Learning-Augmented Online Algorithm for Two-Level Ski-Rental Problem

Keyuan Zhang[1], **Zhongdong Liu**[1], **Nakjung Choi**[2], **Bo Ji**[1]

[1]Virginia Tech, Blacksburg, VA, USA
[2]Nokia Bell Labs, Murray Hill, NJ, USA
keyuanz@vt.edu, zhongdong@vt.edu, nakjung.choi@nokia-bell-labs.com, boji@vt.edu

Abstract

In this paper, we study the two-level ski-rental problem, where a 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 (i.e., rent), buying individual items (i.e., single purchase), and buying all the items (i.e., combo purchase). Without knowing future demands, the user aims to minimize the total cost (i.e., the sum of the rental, single purchase, and combo purchase costs) by balancing the trade-off between the expensive upfront costs (for purchase) and the potential future expenses (for rent). We first design a robust online algorithm (RDTSR) that offers a worst-case performance guarantee. While online algorithms are robust against the worst-case scenarios, they are often overly cautious and thus suffer a poor average performance in typical scenarios. On the other hand, Machine Learning (ML) algorithms typically show promising average performance in various applications but lack worst-case performance guarantees. To harness the benefits of both methods, we develop a learning-augmented algorithm (LADTSR) by integrating ML predictions into the robust online algorithm, which outperforms the robust online algorithm under accurate predictions while ensuring worst-case performance guarantees even when predictions are inaccurate. Finally, we conduct numerical experiments on both synthetic and real-world trace data to corroborate the effectiveness of our approach.

Introduction

Decision-making 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 classic example 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 have been extensively studied and found applications in cloud computing (Khanafer, Kodialam, and Puttaswamy 2013), power management (Antoniadis et al. 2021), serverless edge computing (Pan et al. 2022), etc.

However, the ski-rental problem only involves making a binary decision (i.e., rent or buy for one item only), which makes it inadequate to model more sophisticated scenarios where multiple levels of decisions are involved. Consider cloud computing as an example. An organization may request multiple types of server instances (e.g., HPC instances for complex system simulations, Memory instances for storing data, and GPU instances for AI tasks) from the cloud vendors such as Amazon EC2, Microsoft Azure, and Google Cloud (Miguel 2023). The cloud vendors usually offer multiple pricing options: i) pay-as-you-go: the cost only depends on the demand; ii) instance reservation: paying an upfront cost to cover the demand for that type of instance within a certain period; iii) combo offers: offering increased discounts for reserving multiple types of instances. Hence, making choices wisely among these plans is crucial to reducing expenses on cloud resources. Additionally, similar scenarios are common for individual users. For example, when subscribing to various digital services (e.g., Amazon offers video, music, and games), users often face choices among multiple plans: renting specific content, subscribing to one single service, or subscribing to a combo membership that grants full access to multiple services. Other applications, such as mobile data plans or Bahncard reservations, also offer similar payment options (Wu, Bao, and Yuan 2021).

To model multiple levels of decisions, we consider the two-level ski-rental problem introduced in Wu, Bao, and Yuan (2021). Specifically, it considers scenarios where a user needs to fulfill the demands for multiple items in a given but *unknown* time horizon. Upon the arrival of one demand for some item, the user can cover the demand by choosing one of the three payment options: i) rent, ii) single purchase, and iii) combo purchase. Rent is an on-demand payment with no upfront cost; a single purchase incurs an upfront cost (denoted by C_s) but covers the demands for that specific item since then and onward; a combo purchase has a higher upfront cost (denoted by $C_c \geq C_s$) while covering the demands for all the items till the end. While the ski-rental problem only considers rent and single purchases, a unique challenge in the two-level ski-rental problem lies in the intricate decision-making between single purchases and a combo purchase covering multiple items. To minimize the total cost (i.e., the sum of the rental, single purchase, and combo purchase costs), Wu, Bao, and Yuan (2021) propose online algorithms that make decisions without knowing any future information. *Competitive ratio (CR)* is employed to evaluate the algorithms, which is defined as the ratio be-

tween the cost of the online algorithm and that of the optimal offline algorithm, over all the feasible inputs. While they prove a CR for their deterministic algorithm, the CR can be unbounded when multiple units of demand arrive at the same time (see Remark 1). We propose our robust online algorithm to bridge this gap.

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 useful prediction models. However, they lack robustness guarantees as the predictions can be highly inaccurate due to distribution drift (Lu et al. 2018) or adversarially chosen test data (Szegedy et al. 2014).

In the second part of this work, we first show that a simple algorithm that blindly follows ML predictions cannot guarantee robustness in 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) with an accurate prediction, it offers a better CR than our robust online algorithm *(consistency)* and ii) even if the prediction is inaccurate, it still retains a worst-case guarantee *(robustness)*. Our learning-augmented online algorithm takes the predictions as the black-box oracles without prior knowledge of the prediction quality.

We summarize our main contributions in the following.

First, we propose an online algorithm for the two-level ski-rental problem. We prove that it achieves a CR of $3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s})$, providing a stronger performance guarantee compared to the State-of-the-Art algorithm introduced in Wu, Bao, and Yuan (2021) (see Remark 1).

Second, we design a novel learning-augmented algorithm that integrates ML predictions into robust 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 data and real-world trace data. The results verify our theoretical analyses and corroborate that our learning-augmented algorithm guarantees both consistency and robustness.

Due to space limitations, we omit all the proofs and provide them in our online technical report (Zhang et al. 2023).

Related Work. The ski-rental problem is introduced in Karlin et al. (1988), which gives the best deterministic online algorithm with a CR of 2. Then, an optimal randomized online algorithm achieving $e/(e-1)$-competitiveness is proposed in Karlin et al. (1994) (e is the Euler's number). Several variants of the ski-rental problem are later studied. In the following, we briefly discuss the variants that are most relevant to ours. Fleischer (2001) studies the *Bahncard problem*, where buying a Bahncard provides discounts on subsequent railway tickets for a certain duration. Ai et al. (2014) study the *multi-shop ski-rental problem*, where a user needs to choose when and where to buy from multiple shops with different prices for renting and purchasing. Recently, Wu et al. (2022) study the *multi-commodity ski-rental problem*, where a user requires a bundle of commodities altogether and needs to choose payment options for each commodity.

On the other hand, researchers have designed learning-augmented algorithms for a variety of online optimization problems. In the seminal work (Lykouris and Vassilvtiskii 2018), they show that by incorporating ML predictions, the classic online Marker algorithm can achieve both consistency and robustness for the caching problem. Learning-augmented algorithm design is also studied for the ski-rental problem and its variants (Purohit, Svitkina, and Kumar 2018; Bamas, Maggiori, and Svensson 2020; Wang, Li, and Wang 2020). For example, both single and multiple ML predictions are incorporated into the online algorithm for the multi-shop ski-rental problem (Wang, Li, and Wang 2020). We refer interested readers to the surveys (Boyar et al. 2017; Mitzenmacher and Vassilvitskii 2022) for comprehensive coverage. In our work, however, the combo purchase can cover all the single purchases and thus result in coupled decisions. This "two-level" payment option is more practical in quite a few real-world applications but introduces unique challenges in the algorithm design.

System Model and Problem Formulation

System Model. We consider a time-slotted model with time index $t = 1, 2, \ldots, T$, where T is the length of a finite time horizon *unknown* to the user (i.e., the decision maker). The user needs to fulfill demands for K items arriving over time. Upon the arrival of demands in each time-slot, the user can fulfill them by choosing from different options (rent or purchase). Without loss of generality, we assume that the demands arriving in the same time-slot are for one item only.

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 and $a(t)$ is a non-negative integer representing the amount of the demand. In particular, $d(t) = (0, 0)$ indicates no demand in time-slot t. Let $\mathbf{D} := \{d(t)\}_{t=1}^{T}$ denote the sequence of demands 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) rent, ii) single purchase, and iii) combo purchase.

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

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

iii) **Combo purchase**: The user immediately pays a higher fixed cost $C_c \in (C_s, K \cdot C_s)$, which covers the demands for all the items from time t to T.

We assume that both C_s and C_c are integers and that once the payment is made in some time-slot t, the associated demands can be fulfilled immediately. While choosing to rent may offer a lower upfront cost, it may result in higher ex-

penses due to future demands. In contrast, choosing to purchase (either single or combo) incurs a higher upfront cost, but it covers the potential future demands.

Problem Formulation. Let $r(t)$ be the binary rental decision in time-slot t, where $r(t) = 1$ if the user decides to rent, otherwise $r(t) = 0$. Let t_k denote the time when the user makes a single purchase for item k; similarly, let t_c denote the time when the user makes a combo purchase. We consider an *online* algorithm π that determines $\{r^\pi(t)$ for $t = 1, 2, \ldots, T, \{t_k^\pi\}_{k=1}^K, t_c^\pi\}$ only using information available in time-slot t: the rental decision history $\{r^\pi(\tau)\}_{\tau=1}^{t-1}$, single purchase history $\{t_k^\pi\}_{k=1}^K$, combo purchase history t_c^π, demand history $\{d(\tau)\}_{\tau=1}^t$, number of instances K, single purchase price C_s, and combo purchase price C_c; no future information about the demand arrival is available to the algorithm. 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 algorithm π:

$$\mathcal{C}(\mathbf{D}, \pi) := \sum_{t=1}^{T} a(t)r(t) + \sum_{k=1}^{K} C_s \mathbb{1}_{\{t_k \le T\}} + C_c \mathbb{1}_{\{t_c \le T\}}, \tag{1}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function and the three terms correspond to the total rental cost, total single purchase cost, and combo purchase cost, respectively. Given a demand sequence $\mathbf{D}$, the objective is to minimize the total cost:

$$\min_{r(t), t_k, t_c} \quad \mathcal{C}(\mathbf{D}, \pi) \tag{2a}$$

$$\text{subject to} \quad r(t) + \mathbb{1}_{\{t \ge t_{i(t)}\}} + \mathbb{1}_{\{t \ge t_c\}} \ge 1, \quad t \in \{1, 2, \ldots, T\} \tag{2b}$$

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

$$t_k > 0, \quad k \in \{1, 2, \ldots, K\} \tag{2d}$$

$$t_c > 0, \tag{2e}$$

where Contraint (2b) requires that each demand must be fulfilled via either renting or purchasing (single or combo).

Competitive Ratio. Without knowledge of future demand, it is usually difficult for an online algorithm to attain the same minimum total cost achieved by an optimal offline algorithm, which requires 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), defined as the ratio between the cost of the online algorithm and that of the optimal offline algorithm over all the feasible inputs. Formally, an online algorithm π is called c-competitive if there exists a constant $c \ge 1$ such that for any demand sequence $\mathbf{D}$, there is

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

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

Robust Online Algorithm

In this section, we introduce our robust online algorithm and prove its competitive ratio. To begin with, we first present two important notions that will be used in our algorithm: indicative cost and purchase threshold.

Indicative Cost and Purchase Threshold. We define the indicative cost $\psi_k(t)$ for item k in time-slot t as the total demand for item k during the interval $[1, t]$ that is not covered by any purchase. Then, indicative cost $\psi_k(t)$ evolves as

$$\psi_k(t) := \begin{cases} \psi_k(t-1) + a(t), & i(t) = k \text{ and} \\ & t < \min\{t_k, t_c\}, \\ \psi_k(t-1), & \text{otherwise}, \end{cases} \tag{4}$$

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 is not covered by (either single or combo) purchases (i.e., $t < \min\{t_k, t_c\}$); otherwise, it remains unchanged. Note that $\psi_k(0) = 0$ for all k. Let $\lambda_s \in (1, C_s]$ be the threshold for making single purchases. That is, if the single purchase indicator $\psi_k(t)$ reaches λ_s (i.e., $\psi_k(t) \ge \lambda_s$), we will make a single purchase for item k. We assume the same threshold λ_s for all items because they have the same single purchase price C_s. This is also an assumption made in Wu, Bao, and Yuan (2021). While our algorithm can potentially be adapted to address item-specific purchase costs and thresholds, a more sophisticated competitive analysis is expected.

Similarly, let $\psi_c(t)$ and $\lambda_c \in (1, C_c]$ be the overall indicative cost $\psi_c(t)$ and the combo purchase threshold, respectively. We define the overall indicative cost $\psi_c(t)$ as

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

When $\psi_c(t)$ reaches the combo purchase threshold λ_c (i.e., $\psi_c(t) \ge \lambda_c$), a combo purchase will be made. We assume that thresholds λ_s and λ_c are integers due to the integral demand we consider.

Intuitively, when indicative cost $\psi_k(t)$ reaches the single purchase threshold λ_s, it reflects a substantial demand for item k, leading to a single purchase of item k. Similarly, when the overall indicative cost $\psi_c(t)$ reaches λ_c, it indicates substantial demand for multiple items, leading to a combo purchase rather than single purchases or rent.

Algorithm Description. We now present our *Robust Deterministic Two-level Ski-rental (RDTSR)* Algorithm in Algorithm 1 and explain how it works in the following. When a new demand $d(t)$ arrives in time-slot t, it first updates the indicative costs $\psi_k(t)$ and $\psi_c(t)$ according to Eqs. (4) and (5), respectively (Lines 2-3). If the demand is already covered by a previous (single or combo) purchase (Line 4), it does nothing. Otherwise, if the indicative cost exceeds the threshold, it will purchase by prioritizing the combo purchase (Lines 7-10); if not, it will cover the demand via renting (Line 12).

Competitive Analysis. We derive the CR of RDTSR as a function of thresholds λ_s and λ_c. By choosing proper values of λ_s and λ_c, we show that the CR is upper bounded by 3.

Theorem 1. *The CR of RDTSR is upper bounded by*

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

In particular, this upper bound is achieved by choosing thresholds $\lambda_s = C_s$ and $\lambda_c = C_c$.

Algorithm 1: Robust Deterministic Two-level Ski-rental (RDTSR) Algorithm

Input : $K, C_s, C_c, \lambda_s, \lambda_c, \mathbf{D}$ (revealed in an online manner)
Output: $r(t), t_k, t_c$
Init. : $\psi_c(t), \psi_k(t), r(t) \leftarrow 0, t_k, t_c \leftarrow \infty$

```
while new demand d(t) arrives do
    Update ψ_k(t) according to Eq. (4);
    Update ψ_c(t) according to Eq. (5);
    if d(t) is covered by a previous purchase then
        Do nothing and wait till next time-slot;
    else
        if ψ_c(t) ≥ λ_c then
            t_c ← t; // combo purchase
        else if ψ_{i(t)}(t) ≥ λ_s then
            t_{i(t)} ← t; // single purchase
        else
            r(t) ← 1; // rent
        end
    end
end
```

Proof sketch. The proof has two key steps: 1) given any λ_s and λ_c, we derive an upper bound of the CR, which is a function of λ_s and λ_c; 2) we show that the CR is at most $3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s})$ when $\lambda_s = C_s$ and $\lambda_c = C_c$.

Note that it is typically challenging to directly analyze the performance for an arbitrary demand sequence. Therefore, we break Step 1) into three substeps: 1a) We derive an upper bound of CR over all the demand sequences with the same total demand. The total demand is a vector consisting of K elements where each element represents the total demand for the associated item. 1b) Following a similar line of analysis in Wu, Bao, and Yuan (2021), we define a standard total demand to simplify the upper bound we obtained in Step 1a). 1c) We obtain an upper bound of CR over all the standard total demand, which is $\max\{\frac{\lambda_s - 1 + C_s}{\lambda_s}, \frac{\lambda_c - 1 + C_c + (\lambda_c - 1)\lambda_s^{-1}(C_s - 1)}{\min\{C_c, \lambda_c\}}\}$. In Step 2), we minimize the function of λ_s and λ_c derived in Step 1c). □

Remark 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 construct a simple example to show that the CR of their algorithm can be unbounded when allowing multiple units of demand to arrive in a time-slot (i.e., $a(t) > 1$). Specifically, consider a demand sequence $\tilde{\mathbf{D}} := \{(1, C_s), (2, C_s), \ldots, (K, C_s)\}$, i.e., C_s units of demand for item k arrive in time-slot k for all k. In their algorithm, single purchases will be made for all the items, while the combo purchase will never be made as 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 arbitrarily large when $KC_s \gg C_c$. We address this issue by redefining the indicative cost and assigning the highest priority to the combo purchase. It is also noteworthy that even for the setting with unit demand arrival (i.e., $a(t) = 1$), our algorithm still has a slightly improved upper bound $(3 - \frac{1}{C_s} - \frac{1}{C_c}(2 - \frac{1}{C_s})$ vs. $(3 - \frac{1}{C_s}))$.*

Learning-augmented Online Algorithm

In the previous section, we introduce a robust online algorithm, RDTSR, with a worst-case guarantee. However, online algorithms are often overly conservative and may have a poor average performance in real-world scenarios. In contrast, ML algorithms have the advantage of utilizing extensive historical data and building well-trained models, which enables them to achieve a promising average performance. Nonetheless, ML algorithms lack a worst-case guarantee when facing adversaries, outliers, distribution shifts, etc. To harness the benefits of both methods, we design a learning-augmented online algorithm that achieves both consistency and robustness, two commonly used notions in the literature.

Machine Learning Predictions

We consider the case where an ML algorithm provides a prediction $y := (y_1, \ldots, y_K)$ that represents the predicted total demand for each item. Predictions of the total demand are commonly used in real-world applications. For example, in cloud computing, various resources such as CPU, memory, and network bandwidth can be fed into a predictor to forecast future workload (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's are available). The actual total demand can be represented by a vector $z := (z_1, \ldots, z_K)$, where z_k denotes the total demand for item k, i.e., $z_k = \sum_{t=1}^{T} \mathbb{1}_{\{i(t)=k\}} a(t)$. The prediction error is defined as the $\ell 1$-norm distance from the actual total demand to the predicted ones, and we use $\eta_k := |y_k - z_k|$ and $\eta := \sum_{k=1}^{K} \eta_k$ to denote the prediction error for item k and the total prediction error, respectively.

Consider a learning-augmented online algorithm $\pi^\dagger$ that generates a solution with cost $\mathcal{C}(\mathbf{D}, \pi^\dagger)$ using information available in time-slot t and a prediction y. Algorithm $\pi^\dagger$ is called α-*consistent* if $\mathcal{C}(\mathbf{D}, \pi^\dagger) \leq \alpha \cdot OPT(\mathbf{D})$ for any sequence $\mathbf{D}$ when the prediction is perfect (i.e., $\eta = 0$), and it is called β-*robust* if $\mathcal{C}(\mathbf{D}, \pi^\dagger) \leq \beta \cdot OPT(\mathbf{D})$ for any sequence $\mathbf{D}$, regardless of the prediction error η. Here, consistency measures the performance in the best-case scenario with perfect predictions, while robustness measures the performance of a learning-augmented algorithm in the worst-case scenario regardless of the prediction quality.

A Simple Algorithm: Follow the Prediction

First, we show that a simple algorithm called *Follow the Prediction (FTP)*, cannot guarantee robustness. FTP operates in the following way: it compares the combo purchase cost C_c with the total cost without the combo purchase option (i.e., $\sum_{k=1}^{K} \min\{C_s, y_k\}$), and makes a combo purchase if it is lower (i.e., $\sum_{k=1}^{K} \min\{C_s, y_k\} \geq C_c$); otherwise, it makes a single purchase for item k if $y_k \geq C_s$ or rents for item k when needed. Lemma 1 gives an upper bound on the total cost under FTP.

Lemma 1. *FTP satisfies* $\mathcal{C}(\mathbf{D}, \mathrm{FTP}) \leq OPT(\mathbf{D}) + \eta$.

Proof sketch. We first show that the cost of both FTP and the optimal offline algorithm only depends on total demand. Then, we only need to analyze the CR based on the total demand. We use $ALG(z)$ and $OPT(z)$ to denote the cost of FTP and the optimal offline algorithm under a total demand z, respectively. Depending on whether FTP and the optimal offline algorithm each make a combo purchase or not, we consider four cases. Finally, we show $ALG(z) \leq OPT(z) + \eta$ for each case. □

Remark 2. *When the prediction is perfect (i.e.,* $\eta = 0$*), FTP achieves the optimal cost. However, the performance of FTP can be very poor when the prediction error* η *is large. For example, when* $\sum_{k=1}^{K} \min\{C_s, y_k\} < C_c$ *and* $y_k < C_s$*, but there is a very large total demand for item* k*, FTP would choose to rent for item* k *at all times, resulting in an unbounded cost and thus an unbounded CR as the optimal cost is bounded by* C_c*. Therefore, although FTP can achieve consistency, it lacks a robustness guarantee.*

Design of Learning-augmented Algorithm

As following the prediction in a straightforward way (e.g., FTP) cannot guarantee the worst-case performance, we aim to wisely integrate the ML predictions with RDTSR and design a learning-augmented online algorithm that can achieve both consistency and robustness.

The main idea of the design is to adjust the threshold of RDTSR based on the prediction: if a prediction suggests a (single or combo) purchase, then we should decrease the associated threshold to encourage the purchase (recall that the threshold determines when to make a purchase); if a prediction suggests not making a purchase, we should increase the associated threshold. The extent to which we adjust the threshold relies on our confidence in the prediction.

Based on the above intuition, we design the *Learning-augmented Deterministic Two-level Ski-rental (LADTSR)* Algorithm, presented in Algorithm 2. It is almost the same as RDTSR, except for the choice of thresholds. First, we specify a trust parameter $\theta \in [0, 1]$, reflecting the confidence in the prediction: a smaller θ indicates higher confidence. Then, we adjust the thresholds as follows: for each single purchase threshold $\lambda_{s,k}$ for item k, we reduce it to θC_s if the prediction suggests a single purchase for this item (i.e., $y_k \geq C_s$); otherwise, we increase $\lambda_{s,k}$ to C_s/θ. For the combo purchase threshold λ_c, we reduce it to $\theta^2 C_c$ (see Remark 5 for an explanation of this choice instead of θC_c) if the prediction suggests a combo purchase (i.e., $\sum_{k=1}^{K} \min\{C_s, y_k\} \geq C_c$); otherwise, we increase λ_c to C_c/θ. We set the threshold to infinity in case of division by 0. Note that when running RDTSR (Line 13), the definition of $\psi_c(t)$ (i.e., Eq. (5)) needs to be modified slightly by replacing λ_s with $\lambda_{s,k}$.

Algorithm 2: Learning-augmented Deterministic Two-level Ski-rental (LADTSR) Algorithm

Input : $y, \theta, K, C_s, C_c, \mathbf{D}$ (revealed in an online manner)
Output: $r(t), t_k, t_c$
Init. : $\psi_c(t), \psi_k(t), r(t) \leftarrow 0, t_k, t_c \leftarrow \infty$

1 **for** *item* $k = \{1, 2, \ldots, K\}$ **do**
2 **if** $y_k \geq C_s$ **then**
3 $\lambda_{s,k} \leftarrow \theta C_s$;
4 **else**
5 $\lambda_{s,k} \leftarrow C_s/\theta$;
6 **end**
7 **end**
8 **if** $\sum_{k=1}^{K} \min\{C_s, y_k\} \geq C_c$ **then**
9 $\lambda_c \leftarrow \theta^2 C_c$;
10 **else**
11 $\lambda_c \leftarrow C_c/\theta$;
12 **end**
13 Run RDTSR with thresholds $\lambda_{s,k}$ and λ_c;

Analysis of Learning-augmented Algorithm

In this subsection, we focus on the consistency and robustness analysis of LADTSR with $\theta \in (0, 1)$. The special cases of LADTSR with $\theta = 0$ and $\theta = 1$ correspond to FTP and RDTSR, respectively. It is noteworthy that by choosing different values of θ, LADTSR exhibits a crucial trade-off between consistency and robustness.

Theorem 2. *The CR of LADTSR is upper bounded by*

$$\min\left\{1 + \theta + \theta^2 + \frac{1 + 2\theta}{1 - \theta} \cdot \frac{\eta}{OPT(\mathbf{D})}, 1 + \theta^{-1} + \theta^{-3}\right\}, \tag{7}$$

for $\theta \in (0, 1)$*. Futhermore, LADTSR is* $(1 + \theta + \theta^2)$*-consistent and* $(1 + \theta^{-1} + \theta^{-3})$*-robust.*

Proof sketch. We begin with the proof of the first bound $1 + \theta + \theta^2 + \frac{1+2\theta}{1-\theta} \cdot \frac{\eta}{OPT(\mathbf{D})}$. Our goal is to show that this upper bound holds for any demand sequence and any prediction. First, we consider two cases for the choice of λ_c (i.e., $\lambda_c = \theta^2 C_c$ and $\lambda_c = C_c/\theta$). In each case, we consider four subcases, depending on whether LADTSR and the optimal offline algorithm each make a combo purchase or not. In each subcase, we derive an upper bound of CR, which is a function of η and $OPT(\mathbf{D})$. Combining all eight subcases yields the first bound.

Next, we prove the second bound $1+\theta^{-1}+\theta^{-3}$. The analysis is similar to Step 1) in the proof of Theorem 1 and also has three steps: 1a) Given any prediction, we derive an upper bound of CR over all the demand sequences with the same total demand. 1b) We simplify the upper bound obtained in Step 1a) through standard total demand. 1c) We show that the upper bound derived in 1b) is at most $1+\theta^{-1}+\theta^{-3}$. Note that we need a generalized standard total demand compared to the proof of Theorem 1 because LADTSR has two different values of single purchase threshold (i.e., θC_s and C_s/θ), while RDTSR only has one threshold value (i.e., λ_s). □

Remark 3. *With any* $\theta \in (0, 1)$*, the CR of LADTSR is at most* $1+\theta^{-1}+\theta^{-3}$*, regardless of the prediction error* η*, implying the worst-case performance guarantees (i.e., robustness). In particular, if LADTSR does not trust the prediction*

at all (i.e., $\theta \to 1$), then we have $1+\theta^{-1}+\theta^{-3} \to 3$, achieving a similar performance guarantee to that of RDTSR.

Remark 4. *When the prediction is perfect (i.e., $\eta = 0$), the CR of LADTSR is at most $1 + \theta + \theta^2$ as $1 + \theta + \theta^2 < 1 + \theta^{-1} + \theta^{-3}$ for $\theta \in (0, 1)$, implying $(1 + \theta + \theta^2)$-consistency. In particular, if LADTSR fully trusts the prediction (i.e., $\theta \to 0$), then we have $1 + \theta + \theta^2 \to 1$, achieving a similar performance to the optimal offline algorithm.*

Remark 5. *We now explain the reason why we set the combo purchase threshold λ_c as $\theta^2 C_c$ instead of θC_c when the prediction suggests a combo purchase. Specifically, we present an example to show that LADTSR fails to achieve 1-consistency when $\theta \to 0$ if we choose $\lambda_c = \theta C_c$. Consider $C_c = (K-1)C_s + 1$ and a demand sequence $\tilde{\mathbf{D}} := \{(1, C_s), (2, C_s), \ldots, (K, C_s)\}$. If the prediction is perfect (i.e., $y = (C_s, \ldots, C_s)$), then LADTSR sets $\lambda_{s,k} = \theta C_s$ for all the items since $y_k \geq C_s$ (in fact, $y_k = C_s$) and $\lambda_c = \theta C_c$ since $\sum_{k=1}^{K} \min\{C_s, y_k\} = KC_s > C_c$. When the demand (k, C_s) arrives, according to Eqs. (4) and (5), $\psi_k(k)$ increases by C_s, and $\psi_c(k)$ increases by $\min\{\psi_k(k), \lambda_{s,k}\} = \theta C_s$ (recall that we modify Eq. (5) slightly for LADTSR). So in each time-slot t, the overall indicative cost is $\psi_c(t) = \sum_{k=1}^{K} \min\{\psi_k(t), \lambda_{s,k}\} = t\theta C_s$, and it reaches λ_c only when $t = K$ because $\psi_c(K-1) = (K-1)\theta C_s < \theta(K-1)C_s + \theta = \theta C_c = \lambda_c < K\theta C_s = \psi_c(K)$. That is, LADTSR will make a combo purchase when the last demand arrives; for the previous $K-1$ demands, LADTSR will make single purchases for these items as their demands exceed the single purchase threshold. As a result, the total cost is $(K-1)C_s + C_c$. Since the optimal offline cost is $C_c = (K-1)C_s + 1$ (by assumption), the CR of LADTSR is at least $\frac{(K-1)C_s + C_c}{C_c} = \frac{C_c - 1 + C_c}{C_c} = 2 - \frac{1}{C_c}$, which is larger than 1 for any θ.*

Experimental Results

In this section, we conduct numerical experiments using both synthetic and real-world trace data to evaluate the performance of our proposed algorithms (RDTSR and LADTSR).[1] The results demonstrate the robustness of RDTSR and corroborate that LADTSR offers the desired trade-off between consistency and robustness.

Datasets

Synthetic Dataset. We consider the same settings adopted in Wu, Bao, and Yuan (2021). Specifically, the demand sequence $\mathbf{D}$ is generated in the following way: First, the time horizon T follows a uniform distribution ranging from 1 to 60. The demand in each time-slot is randomly assigned to one item, following two standard distributions: i) **Uniform:** the demand is uniformly distributed among all the items; ii) **Long-tailed:** the demand distribution follows the Pareto law, i.e., 80% of demands are assigned to 20% of items. The theoretical results in Wu, Bao, and Yuan (2021) hold only for unit-demands (i.e., $a(t) = 1$ in all time-slots). In our experiments, we also consider the case where multiple units of demand may arrive in each time-slot (i.e., $a(t) \geq 1$). We set the number of items $K = 6$.

Cloud Cost Management Dataset. We consider the application of cloud service cost management, where a user needs to fulfill the demands from multiple servers via either rental or purchase (single or combo) options. To simulate the demands from multiple servers, we use the traces of Virtual Machine (VM) workloads from Microsoft Azure (Cortez et al. 2017). These traces include the average CPU workload every 5 minutes for 30 consecutive days. We randomly shuffle the workloads from multiple servers to generate the demand sequence $\mathbf{D}$. The number of VMs is 9 (i.e., $K = 9$).

Mobile App Usage Dataset. We consider the application where a user needs to pay for the data usage from multiple mobile Apps. The data usage traces are provided by Yu et al. (2018), which records the user ID, App ID, timestamp, and data traffic. We use the trace for multiple Apps from one user in one day as the demand sequence for one problem instance. The number of Apps is 4 (i.e., $K = 4$).

Unless otherwise specified, we set the single purchase price $C_s = 9$ and the combo purchase price $C_c = 30$ for all the three datasets we consider.

Robust Online Algorithms

In this subsection, we compare RDTSR with the *Deterministic Two-level Ski-Rental (DTSR)* algorithm proposed in Wu, Bao, and Yuan (2021).

Experimental Setup. We vary the value of the combo purchase price C_c from 15 to 40. For each value of C_c, we run our experiments for 10^4 independent demand sequences, with 40% being the uniform sequences and 60% being the long-tailed sequences. All the experiments are implemented in Python and are conducted on a laptop with a 12th Gen Intel(R) i5-12500H processor and 16GB memory.

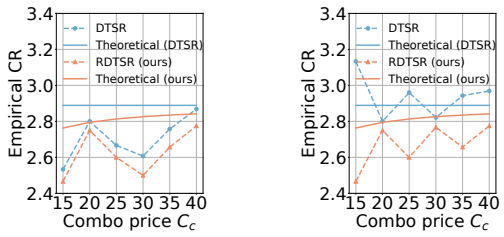
Results. Fig. 1 shows the empirical and theoretical CR of RDTSR and DTSR. Here, the empirical CR represents the worst cost ratio (under the online algorithm and the optimal offline algorithm) over 10^4 independent demand sequences. We can observe that our algorithm (RDTSR) exhibits a lower empirical CR compared to the existing algorithm (DTSR). Furthermore, the empirical CR of RDTSR consistently remains below its theoretical bound, verifying our theoretical results. In the setting of multi-unit demands, the empirical CR of DTSR exceeds the theoretical bound (e.g., when $C_c = 15, 25, 35, 40$ in Fig. 1b). This suggests that the CR of DTSR obtained in Wu, Bao, and Yuan (2021) does not hold when allowing multiple units of demand to arrive in a time-slot (see Remark 1 for a detailed discussion).

Similar results can be observed from the two real-world datasets. Furthermore, the simulation results also show that RDTSR and DTSR exhibit similar average performance. Due to space limitations, we provide the results in our online technical report (Zhang et al. 2023).

Learning-augmented Online Algorithms

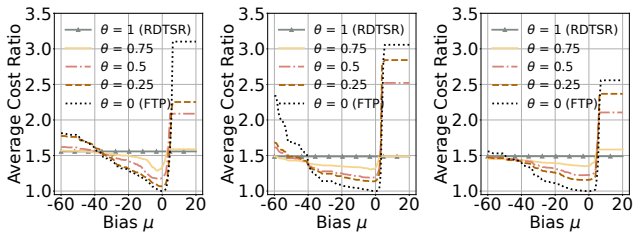
In this subsection, we study the performance of LADTSR under varying prediction errors. We first explain how to gen-

[1] Source code: https://github.com/nauyek/LADTSR.



(a) Unit demands (b) Multi-unit demands

Figure 1: Empirical CR on the synthetic dataset.



(a) Synthetic dataset (b) Azure dataset (c) AppUsage dataset

Figure 2: Average cost ratio under varying prediction errors.

erate ML predictions in our experiments.

ML Prediction Generation. Recall that LADTSR receives the total demand for each item as a prediction. We construct an ML task that uses historical total demand information to predict the next total demand for one item. Suppose that there are N rounds of two-level ski-rental problem instances, and for each round n, we have a total demand z_k^n for item k. Then, we can use a sequence of historical total demand with length $l-1$ (i.e., $z_k^{n-1}, z_k^{n-2}, \ldots, z_k^{n-l+1}$) to predict z_k^n. To generate a sequence of total demand for each item, for the Synthetic dataset, we generate multiple problem instances (40% uniform sequences followed by 60% long-tailed sequences) and then concatenate them together; such a sequence of total demands for each item can be directly obtained in the two real-world datasets. Then, we use a sliding window to generate multiple sequences of total demand for training and testing. The window size l is 10, 513, and 7 for the Synthetic, Azure, and AppUsage datasets, respectively. The last total demand is the ground truth while the previous sequence of total demand is the historical data. The window size is specifically chosen for each dataset due to the performance of the models. Finally, we use 80% of the sequences for training. The Mean Average Percentage Error (MAPE) of our ML models during the test is at most 5%.

To generate predictions with varying qualities, we first train the ML models without any perturbation. Then during the test, we perturb the input of the network by adding a bias μ that is uniform across the whole sequence of input. This represents the scenario where the prediction is biased due to the distribution shift between training and testing data.

Experimental Setup. We train a *Long Short-term Memory (LSTM)* network to predict the total demand for each item. The network has two LSTM layers followed by one fully connected layer. The hidden states of the two LSTM layers are both 10, 256, and 10 for the Synthetic, Azure, and AppUsage datasets, respectively. We use the mean absolute error as the loss function and employ the Adam optimizer to train the weights. The model is implemented in TensorFlow. For each model, the training process takes about 2 minutes on an NVIDIA GeForce RTX 3060 Laptop GPU.

To study the impact of trust parameter θ, we evaluate the performance of LADTSR with different values of $\theta \in \{0, 0.25, 0.5, 0.75, 1\}$. Recall that LADTSR with $\theta = 0$ and $\theta = 1$ corresponds to FTP and RDTSR, respectively.

Results. Fig. 2 shows the average cost ratio of LADTSR with different values of θ under varying prediction errors (corresponding to varying values of bias μ). Here the average cost ratio represents the average ratio of cost (under the online algorithm and the optimal offline algorithm) over all the problem instances. Fig. 2a shows that while FTP (i.e., LADTSR with $\theta = 0$) performs quite well when the prediction is accurate (i.e., $\mu = 0$), its performance degrades significantly when the bias increases. For LADTSR with $\theta \in \{0.25, 0.5, 0.75\}$, when the predictions are accurate, they all outperform RDTSR (i.e., LADTSR with $\theta = 1$), achieving better average performance compared to the robust online algorithm without using any predictions. When the prediction is perturbed by a large bias μ ($\mu \leq -40$ or $\mu \geq 10$), the performance degrades, but not significantly compared to FTP. Furthermore, with different values of θ, LADTSR provides different trade-off curves for consistency and robustness. It is noteworthy that the performance of our learning-augmented algorithms degrades abruptly when the actual demand is overestimated ($\mu > 0$), while at a slower rate when it is underestimated. This happens because overestimation leads to unnecessary (single or combo) purchases, resulting in an immediate rise in the cost ratio. On the other hand, underestimation incurs additional rental costs before the algorithm makes a purchase, rising gradually as the bias level $|\mu|$ increases.

Conclusion

Motivated by real-world applications, we investigated the two-level ski-rental problem with multiple payment options and designed a 3-competitive deterministic online algorithm called RDTSR. By integrating useful ML predictions with RDTSR, we proposed a learning-augmented online algorithm called LADTSR, which achieves both consistency and robustness. As for future work, one important direction is to obtain a non-trivial competitive ratio lower bound for the studied problem as well as the optimal trade-off for the learning-augmented algorithm design; another interesting direction is to design learning-augmented algorithms that can achieve better performance by adaptively choosing the trust parameter θ; it would also be nice to design algorithms that account for the asymmetric impact of overestimation and underestimation in predictions.

Acknowledgments

This work is supported in part by the Commonwealth Cyber Initiative (CCI).

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.

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.

Fleischer, R. 2001. On the Bahncard problem. *Theoretical Computer Science*, 268(1): 161–174.

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*, 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.

Miguel, P. G. 2023. Guide to the 23 Best Cloud Service Providers in 2023. https://thectoclub.com/tools/best-cloud-service-providers. Accessed: 2023-12-19.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13): 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.

Yu, D.; Li, Y.; Xu, F.; Zhang, P.; and Kostakos, V. 2018. Smartphone app usage prediction using points of interest. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4): 174.

Zhang, K.; Liu, Z.; Choi, N.; and Ji, B. 2023. Learning-augmented Online Algorithm for Two-level Ski-rental Problem. https://nauyek.github.io/papers/AAAI24TR.pdf. Accessed: 2024-01-15.