Competitive Analysis for Two-Level Ski-Rental Problem

Binghan Wu, Wei Bao, Dong Yuan
Faculty of Engineering, The University of Sydney
biwu6051@uni.sydney.edu.au, wei.bao@sydney.edu.au, dong.yuan@sydney.edu.au

Abstract

In this paper, we study a two-level ski-rental problem. There are multiple commodities, each one can be "rented" (paying for on-demand usage) or "purchased" (paying for life-time usage). There is also a combo purchase available so that all commodities can be purchased as a combo. Since the usages of the commodities in future are not known in advance, to minimize the overall cost, we design an online algorithm to decide if we rent a commodity, purchase a commodity, or make a combo purchase. We first propose a deterministic online algorithm. It can achieve 3 competitive ratio, which is optimal and tight. Next, we further propose a randomized online algorithm, leading to a $\frac{e^\sigma}{e^\sigma-1}$ competitive ratio, where σ is the ratio between the price of a single commodity and the price of combo purchase. Finally, we apply simulation to verify the theoretical competitive ratios and evaluate the actual performance against benchmarks.

Introduction

The ski-rental problem (Karlin, Kenyon, and Randall 2003) is the dilemma in which a customer trades off between two payment options without the knowledge of future: *rental*, i.e., paying for on-demand usage, and *purchase*, i.e., paying for life-time usage at once. Literature has derived the optimal online algorithms with minimized competitive ratio and it has been applied to different fields such as transportation system (Karlin, Kenyon, and Randall 2003), cloud systems (Yang, Pan, and Liu 2019), and energy systems (Lee et al. 2017). However, existing work cannot address many real-world problems, when options are more complex.

In this paper, we consider the *two-level ski-rental problem* (TLSR) when a decision-maker need to trade off between rental and purchase of multiple instances. (We use the term instance instead of commodity in the rest of the paper as users can purchase intangible service, such as mobile data plans and bus tickets.) For each instance, the decision-maker can rent it or purchase it (which is called single purchase in this paper). In addition, the decision-maker can purchase all instances as a *combo purchase* at a reduced cost. Please note that this problem is equivalent to multiple independent classic ski-rental problems if we ignore the combo purchase. Nevertheless, the combo purchase covers all individual purchases, forming a two-level hierarchy. This brings new challenges to decision-markers as payment options are now correlated, and cannot be addressed by existing solutions.

We present a few real-world examples of the TLSR problem:

1. Mobile data plan reservation with sponsored data. A user uses multiple apps on a mobile phone, and these apps generate data traffic. To cover the cost of data usage, the user has to choose among different payment options provided by the Internet Service Provider (ISP) and Content Providers (CPs). Three payment methods are available: 1) Pay-as-you-go (PAYG). There is no upfront fee and each MB of data volume incurs a constant fee. 2) Sponsored data plan (SDP). There is an upfront fee at the beginning and then there is no more fee on the data generated by a specific app in one month. 3) All-in-one plan (AIP). There is a (much) higher upfront fee at the beginning and then there is no more fee on the data generated by all apps in one month. For this problem, apps can be viewed as instances. PAYG can be viewed as rent. The SDP and AIP can be regarded as single purchase and combo purchase respectively.

2. Bahncard reservation with multiple zones. A traveller frequently takes subways in a city with multiple zones. Three payment methods are offered: 1) Pay-as-you-go (PAYG). There is no upfront fee and each travel incurs a cost. 2) Zone pass. There is an upfront fee and then travels within one zone will be free of charge in one month. 3) City pass. There is an upfront fee and then travels within all zones in the city will be free of charge in one month. In this example, zone pass and city pass can be regarded as single purchase and combo purchase respectively.

The bahncard/mobile plan in the real world is usually valid for each calendar month, and they expire at the end of each calendar month. Therefore, in two different calendar months, the decisions are independent. The decision on bahncard/mobile plan purchase can be regarded as an individual two-level ski-rental problem in each calendar month.

Our Contributions. In this paper, we study the TLSR problem, in which rental, single purchases, and the combo purchase are available. We propose an optimal deterministic online algorithm with two fixed thresholds. When the rental cost of a single instance (resp. all instances) exceeds the threshold, the single purchase (resp. combo purchase) will

be made. We claim an exact optimal and tight competitive ratio of the deterministic algorithm which is asymptotically close to 3, and the optimal thresholds are derived. In addition, we further propose a randomized online algorithm, in which the thresholds are generated with our designed probability distributions. Our competitive analysis shows that the competitive ratio of the randomized online algorithm is $\frac{e^\sigma}{e^\sigma-1}$, where σ is the ratio of prices between the single purchase and the combo purchase. This is a neat extension of the competitive ratio $\frac{e}{e-1}$ of the classic ski-rental problem.

Related Work

The classic ski-rental problem (Karlin, Kenyon, and Randall 2003; Epstein and Zebedat-Haider 2015) is the dilemma between rent and purchase without the knowledge of future (i.e., the future is adversarial). It has been studied by existing literature. This problem is also known as the Bahncard problem in a public transportation system (Fleischer 2001). There are many studies on the extended versions of the ski-rental problem in more complicated scenarios. For example, ski-rental problems with non-linear cost functions were studied by (Lotker, Patt-Shamir, and Rawitz 2012; Shi et al. 2018; Fujiwara et al. 2020). (Feldkord, Markarian, and Meyer Auf der Heide 2017) allowed the rental price to float with time. (Meyerson 2005) considered that the choices of purchase are limited in time, and (Hu et al. 2015) further studied a two-dimension venison of the problem. (Chen and Xu 2018) considered the ski-rental problem in a risk-reward model assuming that the decision-maker is willing to accept some degree of uncertainty. (Khanafer, Kodialam, and Puttaswamy 2013) assumed the user knows some information about the adversary. They constrained the adversary with the first or second moments. (Wang, Li, and Wang 2020; Kodialam 2019) also considered the predictions by machine-learning approaches. Some other relevant studies extended the dimension of some parts of the ski-rental model. (Ai et al. 2014; Wang, Li, and Wang 2020) considered multiple shops providing different prices, and the customer has to stick with one shop to rent or buy. (Zhang, Poon, and Xu 2011) generalized the problem by providing n discount options, and each option has an expiration date. The longer the plan reserved, the greater the discount. (Zang et al. 2019) studied a variant of ski-rental problem offering three purchase options including two single purchases and one combo purchase. (Zhang and Conitzer 2020) considers the ski-rental problem with multiple desired resources. The cost of requiring a new resource is a submodular function of the set of the resources that have not been purchased.

The above studies are substantially different from our study in terms of dimensions and hierarchy of choices: In this work, (1) the number of instances is any K, instead of 1 or 2; (2) the purchase choices form a two-level hierarchy, in which the combo purchase can further cover all single purchases. This is the first study investigating two-level (single purchase and combo purchase) and multi-dimensional (multiple instances with single purchases) ski-rental problem.

Problem Formulation

Overview

The system is operated in continuous time $t \in [0, T)$, where T remains unknown till the end. The user has K instances, and each of them could generate demands (usages). The decision-maker fulfills the demand to choose one of three charging options to cover each demand: (1) Rental, (2) Single purchase, and (3) Combo purchase.

(1) **Rental**: Each 1 unit amount of demand will incur a cost of Δp. Demand of D will incur a rental cost of $D\Delta p$. We assume that demand is an integer value.

(2) **Single purchase**: At time t, the user can make a single purchase for one instance k. It will incur an immediate upfront fee $C_s\Delta p$. Then, in $[t, T)$, the demands generated by instance k will not incur any further cost. This means that the purchase is valid from t to the end. We assume that C_s is the same among different instances.

(3) **Combo purchase**: At time t, the user can make a combo purchase for all instances. It will incur an immediate upfront fee $C_c\Delta p$. Then, from $[t, T)$, the demand generated by all instances will not incur any further cost.

Without loss of generality, we assume that at most one instance at a time generates a demand. For presentation convenience, such demand is formally defined as a *task*. The task is labeled by $i = 1, 2, \ldots, I$ in chronological order. Task i can be presented by a 3-tuple: $(d(i), a(i), t(i))$, where $d(i)$ represents the amount of demand generated by task i. $a(i)$ represents the instance index $a(i) \in \{1, 2, \ldots, K\}$. $t(i)$ represents the time when task i arrives. For any task i and task j, $i < j$ indicates $t(i) < t(j)$.

We assume that users do not have any future information about the demand generated by instances (i.e., adversarial environment). For task i, the user only knows $(d(i), a(i), t(i))$ at time $t(i)$. This model does not assume that the sequence of the tasks respects any prior distribution. Therefore, an online solution is needed.

Since Δp is the proportional coefficient shared by the three payment methods, it is regarded as 1 without loss of generality in the rest of the paper, unless otherwise specified. We assume $C_s \gg 1$, as the price of purchases is much higher than rental. We assume $C_c/C_s \geq 2$ as combo purchase covers wider than single purchase. We also assume $K \geq 2$, indicating the user has at least two instances.

Cost Minimization

We aim to minimize the overall cost, such that each task is covered by rental, single purchase, or combo purchase.

Let x_k (resp. x_o) be a 0-1 variable indicating if the single purchase for instance k (resp. the combo purchase) is made. Let τ_k denote the time we make the single purchase for instance k. If the single purchase is never made, $\tau_k = \infty$. Let τ_o denote the time we make a combo purchase. If the combo purchase is never made, $\tau_o = \infty$.

We aim to minimize the overall cost $\sum_{i=1}^{I} \mathrm{rental}(i) + C_s \sum_{i=1}^{K} x_i + C_c x_o$, where $\mathrm{rental}(i)$ is the rental cost of task i. $\mathrm{rental}(i) = 0$ if it is covered by a purchase ($t(i) \geq \tau_{a(i)}$ or $t(i) \geq \tau_o$). Otherwise, it is equal to $d(i)$.

Online Algorithm and Performance Evaluation

Due to the online setting, we do not know the future, so that it is not possible to derive the minimum cost as an offline optimization problem. In this paper, we focus on online algorithm, and evaluate the proposed algorithm by competitive analysis.

Let $s = \{(d(i), a(i), t(i))\}, i = 1, 2, \ldots, I$, be any input sequence of tasks. A deterministic online algorithm ALG is called c-competitive if it satisfies:

$$\text{ALG}(s) \leq c \cdot \text{OPT}(s) \tag{1}$$

for all possible input sequences, where $\text{ALG}(s)$ is the cost of algorithm ALG given input sequence s. $\text{OPT}(s)$ is the cost of the optimal offline algorithm when the input s is known in advance.

The definition of competitive ratio of randomized online algorithms is defined on the mean performance. A randomized online algorithm ALG is called c-competitive if it satisfies:

$$\mathbb{E}(\text{ALG}(s)) \leq c \cdot \text{OPT}(s) \tag{2}$$

for all possible input sequences.

The competitive ratio is equal to or greater than 1, and the smaller the competitive ratio, the better the performance of the online algorithm. Our aim is to find the optimal online algorithm which gives the smallest competitive ratio.

Optimal Deterministic Online Algorithm

In this section, we introduce the deterministic online algorithm and prove that the optimal competitive ratio is $3-\frac{1}{C_s} \simeq 3$ as $C_s \gg 1$. We first introduce two important concepts: typical cost and threshold values. Then, we formally propose the deterministic online algorithm. Finally, we apply the competitive analysis and show the optimal competitive ratio.

Useful Definitions

Before presenting the deterministic online algorithm, we first define the *typical costs*. Suppose current time (now) is t, the *typical cost of instance k at t*, $\psi_k(t)$, is defined as the rental cost of instance k generated till now, i.e., in $[0, t]$, if we do not make any purchase now. Please note that although the task generated exactly at t may not be paid by rent, it is still counted for the calculation of typical cost at t. The *overall typical cost at t*, $\psi_c(t)$, is defined as the overall rental cost of all instances till now, i.e., in $[0, t]$, if we do not make any purchase now.

$$\psi_k(t) = \sum_{\substack{i:0\leq t(i)\leq t \text{ and } a(i)=k \\ \text{task } i \text{ is not covered by purchase before } t}} d(i), \tag{3}$$

$$\psi_c(t) = \sum_{\substack{i:0\leq t(i)\leq t \\ \text{task } i \text{ is not covered by purchase before } t}} d(i). \tag{4}$$

We define $\lambda_s \leq C_s$ and $\lambda_c \leq C_c$ as two threshold values. If the typical cost of instance k reaches λ_s, we make a single purchase of instance k. If the overall typical cost reaches λ_c, we make a combo purchase.

Algorithm 1: Deterministic Two-level Ski-Rental Algorithm DTSR(λ_s, λ_c)

```
1  if new arrival of task i: (d(i), a(i), t(i)) then
2      if task is covered by a previous purchase then
3          Repeat from Line 1
4      else
5          Set ψ_{a(i)} be the typical cost by (3)
6          if ψ_{a(i)} ≥ λ_s then
7              Make single purchase for instance a(i)
8              Repeat from Line 1
9          Set ψ_c be the overall typical cost by (4)
10         if ψ_c ≥ λ_c then
11             Make combo purchase
12             Repeat from Line 1
13         Pay the task i's demand by rental
14     Repeat from Line 1
```

The deterministic online algorithm is shown by Algorithm 1. Its two threshold values λ_s and λ_c are input parameters, and they are fixed until the end.

The algorithm is activated upon the arrival of a task. If the task has already been covered by a previous purchase (Lines 2), then the algorithm does nothing. Otherwise, it will calculate the typical costs ψ_k and ψ_c (Lines 5 and 9). If the typical cost exceeds the threshold value, the algorithm will make the corresponding purchase (Lines 7 and 11). If not, the algorithm will cover the task by rental (Line 13).

Please note that the algorithm takes both the current task and historical rent costs into account when calculates the typical costs. The main difference between our proposed algorithm and the traditional ski-rental algorithm is that two threshold values λ_s and λ_c are used at the same time, so as to realize the purchases with hierarchical relationship.

Competitive Analysis

In this subsection, we focus on the performance analysis of Algorithm 1 and derive the min competitive ratio through optimizing λ_s and λ_c.

Standardizing the Sequence For an input sequence, we focus on its performance operated by Algorithm 1. Let s denote a valid input sequence. $D(k)$ is defined as the total demand (generated in $[0, T)$) of instance k. We establish "worse cases" to bound the performance. First, we find that by moving demand from one instance to another, the cost of Algorithm 1 may increase but the optimal cost does not, so that we can derive upper bounds. We have the following two Lemmas. Let $\text{DTSR}(s, \lambda_s, \lambda_c)$ denote the cost of algorithm DTSR given input sequence s, λ_s, and λ_c:

Lemma 1. *Given any λ_s and λ_c, for any input sequence s, if there exists two instances, say instances k and l, such that $D(k) < \lambda_s$ and $D(l) < \lambda_s$, then the competitive ratio of Algorithm 1 will not decrease if we move $\min(\lambda_s - D(k), D(l))$ demand from instance l to instance k. In other words, let s' be a sequence as we move $\min(\lambda_s -$*

Algorithm 2: Standardizing Algorithm $STD(s)$

1 Create B instances with 0 demand. B is sufficiently large.
2 **while** *there exists two instances* k *and* l *s.t.* $\lambda_s < D(l) < C_s$ *and* $D(k) < \lambda_s$ **do**
3 | move $\min(\lambda_s - D(k), D(l) - \lambda_s)$ demand form l to k
4 **end**
5 **while** *there exists two instances* k *and* l *s.t.* $D(k) < \lambda_s$ *and* $D(l) < \lambda_s$ **do**
6 | move $\min(D(l), \lambda_s - D(k))$ demand form l to k
7 **end**
8 **return** *the outcome sequence* s_{std}

$D(k), D(l))$ *demand from instance* l *to instance* k*. We have*

$$\frac{DTSR(s, \lambda_s, \lambda_c)}{OPT(s)} \leq \frac{DTSR(s', \lambda_s, \lambda_c)}{OPT(s')}. \tag{5}$$

See our tech report (Wu 2020).

Lemma 2. *Given any* λ_s *and* λ_c*, for any input sequence* s*, if there exists two instances, say instances* k *and* l*, such that* $D(k) < \lambda_s$ *and* $C_s > D(l) > \lambda_s$*, then the competitive ratio of Algorithm 1 will not decrease if we move* $\min(\lambda_s - D(k), D(l) - \lambda_s)$ *demand from instance* l *to* k*. In other words, let* s' *be a sequence as we move* $\min(\lambda_s - D(k), D(l) - \lambda_s)$ *demand from instance* l *to instance* k*. We have*

$$\frac{DTSR(s, \lambda_s, \lambda_c)}{OPT(s)} \leq \frac{DTSR(s', \lambda_s, \lambda_c)}{OPT(s')}. \tag{6}$$

See our tech report (Wu 2020).

By applying the above two Lemmas, we can construct a standardized sequence by moving demand among the instances, and the competitive ratio is not reduced. Then, we can focus on analyzing the standardized sequence, which provides performance bound of any input sequence. The standardizing algorithm is shown by Algorithm 2.

The standardizing Algorithm 2 first creates B instances with 0 demand. By doing so, the cost of Algorithm 1 and the optimal cost does not change. Then, Algorithm 2 repeats operations in Lemma 2 (Lines 2–4) and Lemma 1 (Lines 5–7) to get the standardized sequence s_{std}. Please note that the algorithm can always find instance k in Line 2 since we have created sufficient zero-demand instances in Line 1.

We further define $s_{std} = STD(s)$ to denote s_{std} is a standardized sequence of s, and $\mathcal{S}_{std}$ denote the set of any standardized sequence.

It is straightforward to show that a standardized sequence satisfies the following properties

- m instances have demand no less than C_s, where $m \geq 0$.
- n instances have demand equal to λ_s, where $n \geq 0$.
- At most 1 instance has demand x in $(0, \lambda_s)$.
- All other instances have 0 demand.

From Lemmas 1 and 2, and the above procedure to generate the standardized sequence, we know Algorithm 2 will not reduce the competitive ratio. Thus the competitive ratio of the standardized sequence will not be reduced compared to the original input sequence. Therefore, it is straightforward to reach the following Lemma.

Lemma 3. *Given* λ_s *and* λ_c*,* s *is any input sequence, and* s_{std} *is the standardized sequence generated by* s*.* $s_{std} = STD(s)$*. Then, we have*

$$\frac{DTSR(s, \lambda_s, \lambda_c)}{OPT(s)} \leq \frac{DTSR(s_{std}, \lambda_s, \lambda_c)}{OPT(s_{std})}. \tag{7}$$

In what follows, we can focus to bound the performance of a standardized sequence instead of an arbitrary sequence. Since a standardized sequence is also a valid input sequence, the standardizing procedure does not influence the tightness of the bound.

Bounding Standardized Sequences For any standardized sequence, we first investigate the competitive ratio when $\lambda_s \leq \mathbf{D}(s_{std})$ and $\lambda_s \leq \lambda_c$. $\mathbf{D}(s_{std})$ is the total demand from all instances of s_{std}.

Lemma 4. *Given* λ_s *and* λ_c*,* s_{std} *is any standardized sequence. When* $\lambda_s \leq \mathbf{D}(s_{std})$ *and* $\lambda_s \leq \lambda_c$*, the optimal competitive ratio is* $3 - \frac{1}{C_s}$*.*

$$\max_{s_{std}} \min_{\lambda_s, \lambda_c} \frac{DTSR(s_{std}, \lambda_s, \lambda_c)}{OPT(s_{std})} \leq 3 - \frac{1}{C_s}. \tag{8}$$

See our tech report (Wu 2020).

Lemma 4 shows the competitive ratio of Algorithm 1 when $\lambda_s \leq \mathbf{D}(s_{std})$ and $\lambda_s \leq \lambda_c$. If $\lambda_s > \mathbf{D}(s_{std})$, then we can find λ_s, λ_c such that the competitive ratio is smaller, so that the adversarial s_{std} will not let $\lambda_s > \mathbf{D}(s_{std})$ happen. If $\lambda_s > \lambda_c$, we can find an input sequence s_{std} such that the competitive ratio is larger, so that we will not choose $\lambda_s > \lambda_c$. (The detailed proof can be found in the proof of Theorem 1.)

Based on the findings above, the following theorem can be obtained to give the optimal competitive ratio:

Theorem 1. $3 - \frac{1}{C_s} \simeq 3$ *is the optimal and tight competitive ratio of Algorithm 1 and the optimal thresholds are:*

$$\begin{aligned} \lambda_s^* &= C_s, \\ \lambda_c^* &= \frac{C_s - 1}{C_s} C_c + 1. \end{aligned} \tag{9}$$

See our tech report (Wu 2020).

Please note that we assume $C_c/C_s \geq 2$ in this paper. If it does not hold, i.e., $1 \leq C_c/C_s < 2$, Algorithm 1 still works, and the bound (8) still holds, but it is no longer tight due to the construction of the adversary.

Randomized Online Algorithm

We now discuss the randomized online algorithm. When the values of λ_s and λ_c are generated randomly, we obtain the Algorithm 3. The distribution of λ_s and λ_c are designed so that we can analyze the competitive ratio of Algorithm 3.

Algorithm 3: Randomized Two-level Ski-Rental Algorithm RTSR

```
Generate λ_s and λ_c according to PMFs (10)–(11).
if new arrival of task i: (d(i), a(i), t(i)) then
    Call DTSR(λ_s, λ_c)
Repeat from Line 2
```

The probability mass functions (PMFs) of λ_s and λ_c are

$$\mathbb{P}(\lambda_s = i) \triangleq P_i^{(d)} = \begin{cases} 1, & \text{if } i = C_s, \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

$$\mathbb{P}(\lambda_c = i) \triangleq P_i^{(o)} = \begin{cases} aq^{i-1}, \text{ if } i \in [1, C_s - 1], \\ 1 - \frac{a(1-q)^{C_s-1}}{1-q}, \text{ if } i = C_c, \\ 0, \text{ otherwise,} \end{cases} \tag{11}$$

where $q \triangleq \frac{C_c}{C_c-1}$ and $a \triangleq \frac{3C_c-2}{C_c-1} \cdot \frac{1}{(3C_c-2)(e^\sigma-1)+\sigma(e^\sigma+1)-e^\sigma/C_s}$, and $\sigma \triangleq \frac{C_s-1}{C_c-1}$.

Please note that this distributions will lead to $\frac{e^\sigma}{e^\sigma-1}$ competitive ratio to be discussed shortly.

One observation is that the λ_s is equal to C_s with probability 1. This is derived by the analysis in the proof of Theorem 2. We do not assume λ_s is deterministic at the beginning, but deterministic λ_s will lead to the nice form of bound, and will give a local minimum competitive ratio compared with other PMFs.

Competitive Analysis

We first use a relaxed version of competitive ratio for deterministic λ_s and λ_c.

Corollary 1. *Given λ_s and λ_c, s is any sequence, with total demand $\mathbf{D}(s)$. When $\lambda_s \leq \mathbf{D}(s)$ and $\lambda_s \leq \lambda_c$, $1+\frac{C_c}{\lambda_c-1}+\frac{C_s}{\lambda_s}$ is a upper bound of the competitive ratio.*

Note that Corollary 1 states a relaxed upper bound compared with Lemma 4. The bound is

$$\max_s \frac{\text{DTSR}(s, \lambda_s, \lambda_c)}{\text{OPT}(s)}. \tag{12}$$

Theorem 2. *The competitive ratio of the Algorithm 3 is*

$$\frac{\mathbb{E}[RTSR(s)]}{OPT(s)} \leq \frac{e^\sigma}{e^\sigma - 1}, \tag{13}$$

for any sequence s, where $\sigma = \frac{C_s-1}{C_c-1} \simeq \frac{C_s}{C_c}$.

Theorem 2 finally concludes the competitive ratio of Algorithm 3. This is a neat extension of the competitive ratio $\frac{e}{e-1}$ of the classic ski-rental problem (Manasse 2008).

Evaluation

In this section, we apply simulation to verify the competitive ratio claimed in Theorems 1 and 2, and further evaluate the performance of DTSR and RTSR comparing to benchmarks.

Sequence Generation and Simulation Settings

The analytical competitive ratios of DTSR and RTSR have already been derived. Note that the competitive ratios are reached under the extreme cases when the environment is fully adversarial. In this section, there in no need to further focus on the extreme cases, and we are more interested to study more practical performance of DTSR and RTSR when the environment is stochastic.

The input sequence is generated randomly in this simulation. We set the number of instances as 6. For each single sequence, we generate S tasks. Each task has a unit demand. S is uniformly distributed from 1 to 60. The inter-arrival time of tasks is uniformly distributed in $[0, 1]$. (In fact, the inter-arrival time will not influence the optimal costs or the costs generated by algorithms as long as the tasks follow the same order.) Each task is randomly assigned an instance, following two piratical distributions:

1. Uniform: Tasks are uniformly distributed among all instances.
2. Heavy-tailed: The distribution of tasks among instances is heavy-tailed. 80% tasks are randomly picked and they are assigned one of 2 popular instances (with probability 0.5 each). The rest 20% tasks are assigned one of the rest 4 instances (with probability 0.25 each).

We set the unit demand price as 1, combo purchase fee as $C_c = 30$, and single purchase fee as $C_s = 9$. These values are regarded as default values. Without otherwise specified, these default values are employed throughout this simulation.

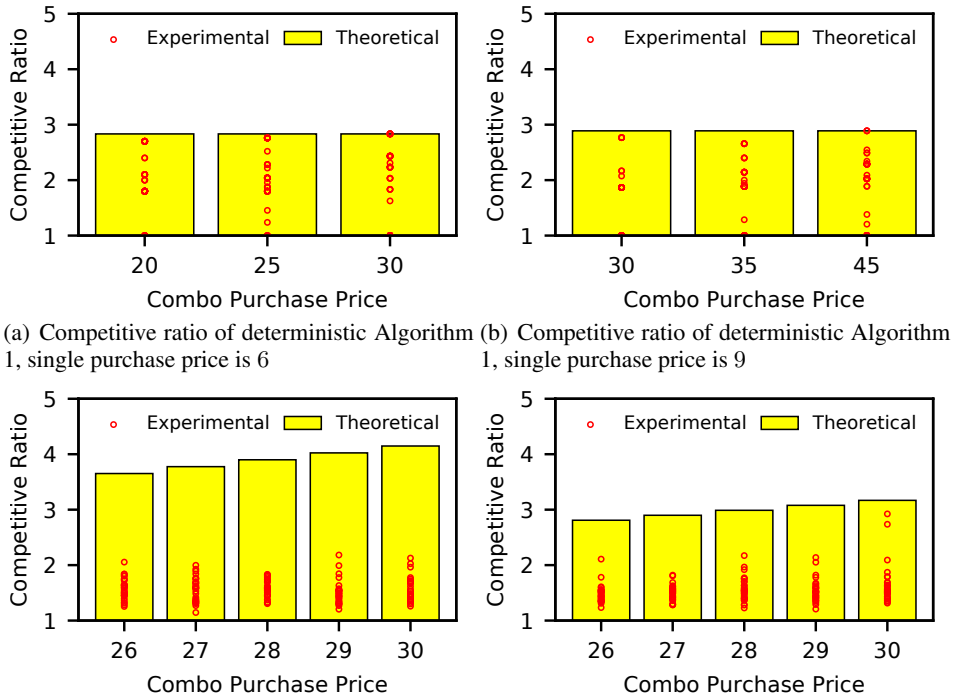
We consider the following benchmarks to compare with DTSR and RTSR:

- Ski-rental combo purchase only [SRCO]: Using the randomized ski-rental algorithm (Mathieu 2007) by only allowing rental and combo purchase.
- Ski-rental single purchase only [SRSO]: Using the randomized ski-rental algorithm by only allowing rental and single purchases.
- Random [RND]: The decision is made randomly. However, once the algorithm makes the combo purchase, there will be no more rental cost or single purchase cost. Similarly, once the algorithm makes a single purchase, there will be no more rental cost of that instance. To balance the costs from the three charging options, we set the probabilities using rental as 0.7, single purchase as 0.2, and combo purchase as 0.1.

We use Python 3.6.8 with numpy 1.18.1 to conduct the simulation in a laptop with CPU i5-8210Y and Memory 8 GB 2133 MHz LPDDR3.

Competitive Ratio Verification

In this subsection, the competitive ratios are verified. We obtain the competitive ratio by running Algorithms 1 and 3 on the randomly generated sequences. The deterministic Algorithm 1 runs on 10 sequences (5 uniform sequences and 5 heavy-tailed sequences). The randomized Algorithm 3 runs on 30 sequences (15 uniform sequences and 15 heavy-tailed



(a) Competitive ratio of deterministic Algorithm 1, single purchase price is 6

(b) Competitive ratio of deterministic Algorithm 1, single purchase price is 9

(c) Competitive ratio of randomized Algorithm 3, single purchase price is 9

(d) Competitive ratio of randomized Algorithm 3, single purchase price is 12

Figure 1: Competitive ratio verification.

sequences) and the performance at each sequence is averaged over 30 times of randomly distributed λ_s and λ_c values. The results are shown in Fig. 1(a)–1(d). The optimal offline solution is $\min(C_c, \sum_{i=1}^{K} min(C_s, r_i))$, where r_i is the rental-only cost of instance i.

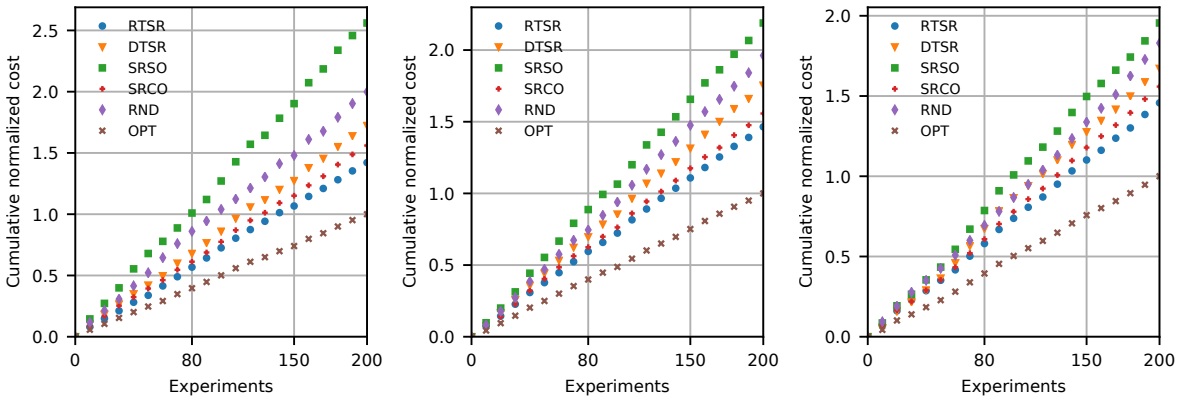
Fig. 1(a) shows the experimental and theoretical competitive ratios of Algorithm 1 with single purchase price 6. The combo purchase prices are 20, 25, and 30 respectively from left to right. The theoretical competitive ratio is shown by the yellow bar and the experimental values are shown by the red circles. As the figure shows, the experimental values are inside the yellow bar, demonstrating that the theoretical competitive ratio will bound the ratio of real cost and optimal cost. Another observation is that we have some experimental results right on the theoretical value, verifying that that the theoretical competitive ratio is tight. Fig. 1(b) shows the simulation results when single purchase price is 9. The combo purchase prices are 30, 35, and 45 respectively. The results are similar to Fig. 1(a).

Fig. 1(c) shows the experimental and theoretical competitive ratios of Algorithm 3 with single purchase price 9. The combo purchase price is from 26 to 30 from left to right. The experimental results are all inside yellow bars, which indicates that the theoretical competitive values hold. One observation is that the theoretical competitive ratios are large, but the actual performance of the algorithm is better. This is because we did not claim tight competitive ratio of Algorithm 3 in Theorem 2. While the theoretical value increases with the rise of combo purchase price, the experimental value does not show this trend. Fig. 1(d) shows the results with the single purchase price 12. In this case, both the theoretical and experimental results are better than those shown in Fig. 1(c). The experimental competitive ratio have averaged value of about 1.5 and 99% of them are lower than 2. As a result, we can see that the actual performance of DTSR and RTSR in stochastic environment is usually much better than the worst-case performance bounded by competitive ratio.

Performance Comparison of Online Algorithms

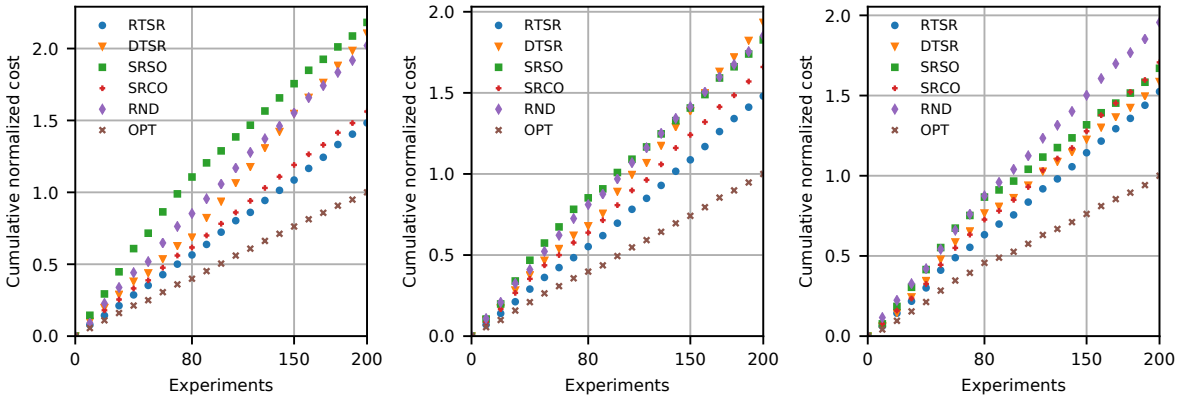
In this subsection, we focus on the evaluation of the DTSR and RTSR, compared with the benchmarks. The experiment is divided into two groups, one group uses uniform sequences, the other group mixes 40% uniform sequences at the beginning and 60% heavy-tailed sequences later. Each figure shows the cumulative costs of 200 randomly generated sequences. Each sequence is run by each algorithm once. The cumulative cost is normalized by the sum of optimal costs of the 200 sequences, so that we can more clearly show the ratio of the cost of an algorithm and the optimal cost. In the figure, coordinate (x, y) indicates the sum cost of an algorithm of first x sequences divided by the sum optimal cost of the 200 sequences. The value at the y-axis when $x = 200$ is actually the realized competitive ratio of the corresponding algorithm. OPT indicates the optimal offline algorithm, so that its cumulative normalized cost is 1 when $x = 200$. The results are shown in Figs. 2 and 3.

Fig. 2 shows the results for uniform sequences, under different prices of combo purchase. We can show that RTSR



(a) Uniform, combo purchase price is 20 (b) Uniform, combo purchase price is 25 (c) Uniform, combo purchase price is 31

Figure 2: Performance of algorithms under uniform sequences.



(a) Mixed, combo purchase price is 20 (b) Mixed, combo purchase price is 25 (c) Mixed, combo purchase price is 33

Figure 3: Performance of algorithms under mixed sequences.

performs the best, followed by SRCO. The performance of DTSR is worse than SRCO, but better than other two benchmarks. It demonstrates that RTSR is the most advantageous algorithm in more realistic stochastic scenarios. For DTSR, the $(3 - \frac{1}{C_s})$-competitive ratio is reached under the extreme condition. However, with the uniform sequence, the future arrival can be partially guessed from the history, DTSR is too conservative to make purchases, leading to less effective performance. Nevertheless, RTSR can address this issue as the threshold values are generated randomly. It leads to more proactive purchases by probability and thus better mean performance. In sum, RTSR can provide a worst-case theoretical performance bound and its real-world performance in less adversarial environment is still robust.

Fig. 3 shows the results of the mixed sequences, under different prices of combo purchase. RTSR performs the best in all situations, showing its robustness under different settings. However, the performance of other algorithms is not stable. In Figs. 3(a)–(b), SRCO performs the second best, and in Fig. 3(c), DRSR performs the second best. It further demonstrates that RTSR is the most advantageous under a variety of system settings.

Conclusion

In this paper, we focused on the two-level ski rental problem dealing with complex payment options in the real world. To handle this problem, we first proposed an optimal $(3 - \frac{1}{C_s})$-competitive deterministic algorithm (DTSR). We claimed that $(3 - \frac{1}{C_s})$ is the tight bound of competitive ratio for DTSR. Then we further proposed a randomized algorithm (RTSR) and investigated its theoretical performance by competitive analysis. For the random thresholds used in RTSR, we derived close-form PMFs and proved that RTSR is $\frac{e^\sigma}{e^\sigma - 1}$-competitive, where σ is the ratio between the single purchase price and the combo purchase price, which is a neat extension of $\frac{e}{e-1}$-competitive in the classical ski-rental problem. In addition, we applied simulation to verify the theoretical results and further concluded that the RTSR performs the best under different settings. For future works, the non-uniformed single purchase prices can be considered. Moreover, for the more general mobile plan reservation problem, analysis should be extended to cover the temporal dimension, as a plan may start and end at any time.

References

Ai, L.; Wu, X.; Huang, L.; Huang, L.; Tang, P.; and Li, J. 2014. The multi-shop ski rental problem. *Performance evaluation review* 42(1): 463–475. ISSN 0163-5999.

Chen, X.; and Xu, W. 2018. A risk–reward model with compound interest rate for non-additive two-option ski rental. *Information Processing Letters* 135: 9–13.

Epstein, L.; and Zebedat-Haider, H. 2015. Rent or Buy Problems with a Fixed Time Horizon. *Theory of Computing Systems* 56(2). ISSN 1432-4350.

Feldkord, B.; Markarian, C.; and Meyer Auf der Heide, F. 2017. Price Fluctuation in Online Leasing. In *Combinatorial Optimization and Applications*, Lecture Notes in Computer Science, 17–31. Cham: Springer International Publishing. ISBN 9783319711461. ISSN 0302-9743.

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

Fujiwara, H.; Shibusawa, K.; Yamamoto, K.; and Yamamoto, H. 2020. Bounds for the Multislope Ski-Rental Problem. *IEICE Trans. Inf. Syst.* 103-D(3): 481–488. URL http://search.ieice.org/bin/summary.php?id=e103-d_3_481.

Hu, X.; Ludwig, A.; Richa, A.; and Schmid, S. 2015. Competitive Strategies for Online Cloud Resource Allocation with Discounts: The 2-Dimensional Parking Permit Problem. In *2015 IEEE 35th International Conference on Distributed Computing Systems*, volume 2015-, 93–102. IEEE. ISBN 9781467372145. ISSN 1063-6927.

Karlin; Kenyon; and Randall. 2003. Dynamic TCP Acknowledgment and Other Stories about e /(e - 1). *Algorithmica* 36(3): 209–224. ISSN 0178-4617.

Khanafer, A.; Kodialam, M.; and Puttaswamy, K. P. N. 2013. The constrained Ski-Rental problem and its application to online cloud cost optimization. In *2013 Proceedings IEEE INFOCOM*, 1492–1500. IEEE. ISBN 9781467359443. ISSN 0743166X.

Kodialam, R. 2019. Optimal Algorithms for Ski Rental with Soft Machine-Learned Predictions. *arXiv: Learning* .

Lee, G.; Saad, W.; Bennis, M.; Mehbodniya, A.; and Adachi, F. 2017. Online Ski Rental for ON/OFF Scheduling of Energy Harvesting Base Stations. *IEEE Transactions on Wireless Communications* 16(5): 2976–2990. ISSN 1536-1276.

Lotker, Z.; Patt-Shamir, B.; and Rawitz, D. 2012. Rent, Lease, or Buy: Randomized Algorithms for Multislope Ski Rental. *SIAM Journal on Discrete Mathematics* 26(2): 718–736. ISSN 0895-4801.

Manasse, M. S. 2008. Ski Rental Problem.

Mathieu, C. 2007. Online algorithms: Ski rental. http://cs.brown.edu/%7eclaire/Talks/skirental.pdf.

Meyerson, A. 2005. The parking permit problem. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, volume 2005, 274–282. IEEE. ISBN 0769524680. ISSN 02725428.

Shi, M.; Lin, X.; Fahmy, S.; and Shin, D.-H. 2018. Competitive Online Convex Optimization with Switching Costs and Ramp Constraints. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, volume 2018-, 1835–1843. IEEE. ISBN 9781538641286. ISSN 0743166X.

Wang, S.; Li, J.; and Wang, S. 2020. Online Algorithms for Multi-shop Ski Rental with Machine Learned Predictions. *arXiv.org* ISSN 2331-8422.

Wu, Bao, Y. 2020. TechReport: Competitive Analysis for Two-Level Ski-Rental Problem. [EB/OL]. https://weibao-cs.github.io/TechReport/AAAI2021TR.pdf.

Yang, S.; Pan, L.; and Liu, S. 2019. An Online Algorithm for Selling Your Reserved IaaS Instances in Amazon EC2 Marketplace. In *2019 IEEE International Conference on Web Services (ICWS)*, 296–303. IEEE. ISBN 9781728127170.

Zang, S.; Bao, W.; Yeoh, P. L.; Vucetic, B.; and Li, Y. 2019. Filling Two Needs With One Deed: Combo Pricing Plans for Computing-Intensive Multimedia Applications. *IEEE Journal on Selected Areas in Communications* 37(7): 1518–1533. ISSN 0733-8716.

Zhang, G.; Poon, C. K.; and Xu, Y. 2011. The ski-rental problem with multiple discount options. *Information Processing Letters* 111(18): 903–906.

Zhang, H.; and Conitzer, V. 2020. Combinatorial Ski Rental and Online Bipartite Matching. In *Proceedings of the 21st ACM Conference on economics and computation*, EC '20, 879–910. ACM. ISBN 1450379753.