \\ u(x) - b(x) & x \in [y_1, y_2] \\ 0 & \text{otherwise,} \end{cases}$$

which is exactly what our algorithm accepts for $\mathcal{M}(y_2)$. Thus, the numerator of ρ is exactly the LHS in Eqn. (9). Thus, we have shown that $\rho^* = \alpha$, which is a contradiction.

E.3 Optimality

We show that no algorithm can have a competitive ratio better than α^*. The proof is very similar to that for the case without predictions in Section D.3. We use the same adversarial instances $\{\mathcal{M}(v)\}$ as we used to compute $b(v)$ in Section 5.1; see Section 3.1.1 for the definition.

Consider any fixed deterministic algorithm $\mathcal{A}$ since randomization doesn't help as observed before. Adopting the notation we defined in Section E.2, let $a_1(v)$ denote the amount of items of value v accepted by $\mathcal{A}$ in the first stage. Define $o(v)$ analogously for the optimum solution. Let $A_1(v)$ denote the total size of items of value at most v accepted by $\mathcal{A}$ in the first stage.

If $A_1(v) = B(v)$ for all v, then it coincides with our algorithm whose competitive ratio is α^*. So, let $v_1 := \arg\min_{x \in [v_{\min}, v_{\max}]} A_1(x) \neq B(x)$. We consider two cases as follows.

If $A_1(v_1) < Z(v_1)$, then the adversary declares that the instance is $\mathcal{M}(v_1)$. Then, $\mathcal{A}$ gets less profit than our algorithm because it produces an identical solution as ours except that it accepts more items of value v_1 in the second stage (but the same amount in both stages together). Thus, $\mathcal{A}$ obtains profit less than the LHS in Eqn. (9) and the optimum is v_1. This implies that $\mathcal{A}$ has a competitive ratio less than α^*.

Consider the other case, $A_1(v_1) > B(v_1)$. Let $v_2 := \arg\min_{x \in (v_1, v_{\max}]} A_1(x) = B(x)$. Note that v_2 must exist as $A_1(v_{\max}) \leq 1$ and $B(v_{\max}) = 1$. As before, we can show $\int_{x=v_{\min}}^{v_2} x \cdot a_1(x) \mathtt{d}x \leq \int_{x=v_{\min}}^{v_2} x \cdot b(x) \mathtt{d}x$. But, we need a slightly stronger claim here. By definition of v_2, we have $A_1(v_2) = B(v_2)$ and $A_1(x) \geq B(x)$ for all $x \leq v_2$. Then, we can define a one-to-one mapping ψ from the items accepted by $\mathcal{A}$ to the set consisting of $b(x)$ items for $x \in [v_{\min}, v_2]$ such that item e has no greater value than $\psi(e)$. This mapping can be constructed by considering items in decreasing order of their value and mapping them sequentially.

What this means is the following. For the sake of analysis, pretend that $\mathcal{A}$ is given an option immediately after the first stage to choose $\psi(e)$ over e it has accepted. It is an easy exercise to see that $\mathcal{A}$ makes all the swaps it is allowed to increase its profit before the second stage starts. Thus, we have shown that $\mathcal{A}$ obtains profit no greater than our algorithm does for $\mathcal{M}(v_2)$. This implies that $\mathcal{A}$'s competitive ratio is at most α^*.