Learning-augmented Online Algorithm for Two-level Ski-rental Problem

Anonymous submission

Abstract

We study the *two-level ski-rental* problem. In this problem, the user needs to fulfill a sequence of demands for multiple items by choosing one of the three payment options: paying for the on-demand usage (rental), purchasing specific items (single purchase), and purchasing all the items (combo purchase). Without knowing future demand, the user must balance the trade-off between the expensive upfront costs and the potential future expenses. To minimize the sum of the rental cost, the single purchase cost, and the combo purchase cost, we propose a robust online algorithm that offers a worst-case performance guarantee. Although online algorithms are robust to the worst-case situations, they are overcautious and often suffer a poor average performance in typical scenarios. On the other hand, Machine Learning (ML) algorithms often show promising average performance in many applications, while lacking a worst-case performance guarantee. To achieve the best of both worlds, we develop a learning-augmented algorithm by integrating ML predictions into the online algorithm. Our learning-augmented algorithm can outperform the robust online algorithm under accurate predictions (i.e., *consistency*), while ensuring a worst-case performance guarantee even when predictions are inaccurate (i.e., *robustness*). Finally, we conduct experiments on both synthetic and real-world trace data to verify our theoretical results (i.e., achieving both consistency and robustness).

1 Introduction

Making decisions under uncertainty is crucial in many real-world scenarios, such as buying or leasing a car, purchasing a daily or annual parking permit, and so on. Such problems are often modeled as online rent-or-buy problems. One of the classic examples is the *ski-rental* problem, where a skier goes skiing for an unknown number of days and can choose to rent skis at a lower daily cost or buy them at a higher price to ski freely thereafter. The ski-rental problem and its variants (e.g., Transmission Control Protocol (TCP) acknowledgment problem (Dooly, Goldman, and Scott 2001)) have been extensively studied and found various applications in cloud computing (Khanafer, Kodialam, and Puttaswamy 2013), power management (Antoniadis et al. 2021), serverless edge computing (Pan et al. 2022), and others.

However, the ski-rental problem only involves making a one-level decision (i.e., rent or buy for only one item), which makes it inadequate to model more complicated scenarios where multiple levels of decisions are involved. For instance, users often engage with various digital services, including video streaming, music streaming, and ebook reading. Many content providers (such as Amazon, Apple, and Disney) provide three distinct payment options: (i) renting specific content (e.g., movies or TV shows) for a limited period; (ii) buying a membership for specific services (like video or music) that includes all the content they offer; (iii) subscribing to a combo membership to access content across all services. In addition, software providers such as Adobe also offer diverse purchase plans for businesses, including options like monthly software rentals, annual individual software purchases, or annual combo purchases for all software. Other applications, such as mobile data plans or banhcard reservations, also offer similar payment options (Wu, Bao, and Yuan 2021). Hence, exploring how to make a cost-effective choice among multiple payment options can benefit not only individual users but also businesses.

To this end, we consider a two-level ski-rental problem and study the trade-off between rental and two different purchase options. Specifically, we focus on scenarios where a user needs to fulfill the demands for multiple items. Upon the arrival of one demand for some item, the user can cover the demand by choosing one of the three payment options: (i) rental, (ii) single purchase, and (iii) combo purchase. Rental is an on-demand payment with no upfront fee; a single purchase incurs an upfront fee (denoted by C_s) but covers the demands for that specific item until the end; a combo purchase has a higher upfront fee (denoted by $C_c \geq C_s$) while covering the demands for all the items until the end. The challenge in the two-level ski-rental problem lies in the intricate decision-making between single purchases and a combo purchase covering multiple items, while the ski-rental problem only considers single purchases. To minimize the total cost (i.e., the sum of the rental cost and the (single and combo) purchase cost), we develop an online algorithm that makes decisions without knowing any future information. We use the *competitive ratio (CR)* to evaluate our algorithm, which is defined as the ratio between the cost of the online algorithm and that of the optimal offline algorithm, over all the inputs (see the formal definition of CR in Section 2). We prove that the CR of our algorithm is at most $3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s})$.

Although online algorithms are robust against the worst-

case situation, they often exhibit excessive caution towards uncertainty, resulting in poor average performance in many real-world scenarios. In contrast, machine learning (ML) algorithms show promising average performance in many applications by using historical data to construct suitable prediction models. However, their drawback is a lack of robustness guarantee. For example, ML predictions can be far from correct if the test data is chosen adversarially (Szegedy et al. 2014) or the distribution drift happens (Lu et al. 2018).

In the second part of this work, we first show that a simple algorithm that blindly follows ML predictions cannot offer a robustness guarantee for worst-case scenarios. Then, we design a novel learning-augmented online algorithm by integrating ML predictions into our online algorithm. Specifically, we show that our algorithm achieves two desired properties: (i) given an accurate prediction, it offers a better CR than our online algorithm *(consistency)* and (ii) even if the prediction is bad, its performance is close to (by a constant factor) that of our online algorithm *(robustness)*.

We summarize our main contributions in the following.

First, we formulate the two-level ski-rental problem and study the trade-off between rental costs and two types of purchase costs. The objective is to minimize the total cost by choosing from rental, single purchase, and combo purchase in an online manner. We propose a online algorithm and prove that it achieves a CR of $3-\frac{1}{C_s}-\frac{1}{C_c}(2-\frac{1}{C_s})$. Our algorithm has a stronger performance guarantee compared to the State-of-the-Art algorithm studied by Wu, Bao, and Yuan (2021) (see Remarks 3.1 & 3.2).

Second, we design a novel learning-augmented algorithm that integrates ML predictions into online decision-making. We prove that the proposed algorithm can guarantee both consistency and robustness. To the best of our knowledge, this is the first work that designs learning-augmented algorithms for the two-level ski-rental problem.

Finally, we conduct numerical experiments to evaluate the performance of our online algorithm and learning-augmented algorithm using both synthetic and real-world datasets. The results corroborate our theoretical analyses and show that our learning-augmented algorithm guarantees both consistency and robustness.

Related Work. The ski-rental problem is first considered by Karlin et al. (1988), which gives the best deterministic online algorithm with a 2 competitive ratio. Then, an optimal randomized online algorithm achieving $e/(e-1)$-competitiveness is proposed in Karlin et al. (1994). Several variants of the ski-rental problem are later introduced, and in this context, we list some work that examines scenarios involving multiple payment options. In Ai et al. (2014), a multi-shop ski-rental problem is considered, where a user needs to choose when and where to buy from multiple shops with different prices for renting and purchasing. Recently, a multi-commodity ski-rental problem is studied in Wu et al. (2022), where a user uses a set of commodities altogether and needs to choose payment options for each commodity. However, all the aforementioned work deals with the case where the purchase option is limited to covering a single item. Closest to our work is the one by Wu, Bao, and Yuan (2021), which considers the combo purchase option to cover all the items and proposes a deterministic online algorithm with competitive ratio $3-\frac{1}{C_s}$. However, we find a counterexample to show that their algorithm's CR will be unbounded under it. On the other hand, our algorithm is proven to be $3-\frac{1}{C_s}-\frac{1}{C_c}(2-\frac{1}{C_s})$ and is tighter than theirs. Furthermore, they do not consider the uses of ML predictions.

On the other hand, researchers start to design learning-augmented algorithms that can achieve robustness and consistency. We adapt the concepts of consistency and robustness from the seminal work (Lykouris and Vassilvtiskii 2018). By incorporating ML predictions, they show that the classic online Marker algorithm can achieve robustness and consistency for the caching problem. Learning-augmented algorithm design is also studied in the ski-rental problem and its variants (Purohit, Svitkina, and Kumar 2018; Bamas, Maggiori, and Svensson 2020; Wang, Li, and Wang 2020). For example, in Wang, Li, and Wang (2020), both single and multiple ML predictions are incorporated into the online algorithm for the multi-shop ski-rental problem. We refer interested readers to the survey (Boyar et al. 2017; Mitzenmacher and Vassilvitskii 2022) for a comprehensive discussion. However, in our work, the combo purchase covers all the single purchases, resulting in coupled decisions. This "two-level" payment option brings new challenges for our algorithm design but is more practical in the real-world.

2 System Model and Problem Formulation

System Model. We consider a discrete-time system where a single user needs to fulfill the demands for K items. Upon the arrival of each demand, the user can fulfill the demands by choosing different renting/purchasing options. Specifically, we assume that the system is time-slotted with time-slot $t = 1, 2, \ldots, T$, where T is a finite time horizon that is unknown to the user and is allowed to be arbitrarily large. Without loss of generality, we assume that in each time-slot, there are demands for one item only.[1] Let $d(t) = (i(t), a(t))$ denote the demand arriving in time-slot t, where $i(t) \in \{1, 2, \ldots, K\}$ is the index of the item that is associated with the demand and $a(t) \geq 0$ is the amount of the demand. In particular, $d(t) = (0, 0)$ means that there is no demand in time-slot t. Then, let $\mathbf{D} = \{d(t)\}_{t=1}^{T}$ denote the sequence of demand over the entire time horizon T.

To fulfill the demand in each time-slot, the user can choose one of the three payment options: (i) Rental, (ii) Single purchase, and (iii) Combo purchase.

(i) **Rental**: The user pays a unit rental cost to cover one unit of demand in time-slot t.

(ii) **Single purchase**: The user pays upfront a single purchase fee $C_s > 1$, which covers all the demands for item $i(t)$ from t to T. For ease of analysis, we assume that the single purchase cost C_s is the same for all the items.

(iii) **Combo purchase**: The user pays a higher upfront fee $C_c \in (C_s, KC_s)$, which covers the demands for all the

[1] We can adapt our algorithm to a more general setting where demands for multiple items arrive simultaneously and prove the same upper bound on the competitive ratio. (see Remark 3.3).

items from t to T.

We assume that once the payment is made in a time-slot t, the associated demands can be fulfilled immediately. While choosing to rent may offer a lower instantaneous cost, this may result in a larger future expense when confronted with increased future demands. In contrast, while choosing to purchase (either single or combo) incurs a higher upfront cost, it may lead to a lower total cost in the long term.

Problem Formulation. Let $r(t) \in \{0, 1\}$ be the rental decision for the demand arriving in time-slot t, where $r(t) = 1$ if the user decides to rent, and $r(t) = 0$ otherwise. In addition, we use $x_k \in \{0, 1\}$ to denote whether the user makes a single purchase for item k ($x_k = 1$) or not ($x_k = 0$); similarly, we use $x_c \in \{0, 1\}$ to denote whether the user makes a combo purchase ($x_c = 1$) or not ($x_c = 0$). A scheduling algorithm π is denoted by $\pi = \{\{r^\pi(t)\}_{t=1}^T, \{x_k^\pi\}_{k=1}^K, x_c^\pi\}$, where $r^\pi(t)$, x_k^π, and x_c^π are algorithm π's rental decision in time-slot t, single purchase decision for item k, and combo purchase decision, respectively. For notational simplicity, we drop the superscript π in the rest of the paper whenever the context is clear. Let $\mathcal{C}(\mathbf{D}, \pi)$ be the total cost under an algorithm π, which can be computed as

$$\mathcal{C}(\mathbf{D}, \pi) = \sum_{t=1}^{T} a(t)r(t) + \sum_{k=1}^{K} C_s x_k + C_c x_c, \qquad (1)$$

where the three terms are the total rental cost, the total single purchase cost, and the combo purchase cost, respectively.

We focus on the class of *online* algorithms, denoted by Π, under which the information available in time-slot t for making decisions includes the rental decision history $\{r(\tau)\}_{\tau=1}^t$, the single purchase history $\{x_k\}_{k=1}^K$, the combo purchase history x_c, the demand history $\{d(\tau)\}_{\tau=1}^t$, the single purchase cost C_s, and the combo purchase cost C_c. No future information is available to the algorithm. Given a demand sequence $\mathbf{D}$, the objective is to minimize the total cost:

$$\min_{r(t), x_k, x_c} \quad \sum_{t=1}^{T} a(t)r(t) + \sum_{k=1}^{K} C_s x_k + C_c x_c \qquad (2a)$$

$$\text{subject to} \quad r(t) + \sum_{k=1}^{K} x_k \mathbb{1}_{(k,i(t))} + x_c \geq 1, \qquad (2b)$$

$$\forall t \in \{1, 2, \ldots, T\},$$

$$r(t) \in \{0, 1\}, \ \forall t \in \{1, 2, \ldots, T\}, \qquad (2c)$$

$$x_k \in \{0, 1\}, \ \forall k \in \{1, 2, \ldots, K\}, \qquad (2d)$$

$$x_c \in \{0, 1\}, \qquad (2e)$$

where $\mathbb{1}_{(k,i(t))}$ in constraint (2b) is the indicator function of whether $k = i(t)$ (i.e., $\mathbb{1}_{(k,i(t))} = 1$ if $k = i(t)$; $\mathbb{1}_{(k,i(t))} = 0$ otherwise). Constraint (2b) states that each demand must be fulfilled by either rental or (single or combo) purchase.

Without the knowledge of the future demand in advance, it is usually hard for an online algorithm to attain the same minimum total cost achieved by an optimal offline algorithm, which requires the knowledge of the entire demand sequence $\mathbf{D}$ beforehand. To measure the performance of an online algorithm, we consider a widely adopted metric called *competitive ratio* (CR) (Borodin and El-Yaniv 2005), which is defined as the ratio of the cost of the online algorithm to that of the optimal offline algorithm, in the worst-case over all feasible inputs. Formally, an online algorithm $\pi \in \Pi$ is called c-competitive if there exists a constant $c \geq 1$ such that for any demand sequence $\mathbf{D}$, we have

$$\mathcal{C}(\mathbf{D}, \pi) \leq c \cdot OPT(\mathbf{D}), \qquad (3)$$

where $OPT(\mathbf{D})$ is the total cost of the optimal offline algorithm under the demand sequence $\mathbf{D}$.

3 Online Algorithm Design and Analysis

In this section, we first present two important notions used in our online algorithm: indicative cost and threshold. Then, we introduce our algorithm and show its competitive ratio.

Indicative Cost and Threshold. Before presenting our online algorithm, we first establish some essential definitions that will be used in the algorithm. The indicative cost $\psi_k(t)$ for item k is defined as the total demand of item k during the interval $[1, t]$ that has not been covered by a previous purchase. Formally, it evolves as follows:

$$\psi_k(t) \coloneqq \begin{cases} \psi_k(t-1) + a(t), & i(t) = k \text{ and } x_k = x_c = 0, \\ \psi_k(t-1), & \text{otherwise}, \end{cases} \qquad (4)$$

where $\psi_k(t)$ increases by $a(t)$ if the demand in time-slot t is for item k (i.e., $i(t) = k$) and neither a single purchase for item k or a combo purchase has been made (i.e., $x_k = x_c = 0$); otherwise, $\psi_k(t)$ remains unchanged. We define $\lambda_s \in (1, C_s]$ as the threshold for making single purchases. That is, if the single purchase indicator $\psi_k(t)$ reaches λ_s (i.e., $\psi_k(t) \geq \lambda_s$), we will make a single purchase for item k. We assume that the threshold λ_s is the same for all the items since they have the same single purchase cost C_s.

Similarly, for the combo purchase, we introduce the overall indicative cost $\psi_c(t)$ as well as the combo purchase threshold $\lambda_c \in (1, C_c]$. The overall indicative cost $\psi_c(t)$ is defined as the sum of the minimum between $\psi_k(t)$ and λ_s over all items, i.e.,

$$\psi_c(t) \coloneqq \sum_{k=1}^{K} \min\{\psi_k(t), \lambda_s\}. \qquad (5)$$

When $\psi_c(t)$ reaches the combo purchase threshold λ_c (i.e., $\psi_c(t) \geq \lambda_c$), we will make a combo purchase.

Algorithm Description. We present our online *Two-level Ski-rental Algorithm* in Algorithm 1 and explain how it works in the following. Algorithm 1 takes the number of items K, the single purchase cost C_s, the combo purchase cost C_c, and the demand sequence $\mathbf{D}$ (revealed in an online manner) as inputs, and outputs the decision $\{\{r(t)\}_{t=1}^T, \{x_k\}_{k=1}^K, x_c\}$. When a new demand $d(t)$ arrives in time-slot t, if it is already covered by a previous (single or combo) purchase (Line 3), Algorithm 1 does nothing. Otherwise, it updates the indicative costs $\psi_k(t)$ and $\psi_c(t)$ according to Eqs. (4) and (5), respectively (Lines 6-7). If the indicative cost exceeds the threshold, Algorithm 1 will

Algorithm 1: Two-level Ski-rental Algorithm

Input : K, C_s, C_c, λ_s, λ_c, $\mathbf{D}$ (revealed in an online manner)
Output: $\{r(t)\}_{t=1}^{T}, \{x_k\}_{k=1}^{K}, x_c$
Init. : $\psi_c(t), \psi_k(t), r(t), x_k, x_c \leftarrow 0$ for all k,t

1 **for** $t = 1, 2, \ldots, T$ **do**
2 Demand $d(t) = ((i(t), a(t))$ arrives;
3 **if** $d(t)$ *is covered by a previous purchase* **then**
4 Do nothing and wait till next time-slot;
5 **else**
6 Update $\psi_k(t)$ according to Eq. (4);
7 Update $\psi_c(t)$ according to Eq. (5);
8 **if** $\psi_c(t) \geq \lambda_c$ **then**
9 $x_c \leftarrow 1$; `// combo purchase`
10 **else if** $\psi_{i(t)}(t) \geq \lambda_s$ **then**
11 $x_{i(t)} \leftarrow 1$; `// single purchase`
12 **else**
13 $r(t) \leftarrow 1$; `// rental`
14 **end**
15 **end**
16 **end**

make the corresponding purchase (Lines 8-11). If not, it will cover the demand by rental (Line 13). Intuitively, when indicative cost $\psi_k(t)$ reaches the single purchase threshold λ_s, it reflects a substantial demand for item k, leading Algorithm 1 to opt for the single purchase of that item. Likewise, when the overall indicative cost $\psi_c(t)$ reaches λ_c, it indicates substantial demand for multiple items. So Algorithm 1 makes the combo purchase immediately, rather than proceeding with additional single purchases or rentals.

Competitive Analysis. The CR of Algorithm 1 is a function of the thresholds λ_s and λ_c. By choosing the thresholds λ_s and λ_c properly, we show the CR of Algorithm 1 is upper bounded by 3.

Theorem 3.1. *The CR of Algorithm 1 is upper bounded by*

$$3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s}). \tag{6}$$

In particular, this upper bound can be derived by choosing $\lambda_s = C_s$ and $\lambda_c = C_c$.

Proof sketch. We provide the proof in the supplementary materials and give a proof sketch in the following. First, given any λ_s and λ_c, we first get an upper bound of the CR, which is a function of λ_s and λ_c. By optimizing λ_s and λ_c, we achieve an upper bound of $3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s})$ when $\lambda_s = C_s$ and $\lambda_c = C_c$.

As it is hard to analyze the CR of Algorithm 1 on arbitrary demand sequences, we follow the proof framework provided by Wu, Bao, and Yuan (2021), which establishes the *standard total demand sequencs* so that the CR of their algorithm can be bounded by these sequences. Similarly, the CR of our algorithm can be bounded by these sequences as well. By doing so, the CR can be expressed using standard total demand sequences, the thresholds (i.e., λ_s and λ_c), and the purchase costs (i.e., C_s and C_c). Next, for all the standard total demand sequences, we split them into two cases. First, if Algorithm 1 makes the combo purchase on the sequences, the CR is upper bounded by $\frac{\lambda_c - 1 + C_c + (\lambda_c - 1)\lambda_s^{-1}(C_s - 1)}{\lambda_c}$. When $\lambda_s = C_s$ and $\lambda_c = C_c$, we achieve an upper bound of $3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s})$. Second, if Algorithm 1 does not make the combo purchase, this case is dominated by the first case. To see this, if the optimal offline algorithm makes the combo purchase, there exists another sequence in the first case without reducing the CR. Otherwise, the CR is upper bounded by $2 - \frac{1}{C_s}$, which is greater than $3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s})$. □

Remark 3.1. *Wu, Bao, and Yuan (2021) proposed a similar deterministic online algorithm and showed that their algorithm is $(3 - 1/C_s)$-competitive. However, we find a counterexample that suggests an issue with their algorithm in the setting we consider. Specifically, consider a demand sequence $\tilde{\mathbf{D}} = \{(1, C_s), (2, C_s), \ldots, (K, C_s)\}$, where each item has only one demand with an amount of C_s. In their algorithm, single purchases are made for all the items, while the combo purchase is never made since their algorithm skips the updates of the overall indicative cost $\psi_c(t)$ after a single purchase. With the fact that the cost of the optimal offline algorithm is at most C_c, their algorithm's competitive ratio is at least KC_s/C_c, which can be very large when $KC_s \gg C_c$. We address this problem by redefining the indicative cost and assigning the highest priority to the combo purchase.*

Remark 3.2. *Compared to Wu, Bao, and Yuan (2021), our work makes two contributions: First, we extend the standard total demand to more general λ_s and λ_c, which helps us establish our main results in Theorem 4.1 (see the proof sketch of Theorem 4.1). Second, we slightly improve the upper bound of CR (i.e., $CR \leq 3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s}) < 3 - \frac{1}{C_s}$).*

Remark 3.3. *Although we assume that at most one item arrives in each time-slot, Algorithm 1 can be adapted to a more general setting where multiple items can arrive in the same time-slot with the same CR guarantee. To accommodate the multi-item arrival setting, Algorithm 1 can be modified as follows: In Lines 6-7, Algorithm 1 first updates the values of $\psi_k(t)$ and $\psi_c(t)$ for the multiple items. Then, Algorithm 1 makes the combo purchase decision based on the updated $\psi_c(t)$ and makes the single purchase decision based on the updated $\psi_k(t)$. By doing this, the cost of Algorithm 1 under the multiple-item arrival setting remains no larger than the cost under the single-item arrival setting. The optimal offline algorithm does not change, as the total demand for each item remains the same. As a result, the CR still holds.*

4 Learning-augmented Online Algorithm

In the last section, we introduce a robust online algorithm that has a worst-case guarantee. However, online algorithms are often overly conservative and may have a poor average performance in real-world scenarios. Conversely, ML algorithms have the advantage of utilizing extensive historical data and building well-fitted models, which enables them to achieve a promising average performance. Nonetheless, ML

algorithms lack a worst-case guarantee when facing adversaries, outliers, and distribution shifts. To attain the best of both approaches, we design a learning-augmented online algorithm that achieves both consistency and robustness.

4.1 Machine Learning Predictions

We consider the case where an ML algorithm provides us a prediction $y = \{y_1, \ldots, y_K\}$, where y_k represents the predicted total demand for item k. Total demand predictions are commonly used in real-world applications. Take cloud computing as an example, various resources such as CPU, memory, and network bandwidth can be fed into a predictor to obtain precise future load (Masdari and Khoshnevis 2020). We assume that the ML prediction y is available in the very beginning (i.e., $t = 0$) and is provided in full (i.e., all y_k are available). The actual total demand can be represented by $z = \{z_1, \ldots, z_K\}$, where z_k denotes the total demand of item k, i.e., $z_k = \sum_{t=1}^{T} \mathbb{1}_{(k,i(t))} \cdot a(t)$. The prediction error is defined as the $\ell 1$-norm distance from the actual total demand to the predicted ones. We use $\eta_k := |y_k - z_k|$ to denote the prediction error for item k and use $\eta := \sum_{k=1}^{K} \eta_k$ to denote the total prediction error, respectively.

A learning-augmented online algorithm receives a prediction y, a trust parameter θ ($0 < \theta \leq 1$), the single purchase cost C_s, the combo purchase cost C_c, and a demand sequence $\mathbf{D}$ (revealed in an online manner) as inputs, and outputs a solution with cost $\mathcal{C}(\mathbf{D}, y, \theta)$. Here θ denotes our confidence in the prediction (smaller θ means higher confidence). The competitive ratio of a learning-augmented algorithm is a function $c(\eta)$ of the prediction error η. That is, for any pair of demand sequence $\mathbf{D}$ and the ML prediction y with a total prediction error η, a learning-augmented online algorithm is called $c(\eta)$-competitive if it can achieve

$$\mathcal{C}(\mathbf{D}, y, \theta) \leq c(\eta) \cdot OPT(\mathbf{D}), \tag{7}$$

where $OPT(\mathbf{D})$ is the total cost of the optimal offline algorithm under the demand sequence $\mathbf{D}$. Furthermore, a learning-augmented algorithm is called γ-*robust* if $c(\eta) \leq \gamma$ for any η, and is called β-*consistent* if $c(0) \leq \beta$. Here robustness measures the performance of a learning-augmented algorithm in the worst-case scenario of extremely inaccurate predictions, while consistency measures the performance in the best-case scenario of perfect predictions. Our goal is to design a learning-augmented online algorithm that achieves robustness and consistency at the same time.

4.2 A Simple Algorithm: Follow the Prediction

First, we show that an algorithm that naively follows the prediction (FTP) cannot guarantee robustness. FTP operates in the following ways: FTP first computes the total demands of the prediction y as $\sum_{k=1}^{K} \min\{C_s, y_k\}$ and makes a combo purchase if the total demands exceed the combo purchase threshold (i.e., $\sum_{k=1}^{K} \min\{C_s, y_k\} \geq C_c$); otherwise, FTP makes the single purchase for item k if $y_k \geq C_s$ or rents for item k forever. We denote the cost of FTP as $\mathcal{C}(\mathbf{D}, y)$.

Lemma 4.1. *FTP satisfies* $\mathcal{C}(\mathbf{D}, y) \leq OPT(\mathbf{D}) + \eta$.

Proof sketch. The detailed proof is provided in the supplementary materials and we give a proof sketch in the following. There are two cases for FTP: (A) If $\sum_{k=1}^{K} \min\{C_s, y_k\} \geq C_c$, FTP will make the combo purchase in $t = 1$; (B) Otherwise, FTP will never make the combo purchase. Similarly, for the optimal offline algorithm, we have: (I) If $\sum_{k=1}^{K} \min\{C_s, z_k\} \geq C_c$, the optimal offline algorithm makes the combo purchase in $t = 1$; (II) Otherwise, the optimal offline algorithm never makes the combo purchase. We show that the result holds for all those 4 cases.

For case (A-I), $\mathcal{C}(\mathbf{D}, y) = OPT(\mathbf{D}) = C_c$. For case (A-II), $\mathcal{C}(\mathbf{D}, y) = C_c$, $OPT(\mathbf{D}) = \sum_{k=1}^{K} \min\{C_s, z_k\}$, and $\sum_{k=1}^{K} \min\{C_s, y_k\} \geq C_c$. For each item, we show that $\min\{C_s, y_k\} \leq \min\{C_s, z_k\} + \eta_k$. Therefore, $\mathcal{C}(\mathbf{D}, y) = C_c \leq \sum_{k=1}^{K} \min\{C_s, y_k\} \leq \sum_{k=1}^{K} \min\{C_s, z_k\} + \eta_k = OPT(\mathbf{D}) + \eta$. For case (B-I), $OPT(\mathbf{D}) = C_c$, $\sum_{k=1}^{K} \min\{C_s, y_k\} \leq C_c$. Denote the cost of FTP as $\mathcal{C}(\mathbf{D}, y)_k$, and $\mathcal{C}(\mathbf{D}, y)_k$ equals C_s if $y_k \geq C_s$ and z_k if not. For each item k, we show that $\mathcal{C}(\mathbf{D}, y)_k \leq \min\{C_s, y_k\} + \eta_k$. Therefore, $\mathcal{C}(\mathbf{D}, y) = \sum_{k=1}^{K} \mathcal{C}(\mathbf{D}, y)_k \leq \sum_{k=1}^{K} \min\{C_s, y_k\} + \eta_k \leq C_c + \eta = OPT(\mathbf{D}) + \eta$. Finally, for case (B-II), both algorithms do not make the combo purchase. Then it is equivalent to the ski-rental problem. Applying Lemma 2.1 from (Purohit, Svitkina, and Kumar 2018) directly yields the result. □

Remark 4.1. *When $\eta = 0$, i.e., the prediction is perfect, FTP performs closely to the optimal offline algorithm. However, FTP can perform badly when the prediction error η is large. This indicates that FTP only can achieve consistency but lacks a robustness guarantee.*

4.3 Learning-augmented Algorithm Design

We have demonstrated that following the prediction naively (i.e., FTP) cannot guarantee the worst-case performance. In this section, we will delve into how to design a learning-augmented online algorithm that achieves both robustness and consistency by appropriately incorporating ML predictions into our online algorithm.

The key idea of our learning-augmented algorithm, Algorithm 2, is to modify the thresholds in our online algorithm according to the prediction y and the trust parameter θ. Specifically, in Algorithm 2, we set an individual single purchase threshold $\lambda_{s,k}$ for item k since each item has its own predicted total demand y_k and still set the combo purchase threshold to be λ_c. Given the prediction y and the trust parameter θ, Algorithm 2 adjusts the thresholds as follows: For each item, if its predicted total demand is less than the single purchase price, (i.e., $y_k < C_s$), Algorithm 2 will increase the single purchase threshold $\lambda_{s,k}$ to C_s/θ. Otherwise, the single purchase threshold will be reduced to θC_s. Intuitively, a larger threshold renders the algorithm reluctant to make a purchase while a smaller threshold encourages the algorithm to make a purchase. Similarly, if the prediction does not suggest making a combo purchase (i.e., $\sum_{k=1}^{K} \min\{C_s, y_k\} < C_c$), Algorithm 2 will increase the combo purchase threshold to C_c/θ; otherwise, it will reduce

Algorithm 2: Learning-augmented Two-level Ski-rental Algorithm

Input : y, θ, K, C_s, C_c, $\mathbf{D}$ (revealed in an online manner)
Output: $\{r(t)\}_{t=1}^{T}, \{x_k\}_{k=1}^{K}, x_c$
Init. : $\psi_c(t), \psi_k(t), r(t), x_k, x_c \leftarrow 0$ for all k,t

for *item* $k = \{1, 2, \ldots, K\}$ **do**
  if $y_k < C_s$ **then**
    $\lambda_{s,k} \leftarrow C_s/\theta$;
  else
    $\lambda_{s,k} \leftarrow \theta C_s$;
  end
end
if $\sum_{k=1}^{K} \min(C_s, y_k) < C_c$ **then**
  $\lambda_c \leftarrow C_c/\theta$;
else
  $\lambda_c \leftarrow \theta^2 C_c$;
end
Run Algorithm 1 with the thresholds λ_c and $\lambda_{s,k}$;

the combo purchase threshold to $\theta^2 C_c$.[2] Once the thresholds are determined, Algorithm 2 operates identically to our online algorithm Algorithm 1.

The intuition of our learning-augmented algorithm is to mimic Algorithm 1 and FTP by choosing different θ. One can observe that when Algorithm 2 does not trust the prediction y at all (i.e., set $\theta = 1$), it is identical to our online algorithm Algorithm 1. In contrast, when Algorithm 2 fully trusts the prediction y (i.e., set $\theta \to 0$), it is identical to FTP. That is, when the prediction y makes an (either combo or single) purchase, then Algorithm 2 will also make the same purchase by decreasing the purchase threshold to 0; while when the prediction y does not make a purchase, Algorithm 2 will also not make such a purchase by increasing the purchase threshold to infinitely large. Finally, when $\theta \in (0, 1)$, Algorithm 2 combines the advice from both algorithms, resulting in a trade-off between robustness and consistency.

4.4 Learning-augmented Algorithm Analysis

Theorem 4.1. *The CR of Algorithm 2 is upper bounded by* $\min\{1 + \theta^{-1} + \theta^{-3}, (1 + \theta + \theta^2) + \frac{1+2\theta}{1-\theta}\frac{\eta}{OPT(\mathbf{D})}\}$. *That is, Algorithm 2 is* $(1 + \theta^{-1} + \theta^{-3})$*-robust and* $(1 + \theta + \theta^2)$*-consistent.*

Remark 4.2. *When the prediction is accurate (i.e.,* $\eta = 0$*), the CR of Algorithm 2 is at most* $1+\theta+\theta^2$ *as* $1+\theta+\theta^2 < 1+\theta^{-1}+\theta^{-3}$*, i.e., it is* $(1+\theta+\theta^2)$*-consistent. In particular, if the algorithm trusts the prediction (i.e.,* $\theta \to 0$*) in the meantime, the CR of Algorithm 2 approaches to* 1 *as* $1 + \theta + \theta^2 \to 1$ *and* $\frac{1+2\theta}{1-\theta}\frac{\eta}{OPT(\mathbf{D})} = 0$*, achieving* 1*-consistency.*

Remark 4.3. *Given any* θ*, the CR of Algorithm 2 is at most* $1 + \theta^{-1} + \theta^{-3}$*, implying that the worst-case performance of Algorithm 2 is always bounded by a constant irrespective of the predictive error, achieving robustness. In particular, if Algorithm 2 does not trust the prediction at all (i.e.,* $\theta = 1$*), then we have* $1 + \theta^{-1} + \theta^{-3} = 3$*. So Algorithm 2 has a CR of* 3 *and performs closely to our online algorithm.*

Remark 4.4. *We present an example to show why Algorithm 2 fails to achieve* 1*-consistency when* $\theta \to 0$ *if the combo purchase threshold is set to* θC_c *instead of* $\theta^2 C_c$*. Suppose* $C_c = (K-1)C_s + 1$ *and consider the demand sequence* $\tilde{\mathbf{D}} = \{(1, C_s), (2, C_s), \ldots, (K, C_s)\}$*. If the prediction is perfect, Algorithm 2 will have* $\lambda_c = \theta C_c$ *and* $\lambda_{s,k} = \theta C_s$ *for all the items. For any* $\theta \in (0, 1]$*, Algorithm 2 makes the single purchase for the first* $K-1$ *items and makes the combo purchase for the last item. Consequently, its cost equals* $(K-1)C_s + C_c$*. With the fact that* $OPT(\tilde{\mathbf{D}})$ *is* C_c*, the CR of Algorithm 2 is at least* $\frac{(K-1)C_s + C_c}{C_c} = 2 - \frac{1}{C_c}$.

We provide the proof sketch of Theorem 4.1 as follows and give the detailed proof in the supplementary materials.

Proof sketch. We first prove the first bound $1 + \theta^{-1} + \theta^{-3}$. The proof idea is similar to that of Theorem 3.1: Bound the CR by the standard total demand sequences then find an upper bound of CR on the sequences.

Specifically, the most important step is to construct the standard total demand sequences for Algorithm 2. Note that the standard total demand we use in the proof of Theorem 3.1 cannot be applied here directly, as Algorithm 2 has two different single purchase thresholds (θC_s and C_s/θ). Therefore, we first establish the standard total demand sequences for Algorithm 2. The key idea is to find a way to delete/move the demand without reducing the CR. Once we get the standard total demand sequence, the following is to analyze the upper bound of CR on these sequences. We consider two cases: $\lambda_c = \theta^2 C_c$ and $\lambda_c = C_c/\theta$. For each case, we consider the same two cases that are discussed in Theorem 3.1 and get an upper bound for both. Finally, combining cases (I) and (II) yields the result.

Then, we prove the second bound $(1 + \theta + \theta^2) + \frac{1+2\theta}{1-\theta}\frac{\eta}{OPT(\mathbf{D})}$. Again, we consider the same two cases: $\lambda_c = \theta^2 C_c$ and $\lambda_c = C_c/\theta$. For each case, we consider the similar 4 cases that are discussed in Lemma 4.1.

Specifically, when $\lambda_c = \theta^2 C_c$, we discuss the CR by 2×2 cases, i.e., whether Algorithm 2 makes the combo purchase (A) or not (B) and whether the optimal offline makes the combo purchase (I) or not (II). For case (A-I), we have the cost of Algorithm 2 $\mathcal{C}(\mathbf{D}, y, \theta) \le (1 + \theta + \theta^2)C_c$. Thus, $\mathcal{C}(\mathbf{D}, y, \theta) \le (1 + \theta + \theta^2)OPT(\mathbf{D}) \le (1 + \theta + \theta^2)(OPT(\mathbf{D}) + \eta)$ due to $OPT(\mathbf{D}) = C_c$. For case (A-II), still, we have $\mathcal{C}(\mathbf{D}, y, \theta) \le (1 + \theta + \theta^2)C_c$. Then we have $C_c \le \sum_{k=1}^{K} \min\{C_s, y_k\} \le \sum_{k=1}^{K} \min\{C_s, x_k\} + \eta_k = OPT(\mathbf{D}) + \eta$, where the second inequality obtained in the proof of Lemma 4.1. Thus, we have $\mathcal{C}(\mathbf{D}, y, \theta) \le (1 + \theta + \theta^2)(OPT(\mathbf{D}) + \eta)$. For case (B-I), we show that it does not exist. For case (B-II), we can use Theorem 2.2 from Purohit, Svitkina, and Kumar (2018) directly and have $\mathcal{C}(\mathbf{D}, y, \theta) \le (1 + \theta) + \frac{\eta}{(1-\theta)OPT(\mathbf{D})}$.

When $\lambda_c = C_c/\theta$, we also discuss 4 cases. For case (A-I), we have $C_c \le \frac{1-\theta}{\theta}\eta$. Then, we show that the

[2] In Remark 4.4, we show that Algorithm 2 fails to achieve consistency by simply setting $\lambda_c = \theta C_c$.

cost of Algorithm 2 is bounded by $C_c + C_c(\frac{1+2\theta}{1-\theta})$. Thus, $\mathcal{C}(\mathbf{D}, y, \theta) \leq C_c + (\frac{1+2\theta}{1-\theta})\eta = OPT(\mathbf{D}) + (\frac{1+2\theta}{1-\theta})\eta$. For case (A-II), we show that it does not exist. For case (B-I), the cost of Algorithm 2 for each item k is bounded by $(1+\theta)(\min\{C_s, y_k\} + \eta_k)$. Thus, we have $\mathcal{C}(\mathbf{D}, y, \theta) \leq (1+\theta)(\sum_{k=1}^{K} \min\{C_s, y_k\} + \eta_k) \leq (1+\theta)(C_c + \eta) = (1+\theta)(OPT(\mathbf{D}) + \eta)$. Case (B-II) is the same as $\lambda_c = \theta^2 C_c$. Combining all the cases, we get the second bound. □

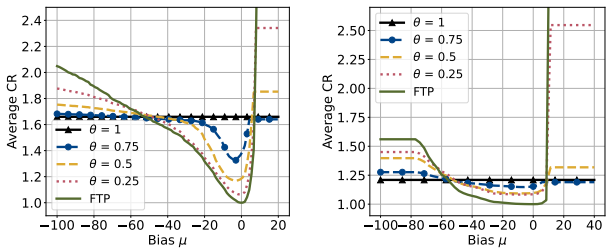
5 Experimental Results

In this section, we conduct experiments using synthetic data as well as real-world data to demonstrate our online algorithm achieves promising average performance on both datasets. In addition, our learning-augmented algorithm showcases both consistency and robustness.

5.1 Synthetic Dataset

First, we show the performance of our online and learning-augmented algorithms on the synthetic dataset. The setting is as follows. We set the number of items as $K = 10$, the single purchase cost as $C_s = 10$, and the combo purchase cost as $C_c = 80$. The demand sequence $\mathbf{D}$ is generated in the following way: First, the time horizon T is a random variable drawn from Gaussian distribution with mean $K \cdot C_s$ and standard deviation 100. Then, for each time-slot $t \in [1, T]$, the demand is assigned to one item in two ways: (i) **Uniform:** The demand is uniformly distributed among all the items. (ii) **Long-tail:** The demand is assigned based on the Pareto law, i.e., 80% demands are assigned to 20% items. Finally, the amount of demand $a(t)$ is drawn from the Poisson distribution with mean 1. For each demand sequence, we construct the prediction y by adding noise to the actual total demand z. Specifically, the actual total demand of each item k is perturbed by a bias of value μ, i.e., $y_k = z_k + \mu$. This represents the scenario where the prediction is biased due to the distribution shift between training and testing data. We provide the detailed setting in the supplementary materials.

Five algorithms are evaluated, which are the naive algorithm that blindly follows the prediction (i.e., FTP) and our learning-augmented algorithm with different trust parameters ($\theta = 1, 0.75, 0.5, 0.25$). Recall that when $\theta = 1$, the learning-augmented algorithm is identical to the online algorithm. In Fig. 1(a), we plot the average CR of each algorithm over varied errors μ. We run our experiment on 10^4 demand sequences, with 50% being uniform sequences and 50% being long-tail sequences. The average CR is the average ratio of the cost of the algorithm and that of the optimal offline algorithm over all the sequences. From Fig. 1(a) we can see that while FTP achieves optimality when the prediction is perfect ($\mu = 0$), its performance degrades abruptly when the prediction error increases, especially when $\mu > 0$. For our learning-augmented algorithms ($\theta = 0.75, 0.5, 0.25$), when the prediction error $\mu = 0$, they all outperform the online algorithm ($\theta = 1$), i.e., achieving consistency. When the prediction is perturbed by a large μ (when $\mu \leq -80$ or $\mu \geq 10$), our learning-augmented algorithm does not degrade quickly, i.e., achieving robustness. Furthermore, with different choices of θ, the algorithm provides different trade-off curves between robustness and consistency.



(a) Synthetic dataset (b) Azure dataset

Figure 1: Average CR over varying prediction errors under different settings.

5.2 Real-world Dataset

We consider the traces of the Virtual Machine (VM) workload of Microsoft Azure collected in 2017 (Cortez et al. 2017), which has the average CPU workloads (reported every 5 minutes) for 30 consecutive days. Given the total workload in 5 minutes for multiple VMs, we can generate the demand sequences. To generate the prediction, we train a 3-layer Long Short-term Memory Network for each item. The network receives the last 512 numbers of CPU utilization values as inputs and predicts the current CPU utilization. We use the first 24 days as the training data and the model is trained by Adam optimization. To get the prediction with different quality, the input of the network is perturbed by a constant bias μ. We provide the detailed setting in the supplementary materials.

We select 15 traces as different items and set the single purchase cost as $C_s = 20$ and the combo purchase cost as $C_c = 200$. In Fig. 1(b), we show the average CR of FTP and our learning-augmented algorithm over the last 6 days (i.e., over 1000 trails). We can observe our learning-augmented outperforms the online algorithm when the prediction is good while maintaining a constant average CR when the prediction is bad. Furthermore, our algorithms show promising empirical performance in the real-world dataset.

6 Conclusion

In this paper, we investigated the two-level ski-rental problem dealing with flexible payment options applicable in various real-world scenarios. We developed a 3-competitive deterministic online algorithm. By exploiting the power of ML predictions, we presented a learning-augmented online algorithm, which achieves robustness and consistency. Finally, we applied experimental results to verify our theoretical observations and further concluded the advantages of our learning-augmented online algorithm. For future work, an interesting direction is to study how to adaptively choose the trust parameter θ to achieve the best performance.

References

Ai, L.; Wu, X.; Huang, L.; Huang, L.; Tang, P.; and Li, J. 2014. The multi-shop ski rental problem. In *The 2014 ACM international conference on Measurement and modeling of computer systems*, 463–475. Association for Computing Machinery.

Antoniadis, A.; Coester, C.; Eliás, M.; Polak, A.; and Simon, B. 2021. Learning-augmented dynamic power management with multiple states via new ski rental bounds. *Advances in Neural Information Processing Systems*, 34: 16714–16726.

Bamas, É.; Maggiori, A.; and Svensson, O. 2020. The primal-dual method for learning augmented algorithms. *Advances in Neural Information Processing Systems*, 33: 20083–20094.

Borodin, A.; and El-Yaniv, R. 2005. *Online computation and competitive analysis*. cambridge university press.

Boyar, J.; Favrholdt, L. M.; Kudahl, C.; Larsen, K. S.; and Mikkelsen, J. W. 2017. Online algorithms with advice: A survey. *ACM Computing Surveys (CSUR)*, 50(2): 1–34.

Cortez, E.; Bonde, A.; Muzio, A.; Russinovich, M.; Fontoura, M.; and Bianchini, R. 2017. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, 153–167.

Dooly, D. R.; Goldman, S. A.; and Scott, S. D. 2001. Online analysis of the TCP acknowledgment delay problem. *Journal of the ACM (JACM)*, 48(2): 243–273.

Karlin, A. R.; Manasse, M. S.; McGeoch, L. A.; and Owicki, S. 1994. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6): 542–571.

Karlin, A. R.; Manasse, M. S.; Rudolph, L.; and Sleator, D. D. 1988. Competitive snoopy caching. *Algorithmica*, 3: 79–119.

Khanafer, A.; Kodialam, M.; and Puttaswamy, K. P. 2013. The constrained ski-rental problem and its application to online cloud cost optimization. In *2013 Proceedings IEEE INFOCOM*, 1492–1500. IEEE.

Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; and Zhang, G. 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12): 2346–2363.

Lykouris, T.; and Vassilvtiskii, S. 2018. Competitive Caching with Machine Learned Advice. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 3296–3305. PMLR.

Masdari, M.; and Khoshnevis, A. 2020. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*, 23(4): 2399–2424.

Mitzenmacher, M.; and Vassilvitskii, S. 2022. Algorithms with predictions. *Communications of the ACM*, 65(7): 33–35.

Pan, L.; Wang, L.; Chen, S.; and Liu, F. 2022. Retention-aware container caching for serverless edge computing. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, 1069–1078. IEEE.

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

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR*.

Wang, S.; Li, J.; and Wang, S. 2020. Online algorithms for multi-shop ski rental with machine learned advice. *Advances in Neural Information Processing Systems*, 33: 8150–8160.

Wu, B.; Bao, W.; and Yuan, D. 2021. Competitive Analysis for Two-Level Ski-Rental Problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 12034–12041.

Wu, B.; Bao, W.; Yuan, D.; and Zhou, B. 2022. Competitive Analysis for Multi-Commodity Ski-Rental Problem. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 4672–4678.