Augmenting Online Algorithms for Knapsack Problem with Total Weight Information

Binghan Wu, Wei Bao, Bing Zhou

Faculty of Engineering, The University of Sydney
biwu6051@uni.sydney.edu.au, wei.bao@sydney.edu.au, bing.zhou@sydney.edu.au

Abstract

In this paper, we augment online algorithms for the knapsack problem using the total weight information. The conventional optimal online algorithm achieves the $\ln(\frac{U}{L})+1$ competitive ratio where L and U are the upper and lower bounds of the value-to-weight ratio. However, it does not consider that decision makers can know the total weight information or obtain it through machine-learned predictions. To fill this gap, we first propose the Known Weight Algorithm (KWA) which uses the exact total weight information to achieve a competitive ratio of $\mathcal{W}(\frac{1}{e}\frac{U-L}{L})+1$, where $\mathcal{W}$ denotes the Lambert-W function. We prove that it is optimal and tight. After that, we extend KWA to the Predicted Weight Algorithm (PWA), a learning-augmented online algorithm that uses predicted total weight. We show the consistency and robustness (Lykouris and Vassilvitskii 2021) of PWA, and prove that its competitive ratio degrades gracefully as the prediction error grows. Finally, we introduce the Limited Volume Algorithm (LWA), which achieves a better competitive ratio than $\ln\frac{U}{L}+1$ when the total weight is less than twice the capacity.

Introduction

The 0–1 knapsack problem (Jooken, Leyman, and De Causmaecker 2022) is one of the most well-known problems in computer science, dealing with the optimization of item selection under capacity constraints. Suppose we have a knapsack with a normalized capacity of 1. In the online setting, items arrive sequentially. Upon the arrival of the ith item, the value v_i and the weight w_i are revealed. We need to make an irrevocable decision to either accept or discard this item without knowledge of future arrivals. Conventionally, the performance of the online knapsack problem is measured by the *competitive ratio*, defined as the maximum ratio of the offline optimal algorithm's value to the online algorithm's value across all possible inputs. In the general setting, the competitive ratio for the online knapsack problem is unbounded. However, the optimal competitive ratio $\ln(\frac{U}{L})+1$ can be achieved when the value-to-weight ratios $\frac{v_i}{w_i}$ for all items are bounded by $[L,U]$, and the weights are infinitesimal compared to the capacity, i.e., $\max_i w_i \ll 1$.

While the classical online knapsack problem assumes no access to future information, the total weight of items is often predictable and sometimes even known to the decision-maker. For example, a logistics company may need to load cargo into containers to maximize the value of the items loaded. Such a company can estimate the total loading demand (Anguita and Olariaga 2023; Filom, Amiri, and Razavi 2022) using machine learning methods based on historical data. This scenario reveals a gap in both the literature and practical applications. We seek to fill this gap by addressing a central question: In situations where the total weight is known or predicted, how can this information be used to improve the competitive ratio and surpass the barrier of $\ln(\frac{U}{L})+1$? We aim to investigate how incorporating total weight information reduces uncertainty and the best ways to utilize it. Besides this cargo loading example, there are many real-world scenarios where total weight information is known or predictable. Here are some more examples.

1. One-way trading problem (Schroeder, Dochow, and Schmidt 2018) with known ending time. Consider a trader planning to exchange a total amount of 1 dollar into yen. Exchange rates $p_1, p_2, \ldots, p_n$ arrive sequentially, where n is the trading horizon or the deadline. The trader must immediately decide whether to accept the exchange rate and trade $w_i = \epsilon$ dollars for $p_i\epsilon = v_i$ yens. ϵ is the trading granularity or the maximum trading amount that does not affect the market price. The goal is to maximize the yen traded after processing the entire sequence while respecting the budget constraint. In the classical one-way trading problem, the deadline n is unknown to the trader, and the trading process can be terminated at any time. However, this is often impractical in real-world scenarios. For example, in forex or stock markets, the market close time is known to traders.

2. Online cloud resource allocation using a posted price mechanism (Zhang, Li, and Wu 2017) with predicted total demand volume. The cloud service provider (CSP) wants to sell a total amount of 1 resource by publishing a unit resource price p_i, which can change over time. A total of n customers arrive sequentially. Each customer wants to buy w_i amount of resources from the CSP, and customer i's intrinsic valuation is v_i. When customer i arrives, the CSP reveals the unit price p_i to them. If $v_i \geq w_i p_i$ and the remaining resources are sufficient, customer i buys w_i units from the CSP by paying $w_i p_i$; otherwise, they leave the system. We aim to maximize social welfare (Zhang, Li, and Wu 2017), defined as the sum of the CSP's revenue and the customers' gains. In traditional methods, the total demand volume of resources is unknown to the CSP. However, this

approach does not align with real-world scenarios where statistical and machine learning methods can predict the total demand volume (Amiri and Mohammad-Khanli 2017; Zhu, Luo, and Deng 2020).

To address this problem, we first introduce the Known Weight Algorithm (KWA) which improves the competitive ratio when the exact total weight is known. An initial intuition might suggest that decisions need to depend on the total weight of the remaining items, which adds an additional dimension to the algorithm. When the total weight of the remaining items can fit into the knapsack, we accept all items. However, if not, we discovered a threshold function that does not rely on the total weight of the remaining items. This function enables us to accept newly arrived items when their value-to-weight ratio exceeds the threshold, and it surprisingly achieves the best possible competitive ratio. We conduct a rigorous competitive analysis and obtain the following results: First, we prove that KWA is $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$-competitive, where $\mathcal{W}$ is the Lambert-$\mathcal{W}$ function (Corless et al. 1996), and it is better than the $\ln(\frac{U}{L}) + 1$ competitive ratio. To the best of our knowledge, we are the first to propose a $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ competitive ratio in this field. Then, we prove that this competitive ratio is *reachable* by a worst-case problem instance. Finally, we prove that $\mathcal{W}(\frac{1}{e}\frac{U-L}{L})+1$ is the best possible competitive ratio for this problem.

Furthermore, we extend the result to the case where the total weight is provided by an imperfect machine-learned prediction, which aligns with the learning-augmented algorithm paradigm (Im et al. 2021). Despite the potential inaccuracies in these predictions, the algorithm is designed to achieve better than $\ln(\frac{U}{L})+1$ performance when the predictions are accurate, while still maintaining strong worst-case guarantees even when the prediction is misleading. We propose the learning-augmented Predicted Weight Algorithm (PWA), using λ capacity to perform KWA, and $(1-\lambda)$ capacity to perform the classical online knapsack algorithm (OKA). We evaluate it based on two critical objectives: (1) *consistency* (Lykouris and Vassilvitskii 2021), which measures the algorithm's performance when the advice is accurate; and (2) *robustness* (Lykouris and Vassilvitskii 2021), indicating how the algorithm performs across all possible input sequences, independent of the advice's accuracy. We use λ as the parameter that allows our algorithm to trade off between consistency and robustness smoothly, because the confidence in prediction varies across different application scenarios. Let $c_1 = \mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$, and $c_2 = \ln(\frac{U}{L}) + 1$. We prove that PWA is $\frac{c_2}{1-\lambda}$ robust and $\frac{c_1 c_2}{\lambda c_2 + (1-\lambda)c_1}$ consistent. Furthermore, since perfect predictions do not exist in real-world scenarios, we show that the competitive ratio degrades gracefully as the prediction error grows.

Finally, as a supplementary method of improving the competitive ratio using total weight information, we propose the Limited Weight Algorithm (LWA), which works in special cases when the exact total weight is known and less than twice the capacity. In this case, the decisions of the online algorithm and the optimal offline algorithm must overlap (considering the extreme case where there is only one item). We exploit this property to improve the competitive ratio to better than $\ln(\frac{U}{L})+1$. It is an alternative approach for using the total weight information to improve the competitive ratio, and can also be used together with PWA to achieve additional gains in special cases.

Literature Review

Many variants of the online knapsack problems and the closely related online one-way trading problems where items can be fractional and unlimited weight (Cao, Sun, and Tsang 2020; Zhang, Li, and Wu 2017) have been investigated by the literature. (Sun et al. 2022) investigate the online knapsack problem with departures, where items only remain in knapsacks for a finite duration. It captures situations where jobs will leave after completion in computational resource allocation. (Han, Kawase, and Makino 2014) allows items to be removed with a cancellation charge or disposal fee. (Zhang, Li, and Wu 2017) jointly study multiple dimensions and reusable resources after item departure. They show a competitive ratio bound in a complex form, and prove its tightness. Unlike these works, our work in focuses on how the total weight information can be utilized to improve competitive ratio. All of these works can use our conclusions to improve performance with the total weight information. (Lin et al. 2019) generalizes the online one-way trading problem to concave revenue functions. They reach the competitive ratio within a small additive constant to $\ln(\frac{U}{L}) + 1$. (Lin et al. 2022) investigate the online one-way trading of multiple inventories. They use a divide-and-conquer approach to treat each single inventory first, and aggregate the result. They achieve the competitive ratio in $[\ln(\frac{U}{L})+1, \ln(\frac{U}{L})+2]$. (Yang et al. 2021) extends the problem to multi-dimensions. They reach a competitive ratio $O(\log(\frac{U}{L}\alpha))$, where α is the maximum ratio between the total capacity and the capacity for one resource. (El-Yaniv et al. 2001) investigates several variants of the online one-way trading problem, including the case when the deadline n is known for the trading. They showed a competitive ratio R^* as the solution of the equation $n(1-(\frac{L(R^*-1)}{U-L})^{\frac{1}{n}}) - R^* = 0$. Interestingly, their algorithm and competitive ratio are equivalent to KWA when n goes to infinity. However, our proof is completely different from (El-Yaniv et al. 2001), which leads to a more essential threshold function and a neat competitive ratio $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$. To the best of our knowledge, we are the first to propose the competitive ratio using the Lambert-W function.

Learning-Augmented Online Algorithms

The study of online algorithms has greatly benefited from worst-case analysis to deal with the uncertainty of the future, which involves the competitive ratio under the worst possible conditions. This type of analysis allows us to use algorithms confidently as building blocks and subroutines, knowing their worst-case performance. However, in scenarios where future information can be obtained through machine learning predictions, we are motivated to enhance the performance of online algorithms with predictions. Therefore, learning-augmented online algorithms have been studied in a recent line of work initiated by Lykouris and Vassilvitskii (Lykouris and Vassilvitskii 2021) (first appeared in 2018) and Kumar, Purohit and Svitkina (Purohit, Svitk-

ina, and Kumar 2018). The metrics of performance then shift to (1) consistency: the competitive ratio when the predictions are accurate; (2) robustness: the competitive ratio when the predictions are arbitrarily bad. Using consistency and robustness as a provable measurement to analyze the enhancement of predictions to the online algorithms are widely adopted in recent years. Studies related to our topic include online knapsack with frequency prediction (Im et al. 2021), online single resource allocation problem (Hwang, Jaillet, and Manshadi 2021), and airline fare class allocation with customer distribution prediction (Balseiro, Kroer, and Kumar 2023). (Zeynali et al. 2021) discusses how the competitive ratio degrades for learning-augmented algorithms when the aggressiveness are adjusted to accommodate predictions in the online knapsack problem. However, we use the prediction of total weight to enhance performance in the online knapsack problem, adding an extra dimension where decisions are related to the remaining items, fundamentally distinguishing our work from theirs.

Problem Formulation

We consider the online knapsack problem with total weight information. Specifically, we are given a knapsack with a capacity of 1. A sequence of n items arrives one by one. Upon the arrival of item i, its value v_i and weight w_i are revealed. The value-to-weight ratios are bounded by $\frac{v_i}{w_i} \in [L, U]$ for all items. Additionally, when the first item arrives, the decision maker is also provided with the total weight information. If this total weight information is exact, denoted by $W = \sum_{i=1}^{n} w_i$, we refer to the problem as the *known total weight problem*. If the total weight information is given as a machine-learned prediction, denoted as $\hat{W}$, we refer to it as the *predicted total weight problem*.

Let x_i be the decision variable for item i, where $x_i = 1$ indicates the acceptance of the item and $x_i = 0$ indicates its rejection. The decision maker must make an irrevocable decision to accept or reject each item upon its arrival, without any knowledge of the subsequent items except for the total weight information W or $\hat{W}$.

The corresponding offline problem can be formulated as follows:

$$\max \quad \sum_{i=1}^{n} v_i x_i, \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leq 1, \tag{2}$$

$$x_i \in \{0, 1\}, \quad \forall i \in [n]. \tag{3}$$

Let $\mathcal{I}$ denote the set of all possible inputs. For a problem instance $I \in \mathcal{I}$, we denote the offline optimal solution of (1–3) as $\texttt{OPT}(I)$. We refer to it as the *value* of OPT when I is clear from the context.

For the known total weight problem (online), let $\texttt{ALG}(I)$ represent the final objective value of the online algorithm ALG for a given instance I. We refer to it as the *value of* ALG when I is clear from the context. The performance of this algorithm is evaluated using the competitive ratio, R^*. We define the competitive ratio R^* of the online algorithm ALG as follows:

$$R^* \triangleq \sup_{I \in \mathcal{I}} \frac{\texttt{OPT}(I)}{\texttt{ALG}(I)}. \tag{4}$$

The competitive ratio is the *worst-case* bound on the ratio of the optimal offline algorithm's value to that of the online algorithm. The lower the ratio, the better the online algorithm.

For the predicted total weight problem (online), we develop our algorithm in the learning-augmented online algorithm framework (Purohit, Svitkina, and Kumar 2018). In this framework, we focus on three issues: First, what is the competitive ratio of the algorithm when the given prediction is accurate (*consistency*)? Second, what is the competitive ratio of the algorithm when the prediction is arbitrary (*robustness*)? Finally, how does the algorithm's competitive ratio change as the error of the prediction increases? Next, we rigorously define the terms and concepts mentioned above.

Let $\texttt{ALG}(I, \hat{W})$ denote the value of ALG given problem instance I and the prediction $\hat{W}$. A prediction of the total weight W is a real number $\hat{W} \in \mathbb{R}$. The prediction error is $\eta \triangleq |\hat{W} - W|$ given the prediction $\hat{W}$ and the problem instance I. Let $\mathcal{I}_{\hat{W}} = \{I \in \mathcal{I} | \sum_{i=1}^{n} v_i = \hat{W}\}$ be the set of input that complies with the prediction $\hat{W}$. The online algorithm ALG with predicted volume $\hat{W}$ is β–consistent (Lykouris and Vassilvitskii 2021) if

$$\sup_{\hat{W}, I \in \mathcal{I}_{\hat{W}}} \frac{\texttt{OPT}(I)}{\texttt{ALG}(I, \hat{W})} \leq \beta. \tag{5}$$

The online algorithm ALG with predicted total weight $\hat{W}$ is γ–robust (Lykouris and Vassilvitskii 2021) if

$$\sup_{\hat{W}, I \in \mathcal{I}} \frac{\texttt{OPT}(I)}{\texttt{ALG}(I, \hat{W})} \leq \gamma. \tag{6}$$

Known Weight Algorithm (KWA)

The Known Weight Algorithm (KWA, Algorithm 1) is a threshold-based approach. The primary difference from the classical online knapsack algorithm (OKA) (Zhang, Li, and Wu 2017) is its new fill-up logic: If the knapsack's remaining capacity can accommodate all remaining items, we accept them. This ensures that the knapsack is effectively filled. When the total weight of items exceeds the knapsack's capacity, the final remaining capacity must be infinitesimal. If the total weight of remaining items exceeds the knapsack's capacity, we only accept items with sufficient value. We define a threshold function to determine whether an item is worth placing in the knapsack. At first glance, it might seem that the total weight of the remaining items should serve as an input to the threshold function, given this additional information compared to the classical algorithm. However, we discovered a threshold function ϕ_1 that does not require the total weight of the remaining items as input and still achieves the optimal competitive ratio:

$$\phi_1(y) = L + (\theta_1 - L) e^{\frac{\theta_1}{L} y} \; \forall y \in [0, 1], \tag{7}$$

$$\theta_1 = L(\mathcal{W}(\frac{1}{e}\frac{U - L}{L}) + 1), \tag{8}$$

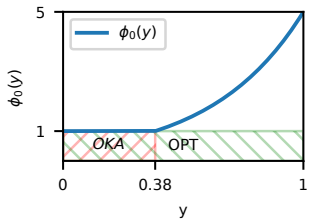


Figure 1: An example of ϕ_0 for OKA, $L = 1$, $U = 5$. When all items' value-to-weight ratios are L, the values of OKA and OPT are shown by the red and green areas, respectively.

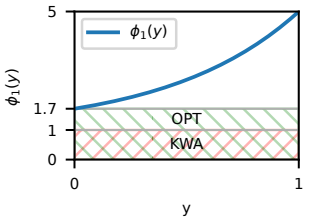


Figure 2: Example of ϕ_1 for KWA, with $L = 1$, $U = 5$. If items with a value-to-weight ratio θ_1 arrive first, then items with ratio L, KWA and OPT are represented by the red and green shaded areas, respectively.

where $\mathcal{W}$ is the Lambert-W function. The Lambert-W function is the inverse function of

$$f(w) = we^w. \tag{9}$$

We defer the discussion of why considering the total weight of the remaining items cannot improve the competitive ratio to Theorem 3. ϕ_1 takes a parameter y which is the current capacity utilization, and $\phi_1(y)$ represents the minimum value-to-weight ratio that is acceptable at the current capacity utilization y.

Algorithm 1 first initializes the current capacity utilization to $y_0 = 0$ (Line 3). Upon the arrival of item i, its weight w_i and value v_i are revealed (Line 5). If the remaining capacity can hold all the remaining items (Line 6), then we accept item i and update the capacity utilization to $y_i = y_{i-1} + w_i$ (Line 7). If not, but the remaining capacity can hold item i and item i's value is higher than the threshold (Line 8), then we accept item i and update the capacity utilization (Line 9). Otherwise, we reject this item, and the capacity utilization remains the same (Line 11).

The performance of Algorithm 1 relies on the threshold function ϕ_1. It is worth noting that the OKA (without the fill-up logic Lines 6–7) adopts a different threshold function ϕ_0:

$$\phi_0(y) = \begin{cases} L, \ y \in [0, \frac{1}{\ln(\frac{U}{L})+1}], \\ \frac{L}{e} e^{\left(\ln(\frac{U}{L})+1\right)y}, \ y \in (\frac{1}{\ln(\frac{U}{L})+1}, 1]. \end{cases} \tag{10}$$

ϕ_0 is a piecewise function. When the capacity utilization $y \leq \frac{1}{\ln(\frac{U}{L})+1}$, the presented price is L. This segment ensures that a portion of the capacity can always be filled by accepting items with the lowest value-to-weight ratio, thereby providing a lower bound for the total value. An example of ϕ_0 with $L = 1$ and $U = 5$ is illustrated in Figure 1, where all items have value-to-weight ratios of L. The value of OKA is $\frac{L}{\ln(\frac{U}{L})+1}$ (shown as the red area), and the value of OPT is L (shown as the green area). Thus, the competitive ratio of OKA is at least $\ln(\frac{U}{L}) + 1$, approximately 2.6. The piecewise property is important when the total weight is unknown. Otherwise, if the algorithm sets a higher threshold than L at $y = 0$, in the worst case, all items' value-to-weight ratios are L, then the value of the online algorithm is 0 (while OPT is L), and therefore the competitive ratio is unbounded.

Algorithm 1: Known Weight Algorithm (KWA)

1: **Input:** $I = \{(w_i, v_i), \forall i \in [n], L, U\}$, $W = \sum_{i=1}^{n} w_i$.
2: **Output:** $x_i, \forall i \in [n]$.
3: $y_0 = 0$;
4: **for** $i \in [n]$ **do**
5: item i arrives, and (w_i, v_i) is revealed;
6: **if** $W - \sum_{j=1}^{i-1} w_j \leq 1 - y_{i-1}$ **then**
7: accept item i, $x_i = 1$, $y_i = y_{i-1} + w_i$;
8: **else if** $y_{i-1} + w_i \leq 1$ and $v_i \geq \int_{y_{i-1}}^{y_{i-1}+w_i} \phi_1(\delta)d\delta$ **then**
9: accept item i, $x_i = 1$, $y_i = y_{i-1} + w_i$;
10: **else**
11: reject item i, $x_i = 0$, $y_i = y_{i-1}$.
12: **end if**
13: **end for**

However, when the total weight is known, this segment is no longer necessary because we can now guarantee that the knapsack's entire capacity will be utilized. An example of ϕ_1 is illustrated in Figure 2 with $L = 1$ and $U = 5$. Suppose the input is composed of two groups of items. The first group of items has value-to-weight ratios of θ_1, and their total weight is 1. The second group of items has value-to-weight ratios of L, and their total weight is also 1. In this case, the value of OPT is θ_1 (shown in the green area). Although no item has a value-to-weight ratio higher than the threshold, KWA can still use up all the capacity due to the fill-up logic (Lines 6–7), and obtain a value of L (shown in the red area). Therefore, we have $\frac{\text{OPT}}{\text{KWA}} = \mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1 \approx 1.7$ for this special input, and this is actually the competitive ratio of KWA.

Through the above examples, we have illustrated at a high level how KWA can outperform the classical algorithm by adding the fill-up logic and redesigning the threshold function. In the next subsection, we will rigorously prove the competitive ratio of KWA and demonstrate that the threshold function ϕ_1 is optimal.

Competitive Analysis of KWA

There are two cases for accepting an item in KWA. One is from Line 9, where Algorithm 1 makes the decision according to ϕ_1. In this case, we say it makes an *active choice*. The other is from the fill-up logic Line 7, where Algorithm 1 tries to fill the knapsack as much as possible. In that case, we say it makes a *passive choice*. Note that once Algorithm 1 i makes a passive choice, all subsequent choices are passive. Define a_i as the binary indicator of whether Algorithm 1 makes an active choice on i:

1. Active choice: $a_i = 1$.
2. Otherwise (passive choice or reject): $a_i = 0$.

To continue our analysis, we need to introduce two observations about the algorithm. First, the value of KWA is at least L, because Algorithm 1 guarantees to use up all capacity, and the lowest value-to-weight ratio is L. Second, the value

of OPT is at least θ_1. Recall the example shown in Figure 2. OPT can achieve θ_1 when KWA remains at the lowest value.

In our analysis, for each item i we evaluate how much it improves the objective value of KWA and OPT as $\Delta\texttt{KWA}_i$ and $\Delta\texttt{OPT}_i$ respectively. Given the above two observations, w.l.o.g. we assume that $\frac{v_1}{w_1} \geq \theta_1$. Otherwise, we can set $\frac{v_1}{w_1} = \theta_1$ to improve OPT without increasing KWA. Define

$$\Delta\texttt{KWA}_i \triangleq \left(\frac{v_i}{w_i} - L\right) w_i x_i a_i, \text{ and} \tag{11}$$

$$\Delta\texttt{OPT}_i \triangleq \begin{cases} (\frac{v_i}{w_i} - \theta_1) w_i, \text{ when } i = 1, \\ \max_{j \leq i} \frac{v_j}{w_j} - \max_{k<i} \frac{v_k}{w_k}, \text{ otherwise.} \end{cases} \tag{12}$$

The definition of $\Delta\texttt{KWA}_i$ is as follows. If KWA accepts item i, its value increase $(\frac{v_i}{w_i} - L)w_i$ compared to the lowest possible value-to-weight ratio L. $\Delta\texttt{OPT}_i$ is an tight upper-bound of the OPT's gain, when a group of subsequent items have the same value-to-weight ratio $\frac{v_i}{w_i}$ with total weight 1. $\Delta\texttt{KWA}_i$ and $\Delta\texttt{OPT}_i$ provide the bound for $\texttt{KWA}(I)$ and $\texttt{OPT}(I)$:

$$\texttt{KWA}(I) \geq L + \sum_{i=1}^{n} \Delta\texttt{KWA}_i, \tag{13}$$

$$\texttt{OPT}(I) \leq \theta_1 + \sum_{i=1}^{n} \Delta\texttt{OPT}_i. \tag{14}$$

Next, we will prove an important lemma that clarifies a sufficient condition for an algorithm to be R^*-competitive.

Lemma 1. *The competitive ratio of Algorithm 1 is no more than R^* if*

1. *The solution is feasible.*
2. *Passive choice inequality (happens when all choices are passive):*
$$\frac{\phi_1(0)}{L} \leq R^*. \tag{15}$$
3. *Boundary condition:*
$$\phi_1(1) = U. \tag{16}$$
4. *Active choice inequality. For any feasible input $I \in \mathcal{I}$ and for each active choice $a_i = 1$, we have*
$$\frac{\Delta OPT_i}{\Delta KWA_i} \leq R^*. \tag{17}$$

The proof is shown in Appendix.

The four conditions in Lemma 1 basically state the following: For our online KWA, if it initially has the $\frac{\texttt{OPT}}{\texttt{KWA}}$ ratio R^* in the basic worst case (Condition 2), and if in each step, the increment of the ratio between $\Delta\texttt{OPT}$ and $\Delta\texttt{KWA}$ does not exceed R^* (Condition 4), then the competitive ratio is R^*. Condition 3 ensures the boundary condition that when capacity utilization reaches 1, $\Delta\texttt{OPT} = 0$ if $\frac{v_i}{w_i} = \phi(y)$ for each item. Otherwise, OPT can gain extra value while KWA cannot because of the capacity constraint. Condition 1 is clear because the solution must be feasible. With Lemma 1, we can prove that the competitive ratio of Algorithm 1 is $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ by verifying its four conditions.

Theorem 1 (Competitive Ratio Upper-Bound)**.** *The competitive ratio of Algorithm 1 is no more than $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ where $\mathcal{W}$ denotes the Lambert-$\mathcal{W}$ function. The proof is shown in Appendix.*

Now, we have shown that the competitive ratio of Algorithm 1 is no more than $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$. However, $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ is an upper bound. Whether this upper-bounded can be reached remains a question. Next, we will show the tightness of $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ using a worst-case example, and therefore prove that $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ is the competitive ratio.

Theorem 2 (Tightness)**.** *The upper bound $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ is tight (the upper bound can be reached), and therefore Algorithm 1 is $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$–competitive. The proof is shown in Appendix.*

Theorem 2 can be proved by expanding the example in Figure 2 to the general cases of U and L. Next, we will demonstrate the superiority of our algorithm. In Theorem 3, we prove that $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ is the lowest competitive ratio by constructing a special type of problem instance where the value-to-weight ratio increases continuously from θ_1 to U. Given such types of instances, no algorithm can achieve a competitive ratio strictly better than $\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$.

Theorem 3 (Optimality)**.** *$\mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ is the lowest possible competitive ratio in the exact volume problem. The proof is shown in Appendix.*

These specially constructed instances explain why including the total weight of the remaining items in the threshold function cannot improve the competitive ratio. First, an intuition suggests that for the item sequence we designed (see the definitions of Appendix (49–51)), adding items with any total weight and a value-to-weight ratio of L at the end will not improve KWA's result, as these are the lowest-value items. Similarly, it will not affect OPT's result. Therefore, KWA's optimal decision should not change based on variations in the total weight of the remaining items. Secondly, our algorithm has already achieved the best balance for this type of input, and any changes would worsen its worst-case performance. Details can be found in (54), (57), and (60) in the Appendix: We have a series of k inputs, and if other algorithms aim to outperform KWA in the worst-case performance on first $k-1$ inputs, they must allocate more capacity than KWA to items with a value-to-weight ratio below U. However, this would cause these algorithms to perform worse than KWA on the kth input. Therefore, KWA has achieved the optimal competitive ratio.

Predicted Weight Algorithm (PWA)

In many real-world scenarios, it is hard to obtain exact information about the total weight. Instead, we can obtain predictions using machine learning methods, and use that to augment our online algorithm. However, given that the quality of predictions is not guaranteed, we need to optimize two measures simultaneously: (1) consistency—the competitive ratio when the prediction is accurate; and (2) robustness—the competitive ratio when the prediction is adversarial. The idea of PWA is: Since KWA achieves the optimal

Algorithm 2: Predicted Weight Algorithm (PWA)

1: **Input:** $I = \{(w_i, v_i),\ \forall i \in [n], L, U\}, \hat{W}$.
2: **Output:** $x_i,\ \forall i \in [n]$.
3: run $\texttt{KWA}(I, \hat{W})$ with λ capacity, and run $\texttt{OKA}(I)$ with $1-\lambda$ capacity.

competitive ratio when the prediction is accurate, and the classical online knapsack algorithm (OKA) achieves the optimal competitive ratio when the total weight information is unavailable (equivalent to adversarial total weight information), we can use $\lambda \in [0, 1]$ of the capacity to run KWA and run OKA with the remaining capacity. Increasing λ will optimize consistency when we are more confident in the prediction but at the expense of robustness, and vice versa.

We run $\texttt{KWA}(I, \hat{W})$ with λ capacity in the following way. First, we virtually run Algorithm 1 exactly as it states as if the total capacity is 1. Then, whenever the virtual $\texttt{KWA}(I, \hat{W})$ decides to accept item i, PWA uses λw_i capacity to accept a λ fraction of item i. In this way, we construct an algorithm whose decisions are λ times that of KWA, and the total capacity used is at most λ. We also treat the algorithm OKA using $(1-\lambda)$ of the capacity in the same manner. However, in a real system, if fractional acceptance is not allowed (e.g., customer i must obtain w_i or 0 resources from the CSP), then we can create a randomized algorithm which has the same behavior in expectation: Initially, the randomized algorithm decides to run $\texttt{KWA}(I, \hat{W})$ with probability λ and run $\texttt{OKA}(I)$ with probability $1-\lambda$.

Analysis of PWA

In this subsection, we analysis the performance of the learning-augmented PWA. In order to clearly distinguish between three algorithms that will be mentioned, we denote the competitive ratio of KWA as $c_1 = \mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ and the competitive ratio of OKA as $c_2 = \ln(\frac{U}{L}) + 1$. $\texttt{PWA}(I, \hat{W}) = \lambda\texttt{KWA}(I, \hat{W}) + (1-\lambda)\texttt{OKA}(I),\ \forall I \in \mathcal{I}$. The optimal offline algorithm is still denoted by OPT.

First, we show the robustness of PWA. The proof is straightforward: When the prediction error is large, the value obtained by KWA could be 0, because KWA believes that there will be sufficient items in the future, but in fact it may not exist. Thus, the competitive ratio of PWA is related to c_2 and the parameter λ.

Theorem 4 (Robustness)**.** *Algorithm 2 is* $\frac{c_2}{1-\lambda} = \frac{\ln(\frac{U}{L})+1}{1-\lambda}$ *robust. The proof is shown in Appendix.*

Next, we prove the consistency of PWA. Please note that when proving the consistency, $\hat{W} = W$, so we can bound $\texttt{KWA}(I, \hat{W})$ in terms of its competitive ratio c_1.

Theorem 5 (Consistency)**.** *Algorithm 2 is* $\frac{c_1 c_2}{c_2\lambda + c_1(1-\lambda)}$ *consistent. The proof is shown in Appendix.*

Let $\eta = |\hat{W} - W|$ denotes the error of the prediction. Next, we prove that the performance of PWA degrades gracefully as the prediction error grows by bounding the competitive ratio of PWA when $\eta \geq 0$. Let $c_1(\eta)$ donate the competitive ratio of KWA given $\eta \geq 0$, and $c_1 = c_1(0) = \mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1$ in short. We have the following lemma to show how the competitive ratio of KWA degrades as η increases.

Lemma 2. *The competitive ratio of* KWA *given* $\eta \geq 0$ *is*

$$c_1(\eta) = \max\left(c_1 + \eta\frac{U-\theta_1}{L}, \frac{c_1}{1-\eta}\right). \tag{18}$$

The proof is shown in Appendix.

No matter how η changes, the competitive ratio of OKA remains c_2. We have the following theorem to show the competitive ratio of PWA when $\eta \geq 0$. The result can be obtained using Lemma 2 and the same derivation in Theorem 5.

Theorem 6 (Graceful Degradation)**.** *The competitive ratio of PWA given* $\eta \geq 0$ *is*

$$\frac{c_1(\eta)c_2}{\lambda c_2 + (1-\lambda)c_1(\eta)}. \tag{19}$$

The proof is shown in Appendix.

Limited Weight Algorithm (LWA)

In this section, we discuss an alternative way of utilizing the total weight information W to improve the competitive ratio beyond $\ln(\frac{U}{L}) + 1$. The idea is: If the total weight is small, say $W \leq 1$, the online algorithm can simply accept all items, which is also the optimal offline solution. In this case, the competitive ratio must be 1. As W increases, it becomes increasingly difficult to utilize it to improve the performance of our algorithms in this way. Eventually, the competitive ratio of the algorithm will approach $\ln(\frac{U}{L}) + 1$ as if W were unknown. However, it does not prevent us from exploring when and how W can be utilized and at what rate the competitive ratio degrades as W grows.

As in the previous motivating example, when $W \leq 1$, the online algorithm can set the value-to-weight threshold to L, making the competitive ratio 1. When $1 < W < 2$, if the online algorithm uses up all capacity, there must be "item overlapping" due to the pigeonhole principle, which means that for any $I \in \mathcal{I}_W$, there must be at least one item i that is accepted by both OPT and ALG. An example is shown in Figure 3. The online algorithm can then utilize this property to improve the competitive ratio because it adds extra constraints to the worst-case scenario. When $W \geq 2$, the "item overlapping" does not necessarily exist, so the competitive ratio cannot be improved in the same way as in the first two cases.

LWA is presented as Algorithm 3. First, it initializes the capacity utilization $y_0 = 0$ (Line 3). Upon the arrival of item i, its weight w_i and value v_i are revealed (Line 5). If the remaining capacity can hold item i, and if item i's value is higher than the threshold (Line 6), then we accept item i and update the capacity utilization (Line 7). Otherwise, we reject this item, and the capacity utilization remains the same (Line 9).

LWA is also a threshold-based algorithm. The performance of this algorithm is determined by the design of the threshold function. Similar to the threshold function (10) for the

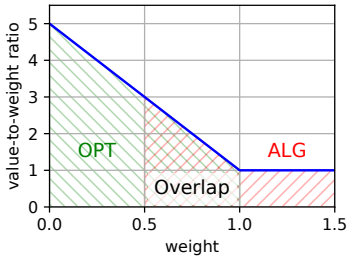


Figure 3: An example of item overlapping with $W = 1.5$. Due to the pigeonhole principle, there must be an overlap of width 0.5 on the horizontal axis between OPT and ALG.

optimal online knapsack OKA, our new threshold function is segmented and can be viewed as introducing a parameter θ_2 that controls the segment lengths and the growth rate of the exponential function. Our threshold function is defined as follows:

$$\phi_2(y) = \begin{cases} L \ \forall y \in [0, \theta_2), \\ \frac{L}{e} e^{\frac{1}{\theta_2} y} \ \forall y \in [\theta_2, 1], \end{cases} \tag{20}$$

where the parameter θ_2 is determined by the following. When $W \leq 1$

$$\theta_2 = 1. \tag{21}$$

When $1 < W < 2$, θ_2 is the solution of the following equation which can be solved numerically:

$$\int_{W-1}^{1} \phi_2(\delta) d\delta + (W-1)U - Le^{\frac{1}{\theta_2}-1} = 0. \tag{22}$$

When $W > 2$

$$\theta_2 = \frac{1}{\ln(\frac{U}{L}) + 1}. \tag{23}$$

It is worth noting that the solution of formula (22) is 1 when $W = 1$ and $\frac{1}{\ln(\frac{U}{L})+1}$ when $W = 2$. We can consider (21) and (23) as the boundary of (22).

Competitive Analysis of LWA

In this subsection, we analyze the performance of Algorithm 3 under three conditions: (1) $W \leq 1$, (2) $W \geq 2$, and (3) $1 < W < 2$. Recall that W is known to the online algorithm. It is helpful to imagine an adversary feeding our algorithm the worst possible sequence (w_i, v_i) for all $i \in [n]$. First, when $W \leq 1$, the problem is trivial because the capacity allows us to accept all items, and this strategy is optimal.

Theorem 7 (Competitive Ratio Case 1)**.** *When $W \leq 1$, the competitive ratio of Algorithm 3 is 1. The proof is shown in Appendix.*

Next, when $W \geq 2$, $\theta_2 = \frac{1}{\ln(\frac{U}{L})+1}$. At this point we can no longer improve the competitive ratio through item overlapping, so we derive the same result as the threshold function for OKA.

Algorithm 3: Limited Weight Algorithm (LWA)

Input: $I = \{(w_i, v_i),\ \forall i \in [n], L, U\}$, $W = \sum_{i=1}^{n} w_i$.
Output: $x_i,\ \forall i \in [n]$.
$y_0 = 0$;
for $i \in [n]$ **do**
  item i arrives, and (w_i, v_i) is revealed;
  if $y_{i-1} + w_i \leq 1$ and $v_i \geq \int_{y_{i-1}}^{y_{i-1}+w_i} \phi_2(\delta) d\delta$ **then**
    accept item i, $x_i = 1$, $y_i = y_{i-1} + w_i$;
  else
    reject item i, $x_i = 0$, $y_i = y_{i-1}$.
  end if
end for

Theorem 8 (Competitive Ratio Case 2)**.** *When $W \geq 2$, the competitive ratio of Algorithm 3 is $\ln(\frac{U}{L}) + 1$. The proof is shown in Appendix.*

When $1 < W < 2$, this is the most interesting case. At this point, our algorithm can obtain a better competitive ratio than $\ln(\frac{U}{L}) + 1$. Unfortunately, when $W \in (1, 2)$, the competitive ratio does not have a close form solution. In this theorem, we prove that competitive ratio is less than $\ln(\frac{U}{L}) + 1$.

Theorem 9 (Competitive Ratio Case 3)**.** *When $1 < W < 2$, the competitive ratio of Algorithm 3 is θ_2, and it is in $(1, \ln(\frac{U}{L}) + 1)$. The proof is shown in Appendix.*

In addition to the theoretical analysis, we also provide numerical verification for the performance of KWA and the LWA. These numerical results offer further validation of the competitive ratios derived. The detailed results and corresponding discussions are presented in Figures 4–5 in the Appendix.

Conclusion

In this paper, we explored innovative approaches to improving the competitive ratio for the online knapsack problem by incorporating total weight information, whether known exactly or predicted. The Known Weight Algorithm (KWA) leverages exact total weight information to achieve a competitive ratio of $\mathcal{W}\left(\frac{1}{e}\frac{U-L}{L}\right) + 1$, which is both optimal and tight, surpassing the traditional $\ln\frac{U}{L} + 1$ ratio. Next, we extended this concept through the learning-augmented Predicted Weight Algorithm (PWA), where total weight predictions are used despite potential inaccuracies. PWA balances consistency and robustness by tuning the algorithm according to prediction accuracy, and its competitive ratio degrades gracefully as the prediction error grows. Additionally, we introduced the Limited Volume Algorithm (LWA) for cases where the total weight is known and less than twice the capacity, further improving the competitive ratio in these scenarios. Future research could apply these techniques to other variants of the online knapsack problem, such as multi-dimensional cases, concave valuation functions, and knapsack problems with departures, to enhance their competitive ratios.

References

Amiri, M.; and Mohammad-Khanli, L. 2017. Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications*, 82: 93–113.

Anguita, J. G. M.; and Olariaga, O. D. 2023. Air cargo transport demand forecasting using ConvLSTM2D, an artificial neural network architecture approach. *Case Studies on Transport Policy*, 12: 101009.

Balseiro, S.; Kroer, C.; and Kumar, R. 2023. Single-Leg Revenue Management with Advice. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, 207. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701047.

Cao, Y.; Sun, B.; and Tsang, D. H. 2020. Optimal online algorithms for one-way trading and online knapsack problems: A unified competitive analysis. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 1064–1069. IEEE.

Corless, R. M.; Gonnet, G. H.; Hare, D. E.; Jeffrey, D. J.; and Knuth, D. E. 1996. On the Lambert W function. *Advances in Computational mathematics*, 5: 329–359.

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

Filom, S.; Amiri, A. M.; and Razavi, S. 2022. Applications of machine learning methods in port operations–A systematic literature review. *Transportation Research Part E: Logistics and Transportation Review*, 161: 102722.

Han, X.; Kawase, Y.; and Makino, K. 2014. Online unweighted knapsack problem with removal cost. *Algorithmica*, 70(1): 76–91.

Hwang, D.; Jaillet, P.; and Manshadi, V. 2021. Online resource allocation under partially predictable demand. *Operations Research*, 69(3): 895–915.

Im, S.; Kumar, R.; Montazer Qaem, M.; and Purohit, M. 2021. Online knapsack with frequency predictions. *Advances in Neural Information Processing Systems*, 34: 2733–2743.

Jooken, J.; Leyman, P.; and De Causmaecker, P. 2022. A new class of hard problem instances for the 0–1 knapsack problem. *European Journal of Operational Research*, 301(3): 841–854.

Lin, Q.; Mo, Y.; Su, J.; and Chen, M. 2022. Competitive Online Optimization with Multiple Inventories: A Divide-and-Conquer Approach. 83 – 84. Cited by: 0.

Lin, Q.; Yi, H.; Pang, J.; Chen, M.; Wierman, A.; Honig, M.; and Xiao, Y. 2019. Competitive online optimization under inventory constraints. 35 – 36. Cited by: 2; All Open Access, Bronze Open Access, Green Open Access.

Lykouris, T.; and Vassilvitskii, S. 2021. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4): 1–25.

Purohit, M.; Svitkina, Z.; and Kumar, R. 2018. Improving online algorithms via ML predictions. *Advances in Neural Information Processing Systems*, 31.

Schroeder, P.; Dochow, R.; and Schmidt, G. 2018. Optimal solutions for the online time series search and one-way trading problem with interrelated prices and a profit function. *Computers & Industrial Engineering*, 119: 465–471.

Sun, B.; Yang, L.; Hajiesmaili, M.; Wierman, A.; Lui, J. C.; Towsley, D.; and Tsang, D. H. 2022. The online knapsack problem with departures. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(3): 1–32.

Yang, L.; Zeynali, A.; Hajiesmaili, M. H.; Sitaraman, R. K.; and Towsley, D. 2021. Competitive algorithms for online multidimensional knapsack problems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(3): 1–30.

Zeynali, A.; Sun, B.; Hajiesmaili, M.; and Wierman, A. 2021. Data-driven competitive algorithms for online knapsack and set cover. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10833–10841.

Zhang, Z.; Li, Z.; and Wu, C. 2017. Optimal posted prices for online cloud resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1): 1–26.

Zhu, H.; Luo, J.; and Deng, H. 2020. Optimizing the Procurement of IaaS Reservation Contracts via Workload Predicting and Integer Programming. *Mathematical Problems in Engineering*, 2020.

Reproducibility Checklist

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes)
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes)
- Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes)
- Does this paper make theoretical contributions? (yes)

If yes, please complete the list below:

- All assumptions and restrictions are stated clearly and formally. (yes)
- All novel claims are stated formally (e.g., in theorem statements). (yes)
- Proofs of all novel claims are included. (yes)
- Proof sketches or intuitions are given for complex and/or novel results. (yes)
- Appropriate citations to theoretical tools used are given. (yes)
- All theoretical claims are demonstrated empirically to hold. (yes)
- All experimental code used to eliminate or disprove claims is included. (yes)

Does this paper rely on one or more datasets? (no)

This is a theoretical work, we only focus on the validation of theoretical results. The data used in the validation experiment is randomly generated, and the methods and code for random generation are provided in Appendix and supplementary materials.

Does this paper include computational experiments? (yes)

If yes, please complete the list below:

- Any code required for pre-processing data is included in the appendix. (yes)
- All source code required for conducting and analyzing the experiments is included in a code appendix. (yes)
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes)
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes)
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes)
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes)
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes)
- This paper states the number of algorithm runs used to compute each reported result. (yes)
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (yes)
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes)
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes)
- This paper states the number and range of values tried per (hyper-)parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes)

Complete Proof for KWA

Proof of Lemma 1

Proof. Let A denote the set of items that are accepted by active choices of KWA, and $\bar{A}$ be the set of items that KWA makes passive choice or reject. Formally, we have

$$A = \{i | a_i = 1\}, \text{ and} \tag{24}$$

$$\bar{A} = \{i | a_i = 0\}. \tag{25}$$

From (13–14), we have

$$\frac{\texttt{OPT}(I)}{\texttt{KWA}(I)} \leq \frac{\theta_1 + \sum_{i\in A} \Delta\texttt{OPT}_i + \sum_{j\in \bar{A}} \Delta\texttt{OPT}_j}{L + \sum_{i\in A} \Delta\texttt{KWA}_i + \sum_{j\in \bar{A}} \Delta\texttt{KWA}_i}. \tag{26}$$

For any $a_i = 0$ (passive choice or reject), we have

$$\Delta\texttt{OPT}_i = \Delta\texttt{KWA}_i = 0, \ \forall i \in \bar{A}. \tag{27}$$

This is because: (1) w.l.o.g. the value-to-weight ratio of all passive choice must be L, and $L < \theta_1$, and (2) from the boundary condition $\phi_1(1) = U$, any value-to-weight ratio greater than θ_1 and the historically highest value-to-weight ratio will be accepted by KWA, i.e. if $\frac{v_i}{w_i} \geq \theta_1$ and $\frac{v_i}{w_i} > \max_{j<i} \frac{v_j}{w_j}$, then KWA will accept item i. Thus, whenever $\Delta\texttt{OPT}_i > 0$, $a_i = 1$. The boundary condition $\phi_1(1) = U$ is needed to prevent the algorithm from running out of capacity when there is an new high $\frac{v_i}{w_i}$. So we have

$$\frac{\texttt{OPT}(I)}{\texttt{KWA}(I)} \leq \frac{\theta_1 + \sum_{i\in A} \Delta\texttt{OPT}_i}{L + \sum_{i\in A} \Delta\texttt{KWA}_i}, \tag{28}$$

$$\leq \max\left(\frac{\theta_1}{L}, \max_{i\in A} \frac{\Delta\texttt{OPT}_i}{\Delta\texttt{KWA}_i}\right) \tag{29}$$

$$\leq R^*. \tag{30}$$

□

Proof of Theorem 1

Proof. We prove this theorem by checking the 4 conditions listed in Lemma 1.

Condition (1). The solution is always feasible because Algorithm 1 never allocates resources more than 1 (see Lines 6 and 8).

Condition (2).

$$\frac{\phi_1(0)}{L} = \frac{\theta_1}{L} = \mathcal{W}\left(\frac{1}{e}\frac{U-L}{L}\right) + 1. \tag{31}$$

Condition (3).

$$\phi_1(1) = L + (\theta_1 - L)e^{\frac{\theta_1}{L}} \tag{32}$$

$$= L + L\mathcal{W}\left(\frac{1}{e}\frac{U-L}{L}\right) e^{\mathcal{W}(\frac{1}{e}\frac{U-L}{L})+1} \tag{33}$$

$$= L + L\left(\frac{1}{e}\frac{U-L}{L}\right) e \tag{34}$$

$$= U. \tag{35}$$

Condition (4).

$$\frac{\Delta\texttt{OPT}_i}{\Delta\texttt{KWA}_i} \leq \frac{\max_{j\leq i} \frac{v_j}{w_j} - \max_{k<i} \frac{v_k}{w_k}}{(\frac{v_i}{w_i} - L)w_i x_i a_i} \tag{36}$$

$$\leq \frac{\phi_1(y_i) - \phi_1(y_{i-1})}{(\phi_1(y_i) - L)w_i} \tag{37}$$

$$\leq \frac{\phi_1(y_{i-1} + w_i) - \phi_1(y_{i-1})}{(\phi_1(y_i) - L)w_i}. \tag{38}$$

We assume that w_i is an infinitesimal, so we have

$$\frac{\Delta\texttt{OPT}_i}{\Delta\texttt{KWA}_i} \leq \lim_{w_i\to 0} \frac{\phi_1(y_{i-1} + w_i) - \phi_1(y_{i-1})}{w_i(\phi_1(y_{i-1} + w_i) - L)} \tag{39}$$

$$= \frac{\phi'(y_{i-1})}{\phi_1(y_{i-1}) - L} \tag{40}$$

$$= \mathcal{W}(\frac{1}{e}\frac{U-L}{L}) + 1. \tag{41}$$

□

Proof of Theorem 2

Proof. Let n be a large number. When

$$w_1 = w_2 = \ldots = w_{2n} = \frac{1}{n}, \tag{42}$$

$$v_1 = v_2 = \ldots = v_n = \frac{\theta_1}{n}, \tag{43}$$

and

$$v_{n+1} = v_{n+2} = \ldots = v_{2n} = \frac{L}{n}. \tag{44}$$

We have

$$\texttt{OPT} = \theta_1 = L\left(\mathcal{W}\left(\frac{1}{e}\frac{U-L}{L}\right) + 1\right), \tag{45}$$

$$\texttt{KWA} = L. \tag{46}$$

The competitive ratio $\mathcal{W}\left(\frac{1}{e}\frac{U-L}{L}\right) + 1$ is reached. □

Proof of Theorem 3

Proof. We prove this lower bound of by contradiction. Let KWA$'$ denote Algorithm 1 using the threshold function ϕ', which achieves competitive ratio $R' < \mathcal{W}\left(\frac{1}{e}\frac{U-L}{L}\right) + 1$.

Let k be a large integer, and $w = \frac{1}{k}$ which is an infinitesimal. $\forall i \in [k]$ define an adversarial active value sequence $\bar{\boldsymbol{\alpha}}^i$ as

$$\bar{\boldsymbol{\alpha}}^i = (\bar{\alpha}_1 = \phi_1(w)w, \ \bar{\alpha}_2 = \phi_1(2w)w, \ \ldots, \bar{\alpha}_i = \phi_1(iw)w). \tag{47}$$

$\forall i \in [k]$ define an adversarial passive sequence $\bar{\boldsymbol{\beta}}^i$ as

$$\bar{\boldsymbol{\beta}}^i = (\bar{\beta}^i_1 = \bar{\alpha}_i, \bar{\beta}^i_2 = \bar{\alpha}_i, \bar{\beta}^i_3 = \bar{\alpha}_i, \ldots, \bar{\beta}^i_k = \bar{\alpha}_i, \tag{48}$$
$$\bar{\beta}^i_{k+1} = Lw, \bar{\beta}^i_{k+2} = Lw, \bar{\beta}^i_{k+3} = Lw, \ldots, \bar{\beta}^i_{2k} = Lw).$$

We use the symbol $\oplus$ to represent the concatenation of two sequences. $\forall i \in [k]$ and given $[L, U]$, define item value sequence $\boldsymbol{V}_i$:

$$\boldsymbol{V}_1 = \bar{\boldsymbol{\alpha}}^1 \oplus \bar{\boldsymbol{\beta}}^1, \tag{49}$$

$$\boldsymbol{V}_2 = \boldsymbol{V}_1 \oplus \bar{\boldsymbol{\alpha}}^2 \oplus \bar{\boldsymbol{\beta}}^2, \tag{50}$$

$$\dots,$$

$$\boldsymbol{V}_k = \boldsymbol{V}_{k-1} \oplus \bar{\boldsymbol{\alpha}}^k \oplus \bar{\boldsymbol{\beta}}^k, \tag{51}$$

$\forall i \in [k]$ define an adversarial instance I_i

$$I_i = \{(w, v_j), \forall v_j \in \boldsymbol{V}_i, L, U\}. \tag{52}$$

Please note that adding items with any total weight and a value-to-weight ratio of L at the end will not improve KWA's result, because (1) the total weight of I_i $\forall i \in [k]$ is greater than 1, and (2) these are the lowest-value items. Similarly, it will not affect OPT's result.

Let z_i and z_i' denote the number of active choices made by KWA and KWA$'$ given the adversarial instance I_i. First, we have

$$z_i = i, \forall i \in [k]. \tag{53}$$

z_1' is the number of active choices that accepting $\bar{\alpha}_1$, and $z_{i+1}' - z_i'$ is the number of active choices that accepting the exchange rate $\bar{\alpha}_i$, $\forall i \in [k-1]$.

Since the competitive ratio of $R' < W(\frac{1}{e}\frac{U-L}{L}) + 1$, for I_i, we must have

$$z_1' > z_1 = 1. \tag{54}$$

Otherwise, KWA$' \leq$ KWA, which contradicts to $R' < W(\frac{1}{e}\frac{U-L}{L}) + 1$. This means that in order to perform better than KWA, KWA$'$ has to allocate more capacity than KWA on item with value-to-weight ratio $\phi_1(w)$. Let j denote the smallest index that $z_j' - z_j < 0$. Then, we can construct another algorithm $\overline{\text{KWA}}$ where

$$\bar{z}_i = z_i, \ \forall i \in [j-1], \text{and } \bar{z}_j = z_j' - z_{j-1}. \tag{55}$$

Since $0 < \bar{\alpha}_1 < \bar{\alpha}_2 < \ldots < \bar{\alpha}_k$, and $z_j' - z_j < 0$, we have

$$\text{KWA}' \leq \overline{\text{KWA}} < \text{KWA}. \tag{56}$$

Then it contradicts to $R' < W(\frac{1}{e}\frac{U-L}{L}) + 1$. So we have

$$z_i' \geq z_i, \ \forall i \in [k]. \tag{57}$$

It means that if KWA$'$ out-performs KWA, KWA$'$ has to allocate more capacity than KWA on item with value-to-weight ratio smaller than U. However, please note that KWA$'$ can accept at most k exchange rate, because of the we only have a unit capacity in total, so

$$z_i' \leq k, \ \forall i \in [k], \text{ then} \tag{58}$$

$$z_k' - z_k = z_k' - k \leq 0. \tag{59}$$

Combine with (57),

$$z_k' - z_k = 0, \text{ which implies} \tag{60}$$

$$\text{KWA}(I_k) - \text{KWA}'(I_k) = \bar{\alpha}_1(z_1 - z_1') + \tag{61}$$

$$\sum_{i=2}^{k} \bar{\alpha}_i[(z_i - z_{i-1}) - (z_i' - z_{i-1}')]$$

$$\geq (\bar{\alpha}_k - \bar{\alpha}_1)(z_1' - z_1) \tag{62}$$

$$> 0. \tag{63}$$

The extra capacity that KWA$'$ allocates to items with a value-to-weight ratio lower than U is equal to the extra capacity that KWA (compared to KWA$'$) allocates to items with a value-to-weight ratio equal to U. Therefore, KWA outperforms KWA$'$ on I_k. It contradicts to $R' < W(\frac{1}{e}\frac{U-L}{L}) + 1$. □

Complete Proof for PWA

Proof of Theorem 4

Proof. In the worst case, KWA allocates no capacity.

$$\sup_{I \in \mathcal{I}} \frac{\text{OPT}(I)}{\text{PWA}(I)} = \sup_{I \in \mathcal{I}} \frac{\text{OPT}(I)}{\lambda \text{KWA}(I) + (1-\lambda)\text{OKA}(I)} \tag{64}$$

$$= \sup_{I \in \mathcal{I}} \frac{\text{OPT}(I)}{(1-\lambda)\text{OKA}(I)} \tag{65}$$

$$\leq \frac{c_2}{1-\lambda}. \tag{66}$$

Thus, Algorithm 2 is $\frac{c_2}{1-\lambda}$ robust. □

Proof of Theorem 5

Proof. Please note that when proving the consistency, we assume $\hat{W} = W$. Directly from the definition of the competitive ratio, we have

$$\lambda \text{KWA}(I, W) \geq \frac{\lambda \text{OPT}(I)}{c_1}, \text{and} \tag{67}$$

$$(1-\lambda)\text{OKA} \geq \frac{(1-\lambda)\text{OPT}}{c_2}, \ \forall I \in \mathcal{I}. \tag{68}$$

$$\sup_{I \in \mathcal{I}} \frac{\text{OPT}(I)}{\lambda \text{KWA}(I, W) + (1-\lambda)\text{OKA}(I)} \leq \frac{1}{\frac{\lambda}{c_1} + \frac{(1-\lambda)}{c_2}} \tag{69}$$

$$= \frac{c_1 c_2}{\lambda c_2 + (1-\lambda) c_1}. \tag{70}$$

□

Proof of Lemma 2

Proof. $\forall I \in \mathcal{I}$ and $\forall \hat{W} > 0$, let $y_{I,\hat{W}}$ denote the total capacity utilization by active choices of KWA given instance I and prediction $\hat{W}$.

Let A denote the set of items that KWA makes active choices, and $\bar{A}$ be the set of items that KWA makes passive choices or reject. Formally, we have

$$A = \{i | a_i = 1\}, \text{ and} \tag{71}$$

$$\bar{A} = \{i | a_i = 0\}. \tag{72}$$

When $W - \hat{W} \geq 0$, the actual total weight is later than the predicted total weight. In this case, KWA can still allocates all capacity, but after $\hat{W}$ amount of weight arrival, KWA has no capacity left. Then, setting the value-to-weight ratio to U is

advantageous to OPT. Then we have

$$\sup_{I\in\mathcal{I}} \frac{\texttt{OPT}(I)}{\texttt{KWA}(I,\hat{W})} \le \sup_{I\in\mathcal{I}} \frac{(1-\eta)[\theta_1+\sum_{i\in A}\Delta\texttt{OPT}_i]+\eta U}{L+\sum_{i\in A}\Delta\texttt{KWA}_i} \tag{73}$$

$$\le \sup_{I\in\mathcal{I}} \frac{\theta_1+\sum_{i\in A}\Delta\texttt{OPT}_i+\eta(U-\phi(y_{I,\hat{W}}))}{L+\sum_{i\in A}\Delta\texttt{KWA}_i}. \tag{74}$$

Let $I'_{\hat{W}}$ be the problem instance that contains only the first $\hat{W}$ weight of I. We have

$$L+\sum_{i\in A}\Delta\texttt{KWA}_i = \texttt{KWA}(I'_{\hat{W}}), \text{ and} \tag{75}$$

$$\theta_1+\sum_{i\in A}\Delta\texttt{OPT}_i = \texttt{OPT}(I'_{\hat{W}}). \tag{76}$$

Take them into (74), we can obtain

$$\sup_{I\in\mathcal{I}} \frac{\texttt{OPT}(I)}{\texttt{KWA}(I,\hat{W})} \le \frac{c_1\texttt{KWA}(I'_{\hat{W}})+\eta(U-\phi(y_{I,m}))}{\texttt{KWA}(I'_{\hat{W}})} \tag{77}$$

$$\le c_1+\eta\frac{U-\theta_1}{L}. \tag{78}$$

(78) because $\phi(y_{I,\hat{W}}) \ge \phi(0)$ and $\texttt{KWA}(I'_{\hat{W}}) \ge L$. The inequality is tight. This ratio is reached when k is a large integer and $w=\frac{1}{k}$, and $I'_{\hat{W}}$ contains value ratio $v'_1=v'_2=\ldots=v'_k=\phi(w)w$, $v'_{k+1}=v'_{k+2}=\ldots=v'_{2k}=Lw$, and weights $w_1=w_2=\ldots=w_{2k}=w$.

When $W-\hat{W}\le 0$, the actual total weight is smaller than the predicted total weight. Although it does not influence active choices, but the capacity may not be used up in the end. Recall that for the passive choice, we always assume that KWA can only gets the rate L, so now $\forall\hat{W}>0$ we have

$$\sup_{I\in\mathcal{I}} \frac{\texttt{OPT}(I)}{\texttt{KWA}(I,\hat{W})+\eta L} \le c_1. \tag{79}$$

So we have

$$\sup_{I\in\mathcal{I}} \frac{\texttt{OPT}(I)}{\texttt{KWA}(I,\hat{W})} \le \sup_{I\in\mathcal{I}} c_1+\frac{c_1\eta L}{\texttt{KWA}(I,\hat{W})} \tag{80}$$

$$\le c_1+\frac{c_1\eta L}{L(1-\eta)} \tag{81}$$

$$= \frac{c_1}{1-\eta}. \tag{82}$$

Since the competitive ratio considers the worst-case, we have

$$c_1(\eta) = \max\left(c_1+\eta\frac{U-\theta_1}{L}, \frac{c_1}{1-\eta}\right). \tag{83}$$

□

Proof of Theorem 6

Proof. No matter how η changes, the competitive ratio of OKA remains c_2. The result can be obtained using Lemma 2 and the same derivation in Theorem 5. □

Complete Proof of LWA

Proof of Theorem 7

Proof. When $W\le 1$, the problem is trivial, because the capacity allows us to hold all items, and this strategy is optimal. Note that we have $\theta_2=L$ when $W\le 1$, which means that by Algorithm 3 accepts all items. It behaves the same as the offline optimal algorithm, so the competitive ratio is 1. □

Proof of Theorem 8

Proof. When $W\ge 2$, Algorithm 3 works as same as OKA. The competitive ratio is known as $\ln\frac{U}{L}+1$. □

Proof of Theorem 9

Proof. Let y denote the final capacity utilization of LWA. Please note that when $y<\min(W,\theta_2)$, $\frac{\texttt{OPT}}{\texttt{LWA}}=1$, so we consider $y\ge\min(W,\theta_2)$ in the following. When $y<1$, we have

$$\frac{\texttt{OPT}}{\texttt{LWA}} \le \frac{\phi_2(y)}{\int_0^y\phi_2(\delta)d\delta} \tag{84}$$

$$= \frac{\frac{L}{e}e^{\frac{1}{\theta_2}y}}{\theta_2 L+\int_{\theta_2}^y\frac{L}{e}e^{\frac{1}{\theta_2}\delta}d\delta} \tag{85}$$

$$= \frac{1}{\theta_2}. \tag{86}$$

When $y=1$, for the sake of notational simplicity, we assume that LWA allocates to first k items with total weight W, and $\frac{v_i}{w_i}\le\frac{v_j}{w_j}\ \forall i\le j$. So we have

$$\frac{\texttt{OPT}}{\texttt{LWA}} = \frac{\sum_{i=n-k+1}^n v_i}{\sum_{i=1}^k v_i}. \tag{87}$$

Since $1<W<2$

$$\frac{\texttt{OPT}}{\texttt{LWA}} \le \frac{\int_{W-1}^1\phi_2(\delta)d\delta+(W-1)U}{\int_0^{W-1}\phi_2(\delta)d\delta+\int_{W-1}^1\phi_2(\delta)d\delta} \tag{88}$$

$$= \frac{\int_{W-1}^1\phi_2(\delta)d\delta+(W-1)U}{L\theta_2 e^{\frac{1}{\theta_2}-1}}. \tag{89}$$

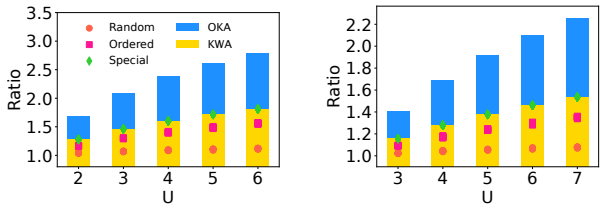
Because of (22), we have

$$\frac{\texttt{OPT}}{\texttt{LWA}} \le \frac{1}{\theta_2}. \tag{90}$$

□

Numerical Results

In this section, all experiments are conducted on a system equipped with an Intel Core i9-13900K CPU, and 16GB of memory. No GPU is used. The software environment for the experiments is as follows: Windows 10 operating system, Python 3.11.9, with dependencies on the numpy 1.25.1 and scipy 1.11.1 packages.



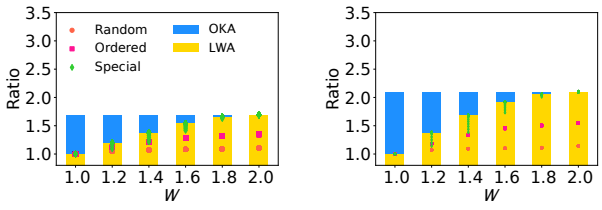
(a) $L = 1$ (b) $L = 2$

(c) $L = 3$ (d) $L = 4$

(e) $L = 5$ (f) $L = 6$

(g) $L = 7$ (h) $L = 8$

Figure 4: Theoretical result Verification for KWA



(a) $U = 2$ (b) $U = 3$

(c) $U = 4$ (d) $U = 5$

(e) $U = 6$ (f) $U = 7$

(g) $U = 8$ (h) $U = 9$

Figure 5: Theoretical result Verification for LWA

Numerical Verification for KWA

Figure 4 shows our numerical verification of KWA. We adopt $W = 3$, $L = 1, 2, \ldots, 8$. For each sub-figure, we adopt and $U = L + 1, L + 2, \ldots, L + 5$. In the bar graph, the blue bar represents the competitive ratio of the OKA algorithm and the yellow bar represents the theoretical competitive ratio of the KWA algorithm. The points contained in the bar represent the OPT/KWA ratio for each experiment. Each bar contains three sets of experiments with uniform distribution (random), uniform distribution + sort (ordered), and worst case input (special). The first two inputs were performed with 80 repetitions per group.

From the experimental results, we can see that all the OPT/KWA ratios are within the theoretical values, and the worst case can reach the theoretical values. Ordered is always worse than random, which is due to the lower threshold of our algorithm when the resource utilization level is low. With ordered input, the algorithm accepts more lower intrinsic values at the very beginning.

Numerical Verification for LWA

Figure 5 shows the verification result of LWA. We adopt W from 1 to 2 with a step size of 0.2 and show the experimental ratio $\frac{\text{OPT}}{\text{LWA}}$. We assume $L = 1$ and $U = 5$. For each W, we generate 80 inputs that contain prices generated uniformly randomly from L to U, and ordered inputs that sort the uniformly random samples in ascending order. Then we generate special case inputs, which serve as bad cases (some are worst cases).

From the experimental results, we can see that all the OPT/LWA ratios are within the theoretical values, and the worst case can reach the theoretical values. Ordered is always worse than random, and the reason is the same as it in the KWA experiment.

Reproducibility Checklist

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes)
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes)
- Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes)
- Does this paper make theoretical contributions? (yes)

If yes, please complete the list below:

- All assumptions and restrictions are stated clearly and formally. (yes)
- All novel claims are stated formally (e.g., in theorem statements). (yes)
- Proofs of all novel claims are included. (yes)
- Proof sketches or intuitions are given for complex and/or novel results. (yes)
- Appropriate citations to theoretical tools used are given. (yes)
- All theoretical claims are demonstrated empirically to hold. (yes)
- All experimental code used to eliminate or disprove claims is included. (yes)

Does this paper rely on one or more datasets? (no)

This is a theoretical work, we only focus on the validation of theoretical results. The data used in the validation experiment is randomly generated, and the methods and code for random generation are provided in Appendix and supplementary materials.

Does this paper include computational experiments? (yes)

If yes, please complete the list below:

- Any code required for pre-processing data is included in the appendix. (yes)
- All source code required for conducting and analyzing the experiments is included in a code appendix. (yes)
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes)
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes)
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes)
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes)
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes)
- This paper states the number of algorithm runs used to compute each reported result. (yes)
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (yes)
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes)
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes)
- This paper states the number and range of values tried per (hyper-)parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes)