

# Augmenting Online Algorithms for Knapsack Problem with Total Weight Information

Binghan Wu, Wei Bao, Bing 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 (Jookan, 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  $i$ th 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 of-

ten 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  $p_1, p_2, \dots, p_n$  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.  $\epsilon$  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  $\lambda$  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  $\lambda$  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  $\alpha$  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 Vas-

silvitskii (Lykouris and Vassilvitskii 2021) (first appeared in 2018) and Kumar, Purohit and Svitkina (Purohit, Svitkina, 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:

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

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

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

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  $\text{OPT}(I)$ . We refer to it as the *value* of  $\text{OPT}$  when  $I$  is clear from the context.

For the known total weight problem (online), let  $\text{ALG}(I)$  represent the final objective value of the online algorithm  $\text{ALG}$  for a given instance  $I$ . We refer to it as the *value of*

$\text{ALG}$  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  $\text{ALG}$  as follows:

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

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  $\text{ALG}(I, \hat{W})$  denote the value of  $\text{ALG}$  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  $\mathcal{I}_{\hat{W}} = \{I \in \mathcal{I} \mid \sum_{i=1}^n v_i = \hat{W}\}$  be the set of input that complies with the prediction  $\hat{W}$ . The online algorithm  $\text{ALG}$  with predicted volume  $\hat{W}$  is  $\beta$ -consistent (Lykouris and Vassilvitskii 2021) if

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

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

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

### 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  $\phi_1$  that does not require the total weight of the remaining items as input and still achieves the optimal competitive ratio:

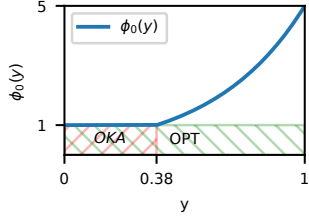


Figure 1: An example of  $\phi_0$  for OKA,  $L = 1$ ,  $U = 5$ . When all items' value-to-weight ratios are  $L$ , the values of OKA and OPT are shown by the red and green areas, respectively.

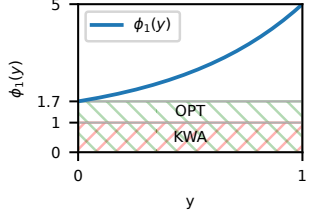


Figure 2: Example of  $\phi_1$  for KWA, with  $L = 1$ ,  $U = 5$ . If items with a value-to-weight ratio  $\theta_1$  arrive first, then items with ratio  $L$ , KWA and OPT are represented by the red and green shaded areas, respectively.

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

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

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

$$f(w) = we^w. \quad (9)$$

We defer the discussion of why considering the total weight of the remaining items cannot improve the competitive ratio to Theorem 3.  $\phi_1$  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  $\phi_1$ . It is worth noting that the OKA (without the fill-up logic Lines 6–7) adopts a different threshold function  $\phi_0$ :

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

---

#### 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**
- 

$\phi_0$  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  $\phi_0$  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.

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  $\phi_1$  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  $\theta_1$ , 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  $\theta_1$  (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  $\phi_1$  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  $\phi_1$ . 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  $\theta_1$ . Recall the example shown in Figure 2. OPT can achieve  $\theta_1$  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  $\Delta\text{KWA}_i$  and  $\Delta\text{OPT}_i$  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

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

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

The definition of  $\Delta\text{KWA}_i$  is as follows. If KWA accepts item  $i$ , its value increase  $\left( \frac{v_i}{w_i} - L \right) w_i$  compared to the lowest possible value-to-weight ratio  $L$ .  $\Delta\text{OPT}_i$  is a 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.  $\Delta\text{KWA}_i$  and  $\Delta\text{OPT}_i$  provide the bound for  $\text{KWA}(I)$  and  $\text{OPT}(I)$ :

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

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

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):*

$$\frac{\phi_1(0)}{L} \leq R^*. \quad (15)$$

3. *Boundary condition:*

$$\phi_1(1) = U. \quad (16)$$

4. *Active choice inequality. For any feasible input  $I \in \mathcal{I}$  and for each active choice  $a_i = 1$ , we have*

$$\frac{\Delta\text{OPT}_i}{\Delta\text{KWA}_i} \leq R^*. \quad (17)$$

See our tech report (Binghan, Wei, and Bing 2024).

The four conditions in Lemma 1 basically state the following: For our online KWA, if it initially has the  $\frac{\text{OPT}}{\text{KWA}}$  ratio  $R^*$  in the basic worst case (Condition 2), and if in each step, the increment of the ratio between  $\Delta\text{OPT}$  and  $\Delta\text{KWA}$  does not exceed  $R^*$  (Condition 4), then the competitive ratio is  $R^*$ . Condition 3 ensures the boundary condition that when capacity utilization reaches 1,  $\Delta\text{OPT} = 0$  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  $\mathcal{W}\left(\frac{1}{e} \frac{U-L}{L}\right) + 1$  by verifying its four conditions.

**Theorem 1** (Competitive Ratio Upper-Bound). *The competitive ratio of Algorithm 1 is no more than  $\mathcal{W}\left(\frac{1}{e} \frac{U-L}{L}\right) + 1$  where  $\mathcal{W}$  denotes the Lambert-W function. See our tech report (Binghan, Wei, and Bing 2024).*

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

**Theorem 2** (Tightness). *The upper bound  $\mathcal{W}\left(\frac{1}{e} \frac{U-L}{L}\right) + 1$  is tight (the upper bound can be reached), and therefore Algorithm 1 is  $\mathcal{W}\left(\frac{1}{e} \frac{U-L}{L}\right) + 1$ -competitive. See our tech report (Binghan, Wei, and Bing 2024).*

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  $\mathcal{W}\left(\frac{1}{e} \frac{U-L}{L}\right) + 1$  is the lowest competitive ratio by constructing a special type of problem instance where the value-to-weight ratio increases continuously from  $\theta_1$  to  $U$ . Given such types of instances, no algorithm can achieve a competitive ratio strictly better than  $\mathcal{W}\left(\frac{1}{e} \frac{U-L}{L}\right) + 1$ .

**Theorem 3** (Optimality).  *$\mathcal{W}\left(\frac{1}{e} \frac{U-L}{L}\right) + 1$  is the lowest possible competitive ratio in the exact volume problem. See our tech report (Binghan, Wei, and Bing 2024).*

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, 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 (see our tech report (Binghan, Wei, and Bing 2024)). 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 per-

---

**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  $\text{KWA}(I, \hat{W})$  with  $\lambda$  capacity, and run  $\text{OKA}(I)$  with  $1 - \lambda$  capacity.
- 

form worse than  $\text{KWA}$  on the  $k$ th input. Therefore,  $\text{KWA}$  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  $\text{PWA}$  is: Since  $\text{KWA}$  achieves the optimal competitive ratio when the prediction is accurate, and the classical online knapsack algorithm ( $\text{OKA}$ ) 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  $\text{KWA}$  and run  $\text{OKA}$  with the remaining capacity. Increasing  $\lambda$  will optimize consistency when we are more confident in the prediction but at the expense of robustness, and vice versa.

We run  $\text{KWA}(I, \hat{W})$  with  $\lambda$  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  $\text{KWA}(I, \hat{W})$  decides to accept item  $i$ ,  $\text{PWA}$  uses  $\lambda w_i$  capacity to accept a  $\lambda$  fraction of item  $i$ . In this way, we construct an algorithm whose decisions are  $\lambda$  times that of  $\text{KWA}$ , and the total capacity used is at most  $\lambda$ . We also treat the algorithm  $\text{OKA}$  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  $\text{KWA}(I, \hat{W})$  with probability  $\lambda$  and run  $\text{OKA}(I)$  with probability  $1 - \lambda$ .

### Analysis of PWA

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

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

the parameter  $\lambda$ .

**Theorem 4 (Robustness).** *Algorithm 2 is  $\frac{c_2}{1-\lambda} = \frac{\ln(\frac{U}{L})+1}{1-\lambda}$  robust. See our tech report (Binghan, Wei, and Bing 2024).*

Next, we prove the consistency of  $\text{PWA}$ . Please note that when proving the consistency,  $\hat{W} = W$ , so we can bound  $\text{KWA}(I, \hat{W})$  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. See our tech report (Binghan, Wei, and Bing 2024).*

Let  $\eta = |\hat{W} - W|$  denotes the error of the prediction. Next, we prove that the performance of  $\text{PWA}$  degrades gracefully as the prediction error grows by bounding the competitive ratio of  $\text{PWA}$  when  $\eta \geq 0$ . Let  $c_1(\eta)$  denote the competitive ratio of  $\text{KWA}$  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  $\text{KWA}$  degrades as  $\eta$  increases.

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

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

*See our tech report (Binghan, Wei, and Bing 2024).*

No matter how  $\eta$  changes, the competitive ratio of  $\text{OKA}$  remains  $c_2$ . We have the following theorem to show the competitive ratio of  $\text{PWA}$  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  $\text{PWA}$  given  $\eta \geq 0$  is*

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

*See our tech report (Binghan, Wei, and Bing 2024).*

### 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  $\text{OPT}$  and  $\text{ALG}$ . 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

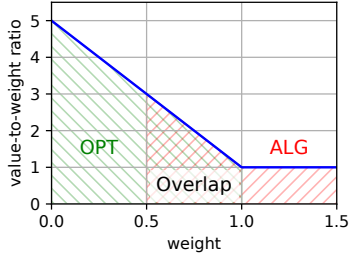


Figure 3: An example of item overlapping with  $W = 1.5$ . Due to the pigeonhole principle, there must be an overlap of width 0.5 on the horizontal axis between OPT and ALG.

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 optimal online knapsack OKA, our new threshold function is segmented and can be viewed as introducing a parameter  $\theta_2$  that controls the segment lengths and the growth rate of the exponential function. Our threshold function is defined as follows:

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

where the parameter  $\theta_2$  is determined by the following. When  $W \leq 1$

$$\theta_2 = 1. \quad (21)$$

When  $1 < W < 2$ ,  $\theta_2$  is the solution of the following equation which can be solved numerically:

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

When  $W > 2$

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

It is worth noting that the solution of formula (22) is 1 when  $W = 1$  and  $\frac{1}{\ln\left(\frac{U}{L}\right) + 1}$  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)  $W \geq 2$ ,

---

### Algorithm 3: Limited Weight Algorithm (LWA)

---

```

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  $y_{i-1} + w_i \leq 1$  and  $v_i \geq \int_{y_{i-1}}^{y_{i-1}+w_i} \phi_2(\delta) d\delta$  then
7:     accept item  $i$ ,  $x_i = 1$ ,  $y_i = y_{i-1} + w_i$ ;
8:   else
9:     reject item  $i$ ,  $x_i = 0$ ,  $y_i = y_{i-1}$ .
10:  end if
11: end for

```

---

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. See our tech report (Binghan, Wei, and Bing 2024).*

Next, when  $W \geq 2$ ,  $\theta_2 = \frac{1}{\ln\left(\frac{U}{L}\right) + 1}$ . 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.

**Theorem 8** (Competitive Ratio Case 2). *When  $W \geq 2$ , the competitive ratio of Algorithm 3 is  $\ln\left(\frac{U}{L}\right) + 1$ . See our tech report (Binghan, Wei, and Bing 2024).*

When  $1 < W < 2$ , this is the most interesting case. At this point, our algorithm can obtain a better competitive ratio than  $\ln\left(\frac{U}{L}\right) + 1$ . 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  $\ln\left(\frac{U}{L}\right) + 1$ .

**Theorem 9** (Competitive Ratio Case 3). *When  $1 < W < 2$ , the competitive ratio of Algorithm 3 is  $\theta_2 \in (1, \ln\left(\frac{U}{L}\right) + 1)$ . See our tech report (Binghan, Wei, and Bing 2024).*

## Conclusion

This paper presents innovative approaches to improving the competitive ratio for the online knapsack problem by leveraging total weight information, whether exact or predicted. The Known Weight Algorithm (KWA) utilizes exact total weight information to achieve an optimal and tight competitive ratio of  $\mathcal{W}\left(\frac{1}{e} \frac{U-L}{L}\right) + 1$ , surpassing the traditional  $\ln\left(\frac{U}{L}\right) + 1$ . Building on this, the Predicted Weight Algorithm (PWA) incorporates total weight predictions, balancing consistency and robustness as prediction accuracy varies, with its competitive ratio degrading gracefully under error. For cases where the total weight is known and less than twice the capacity, the Limited Volume Algorithm (LWA) achieves further improvements. These techniques provide a foundation for future exploration of other variants, including multi-dimensional cases, concave valuation functions, and knapsack problems with departures.

## Acknowledgments

This research was supported by the Australian Research Council (ARC) Discovery Project (DP) DP200103718.

## 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.
- Binghan, W.; Wei, B.; and Bing, Z. 2024. TechReport: Augmenting Online Algorithms for Knapsack Problem with Total Weight Information. <https://github.com/biwu6051/Technical-Reports/blob/main/Online-Knapsack-Problem-With-Total-Weight-Information-Extended-Version.pdf>. Accessed: 2024-12-18.
- 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.
- Jookan, 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.