Online algorithms for cost-effective cloud selection with multiple demands

Youngmi Jin, Michiaki Hayashi, Atsushi Tagami
KDDI Research, Inc.
Saitama, 356-8502, Japan
Email:{yo-jin, mc-hayashi, tagami}@kddi-research.jp

***Abstract*—Cloud computing provides high flexibility for users by offering diverse cloud instances with various leasing periods and prices. Depending on the amount and duration of workload, a user can flexibly choose proper cloud instances to meet her demands. An intrinsic challenge facing the user is which classes of clouds and how many of them to purchase in order to meet her unpredictable demands at minimum cost. We consider an online problem deciding cost-effectively cloud classes and amount of clouds to meet dynamic multiple demands among many cloud classes when no future information of demands is available. We propose two online algorithms achieving $O(M)$ and $O(\log M + \log d_{\max})$ competitive ratios where M is the number of available cloud classes and $d_{\max}$ is the maximum demand of a given demand sequence.**

I. Introduction

Cloud computing offers various computing services in the leasing form at any time, anywhere, by accessing virtual machines that reside over a network. The virtualization and remote access over a network provides high flexibility both for cloud service providers and users.

Users can have many choices in purchasing clouds to meet their demands as diverse cloud instances with different purchasing options are offered. For example, Amazon EC2 offers on-demand instances and reserved instances with diverse leasing periods as in Table I. [1] Reserved instances offer users to make one time payment for reserving instances over the contract leasing period at discounted price for the long-term commitment of users. On-demand instances allows users to pay only for the incurred instances by the hour without any long-term commitment. Reserved instances are beneficial for services with long running time and on-demand services are for services with sporadic short running time.

TABLE I
Pricing of Amazon EC2 instances (t2.nano)

instance	contract period	price	hourly rate
on-demand	1 hour	$0.01	0.01
1-year term reserved	1 year (8760 hours)	$63.00	0.0072
3-year term reserved	3 years (26280 hours)	$124.00	0.0047

[1]Originally the price of a reserved instance consists of up-front fee and monthly payment. The prices in Table I corresponds to the total cost for the whole leasing periods, i.e., for 1-year term reserved, the total cost is (upfront payment + 12 times monthly payment).

If a user has a workload with short run time, then she buys an on-demand instance, and pays only for the amount of time she has used. If she has a workload taking long time, then she purchases a reserved instance for a fixed leasing period and can enjoy discounted price compared to the price of the on-demand instance. However, if she does not have prior knowledge of the duration or amount of her job requests, which instances and how many of them should she buy to execute her jobs while minimizing the cost?

A common example for this question is a small company (or a cloud broker) who does not have her own resources but leases resources from cloud providers and offers the leased resources to her customers. This paper considers the decision problem of such a cloud user. The fundamental challenge of this problem arises from its online setting where she has to make decisions without whole information of demand requests. The purpose of this paper is to design cost effective online algorithms deciding the class of clouds and the number of the clouds to purchase to meet multiple demands when there exist multiple classes of clouds and no future information of the demand requests is available in advance.

This decision problem is a typical variant of the ski rental problem [1], [2] that captures the trade-off between buying and renting skis when the snowing time (the usage period) is not known in advance. As cloud service becomes popular, many variants of ski-rental problems are reconsidered and generalized to capture complex service scenarios and to fully utilize the flexibility of leasing clouds [3]–[7]. One simple extension is the ski-rental problem with multiple demands, which is analyzed in [3]. Another natural extension of the ski rental problem is multiple renting options instead of two choices of renting and buying, which is called the multi-level ski rental problem. The multi-level ski rental problem is extensively studied in various forms, for example, [8], [9]. Ai et al. consider a generalization of the multi-level ski rental problem, named by multi-shop ski-rental problem, in [4] where each shop has different prices for renting and purchasing skis and customers make decisions on when and where to buy. Khanafer et al. show that the performance of online algorithms of the ski rental problem can be enhanced by the use of statistical information and propose online algorithms exploiting the mean (or variance) of the distribution of the number of ski-days in [5].

Note that the buying option in ski rental problem has no

978-0-9883045-5-0/18/$31.00 ©2018 ITC
DOI 10.1109/ITC30.2018.00014

expiration date. Allowing expiration for the options yields two different variants, the Bahncard problem [10] and the parking permit problem [11]. The Bahncard problem is an online ticketing problem between two choices, buying a Bahncard or not. A Bahncard holder enjoys price reduction (for example 50% price reduction) on all train tickets for one year from the purchase date of it. The parking permit problem is an online problem of a commuter who drives only rainy days and purchase a parking permit among many options, for example, daily, weekly, monthly permits. Both the parking permit problem and the Bahncard problem consider on-off demands, that is, each demand is 0 or 1. The main difference between the Bahncard problem and the parking permit problem is pricing scheme. In the Bahncard problem, a Bahncard holder receives discounts for each trip she has. Hence it is a kind of usage-based pricing. However, in the parking permit problem, a commuter pays a fixed amount at the purchasing time, even though she does not park a car on sunny days during the allowed parking period, and there is no extra payment after the purchase.

A generalization of the Bahncard problem in the context of cloud selection, i.e. the Bahncard problem with multiple demands, is studied in [6]. The authors consider online algorithms that decide when to purchase reserved instances and how many of them to purchase when two instances, reserved and on-demand instances, are available. In this problem, when a user purchases reserved instances, for example 1 year instances, she pays ahead fixed up-front fee and then additionally makes monthly payments (whose hourly rates are discounted ones compared to the online distances) for each month she uses (reserved) instances.

We consider a generalization of the parking problem, i.e., the parking permit problem with multiple (parking) demands. Our problem is similar to [6] in that time varying multiple demands are considered. But users do not make additional payment for each month in our problem setting but they make one time payment for the whole contract period. The parking permit problem with multiple demands is studied also in [7]. The work of [7] is different from ours in that [7] considers two dimensional parking permits whose class depending on leasing period and the number of clouds (or parking lots) so that one permit of a class means a set of multiple clouds of the class; for example one permit of a class consists of 10 clouds and buying one permit of the class means that maximum 10 demand requests can be simultaneously executed at each time during the leasing period. In our setting, one permit of a class is a single cloud and can execute only one single request.

Our contributions are

1) We find the (offline) optimal solution of the relaxed parking permit problem with multiple parking demands.
2) We propose an (integer) online algorithm with $O(\log M)$-competitive ratio and a fractional online algorithm with $O(\log M + \log(d_{\max}))$-competitive ratio where M is the total number of cloud classes and $d_{\max}$ is the maximum demand of a given demand sequence.

The rest of the paper is organized as follows. Section II formulates the problem under consideration. We describe the interval model introduced in [11] that relaxes the original cost optimization problem, and find the optimal solution of the relaxed problem in Section III. We propose an online algorithm achieving M-competitive ratio in Section IV based on the optimal solution of the relaxed optimization problem and a fractional online algorithm with $O(\log M + \log d_{\max})$ using primal-dual approach [12] in Section V, where M is the number of classes of clouds and $d_{\max}$ is the maximum demand request. Section VI concludes with the summary of the results of the paper.

II. Problem formulation

Let $\mathcal{M} = \{1, 2, ..., M\}$ be the set of cloud classes. Each cloud i has its own pricing scheme, consisting of two components, l_i the length of contract period, and p_i the unit price of a sinlge cloud of class i for the contract period l_i. We assume that l_i and p_i are increasing with i,

$$l_1 < l_2 < .... < l_i < l_{i+1} < ... < l_M,$$
$$p_1 < p_2 < .... < p_i < p_{i+1} < ... < p_M.$$

A user pays p_i for the use of a single cloud of class i when she makes a contract for it. Once she purchases clouds of some class, she can use the clouds for the given fixed leasing term of the class. For example, if a user buys a cloud for 30 days on April 1st, then she is allowed to use the cloud by April 30^{th} regardless of actual usage time. After 30 days, the cloud becomes invalid and she cannot use it any more. She can purchase multiple clouds of various classes to meet her demands.

For each class i, the hourly rate is defined as $r_i = \frac{p_i}{l_i}$. We assume that the hourly rate $r_i = \frac{p_i}{l_i}$ is decreasing with i. This means that the longer the contract, the cheaper the hourly rate,

$$r_1 > r_2 > .... > r_i > r_{i+1} > ... > r_M.$$

It is assumed that time axis is divided into many time slots and the length of a single time slot is l_1. A set of job requests d_t enters at the beginning of each time slot t. The job requests are expected to be finished by the end of time slot t. Without the entire information of demands such that how many of demands enter at each time and when the demands stop to enter, she wants to decide the purchase time and number of clouds of each i^{th} class to minimize the total cost while meeting each job request for the time interval $[0, T-1]$.

If the whole demand requests d_t for $0 \le t \le T-1$ are known (which is called an offline setting), she can solve the following optimization problem

$$\text{Minimize} \quad \sum_{t=0}^{T-1} \sum_{i=1}^{M} n_{i,t} p_i \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{M} \sum_{s=t-l_i+1}^{t} n_{i,s} \ge d_t \text{ for all } t \tag{2}$$

where $n_{i,t}$ is the number of i^{th} level clouds purchased at time t. We assume that $n_{i,t} = 0$ for all i if $t < 0$. If $s \in [t - l_i +$

$1, t]$, then the clouds of class i purchased at time s are valid to use. Taking account into this, we can know that the term $\sum_{s=t-l_i+1}^{t} n_{i,s}$ is the total number of clouds of class i that are valid to use at time t.

However she does not have the entire information of demands but she has to make her decision at each time the demands enters. The main challenge of her decisions arises from such online setting where each constraint (2) reveals one by one as demands enter, while in the offline setting, the set of constraints (2) for all t are given at time 0 and the optimization is done with the whole demand information. We are interested in online algorithms that decide the number of clouds of each class to purchase at each time the demands enter.

Because of the lack of the whole information, the decision made by an online algorithm may turn out later to be non-optimal. Hence the important issue in the design of an online algorithm is how much an online algorithm reduces the performance gap between the cost of the online algorithm and the (offline) optimal cost. This performance gap is called the competitive-ratio of an online algorithm A [13], which is defined by α_A such that

$$\mathcal{C}_A(\underline{d}(t)) \leq \alpha_A \mathcal{C}^*(\underline{d}(t))$$

for all $\underline{d}(t)$ and for all $0 \leq t \leq T-1$, where $\mathcal{C}_A(\underline{d}(t))$ is the total purchase cost decided by algorithm A and $\mathcal{C}^*(\underline{d}(t))$ the optimal cost during the time interval $[0, t]$ with a demand request vector $\underline{d}(t) = (d_0, d_1, .., d_t)$. The smaller α, the better algorithm.

The aim of this paper is to propose cost-effective online algorithms for cloud selection and find their competitive ratios. Note that if $M = 2$, $l_2 = \infty$, and $d_t \in \{0, 1\}$, then our problem becomes a ski rental problem [1], [2]. If $M \geq 2$ and $d_t \in \{0, 1\}$, then it becomes a parking permit problem [11].

III. Optimal offline solution in interval model

This section studies the optimal offline solution of (1). Recall that in the offline setting, all the demand requests d_t's are known at the beginning.

A. Cloud selection problem in the interval model

The optimization problem of (1) can be solved by dynamic programming that usually requires heavy computation. To overcome such computational complexity, we consider the interval model proposed in [11] which is a common model for infrastructure leasing problems [14]. The main assumptions for the interval model are as follows.

- Clouds of class i are purchased only for the time intervals of the form $[jl_i, (j+1)l_i - 1]$ for some integer j.
- T is divisible by l_i for all i, i.e., $\frac{T}{l_i}$ is an integer. Let $\tau_i = \frac{T}{l_i} - 1$.
- l_i is divisible by l_m for $i > m$, i.e., $l_i = kl_m$ for some integer k.

In the offline setting with the interval model, it is obvious that a user will buy clouds of class i at time jl_i, the beginning of the interval, to save or minimize the cost if a user knows all demand requests and buys clouds of class i for the interval $[jl_i, (j+1)l_i - 1]$. Let $\mathcal{C}^*$ be the optimal cost of the original problem and $\mathcal{C}^*_{\mathsf{INT}}$ be the optimal cost in the interval model. Since any leasing interval of a single cloud of i^{th} class can be covered by at most two consecutive i^{th} level intervals in the interval model (see Figure 1), the following relationship holds

$$\mathcal{C}^* \leq \mathcal{C}^*_{\mathsf{INT}} \leq 2\mathcal{C}^*.$$

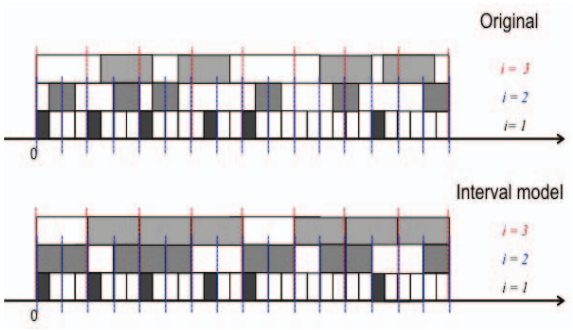


Fig. 1. interval model

This subsection finds the offline optimal solution in the interval model. The offline optimal solution in the interval model can be found by minimizing the cost interval by interval. With this observation, we consider only one single interval $L_{M,0} = [0, T-1]$ with $T = l_M$ in the whole paper from now on. [2]

Suppose that we know the whole demand sequence $(d_0, d_1, ..., d_{T-1})$ at time 0. In the interval model, as mentioned above, we can minimize the cost by purchasing clouds of class i at the very beginning time jl_i of the interval $L_{i,j} = [jl_i, (j+1)l_i - 1]$. Therefore, at the minimum cost, we have $n_{i,jl_i} \geq 0$ for an integer j and $n_{i,t} = 0$ for t that is not divisible by l_i. This implies that we can consider only n_{i,jl_i} and ignore $n_{i,t}$ for t non-divisible by l_i. Since only n_{i,jl_i}'s are considered, we will use $n_{i,j}$ instead of n_{i,jl_i} for simple notation from now on. Let $j_i^t = \lfloor \frac{t}{l_i} \rfloor$ for $i = 1, ..., M$. Note that this j_i^t is the unique integer j for cloud class i such that $jl_i \leq t \leq (j+1)l_i - 1$, i.e., L_{i,j_i^t} is the unique interval of level i containing t. Using the interval model, the optimization problem (1) can be rewritten as follows

$$\begin{aligned} \min_{\underline{n}} \quad & \sum_{i=1}^{M} \sum_{t=0}^{\tau_i} p_i n_{i,j} \\ \text{subject to} \quad & \sum_{i=1}^{M} n_{i,j_i^t} \geq d_t \text{ for } 0 \leq t \leq T-1 \end{aligned} \tag{3}$$

where $\underline{n} = (n_{i,j})$ for all (i, j) pairs.

[2] If T is not divisible by l_M, we let $d_t = 0$ for the job requests at time $T \leq t \leq (\lfloor \frac{T}{l_M} \rfloor + 1)l_M - 1$. With this reasoning, we assume T is divisible by l_M.

B. Offline optimal solution in the interval model

To find the optimal offline solution of (3), we consider the Lagrangian of it,

$$\mathcal{L}(\underline{n},\underline{\lambda}) = \sum_{i=1}^{M}\sum_{j=0}^{\tau_i} p_i n_{i,j} + \sum_{t=0}^{T-1}\lambda_t(d_t - \sum_{i=1}^{M} n_{i,j_i^t}),$$

and the corresponding Lagrange dual problem,

$$\begin{array}{lll} \max\limits_{\underline{\lambda}} \ \min\limits_{\underline{n}} & \mathcal{L}(\underline{n},\underline{\lambda}) & (4)\\ \text{subject to} & \lambda_t \geq 0 \ \text{ for all } t, & \end{array}$$

where $\underline{\lambda} = (\lambda_0, ..., \lambda_{T-1})$. Considering the last term of $\mathcal{L}(\underline{n},\underline{\lambda})$,

$$\begin{aligned}\sum_{i=1}^{M}\sum_{t=0}^{T-1}\lambda_t n_{i,j_i^t} &= \sum_{i=1}^{M}\sum_{j=0}^{\tau_i}\sum_{t=jl_i}^{(j+1)l_i-1}\lambda_t n_{i,j_i^t}\\ &= \sum_{i=1}^{M}\sum_{j=0}^{\tau_i} n_{i,j}(\lambda_{jl_i} + \lambda_{jl_i+1}+, ..., \lambda_{(j+1)l_i-1}),\end{aligned}$$

we have

$$\begin{aligned}\mathcal{L}(\underline{n},\underline{\lambda}) &= \Big(\sum_{i=1}^{M} p_i \sum_{j=0}^{\tau_i} n_{i,j}\Big) + \sum_{t=0}^{T-1}\lambda_t d_t - \sum_{t=0}^{T-1}\sum_{i=1}^{M}\lambda_t n_{i,j_i^t}\\ &= \sum_{t=0}^{T-1}\lambda_t d_t + \sum_{i=1}^{M}\sum_{j=0}^{\tau_i} n_{i,j}(p_i - \sum_{s\in L_{i,j}}\lambda_s)\end{aligned}$$

where $L_{i,j} = \{s \mid jl_i \leq s \leq (j+1)l_i - 1\}$.

The Lagrange dual problem (4) becomes

$$\max_{\underline{\lambda}} \sum_{t=0}^{T-1}\lambda_t d_t \quad (5)$$

$$\text{subject to} \quad \sum_{s\in L_{i,j}}\lambda_s \leq p_i \text{ for all } (i,j) \text{ pairs.} \quad (6)$$

The Lagrange dual problem (5) is equivalent to the following one

$$\max_{\underline{\lambda}} \sum_{t=0}^{T-1}\lambda_t d_t \quad (7)$$

subject to

$$\begin{aligned}\lambda_t \geq 0 \quad &\text{for all} \quad t,\\ n_{i,j}(p_i - \sum_{s\in L_{i,j}}\lambda_s) = 0 \quad &\text{for all} \quad (i,j). \quad (8)\end{aligned}$$

From (8) (and (6)), we know that $n_{i,j} = 0$ if $p_i > \sum_{s\in L_{i,j}}\lambda_s$ and $p_i = \sum_{s\in L_{i,j}}\lambda_s$ if $n_{i,j} > 0$. Hence, for the case of $i = 1$, we have $\lambda_t \leq p_1 = 1$ for any t. To maximize $\sum_t \lambda_t d_t$, we assign $\lambda_t = 1$ for the highest demand d_t and then assign $\lambda_t = 1$ for the second highest demand. We consecutively assign $\lambda_t = 1$ in decreasing order of d_t if $\sum_{s\in L_{i,j}, d_s > d_t}\lambda_s \leq p_i - 1$ for all i. After the decision of λ_t, we find (i,k) pairs such that $n_{i,k} = 0$ and decide other $n_{i,j}$'s values using the constraint

$$\sum_{i=1}^{M} n_{i,j_i^t} \geq d_t \quad \text{for all} \quad t. \quad (9)$$

These processes are implemented in Algorithm 1. From now on, without loss of generality, we suppose that p_i is divisible by p_1, and $p_1 = 1$ for simple analysis.

Algorithm 1 Finding the offline optimal solution

Input: $(d_0, d_1, ..., d_{T-1})$, a demand sequence
Output: $n^*_{i,j}$ for all (i,j) pairs.

Sort $(d_0, d_1, ..., d_{T-1})$ in decreasing order:
    $(\sigma_0, \sigma_1, ..., \sigma_{T-1})$ is the sorted one.

Initialization:

$$\begin{aligned}\lambda^*_t &= 0 \quad \forall t,\\ D^+_{i,j} &= \emptyset \quad \forall (i,j),\\ n^*_{i,j} = \theta_{i,j} &= 0 \quad \forall (i,j).\end{aligned}$$

for $r = 0$ to $T-1$ **do**
    $s_r = \pi(\sigma_r)$.
    if $|D^+_{i,j_i^{s_r}}| \leq p_i - 1$ for all $i \geq 2$ **then**

$$\lambda^*_{s_r} = 1, \quad (10)$$

$$D^+_{i,j_i^{s_r}} = D^+_{i,j_i^{s_r}} \cup \{\sigma_r\} \quad \forall i. \quad (11)$$

    end if
end for

for all (i,j) pairs **do**
    $\theta_{i,j} = \min D^+_{i,j}$ **if** $|D^+_{i,j}| = p_i$.
end for

for all (i,j) **pairs do**
    $U_{i,j} = \{(m,k) \mid L_{i,j} \subset L_{m,k},\ |D^+_{m,k}| = p_m\}$.
    if $U_{i,j} \neq \emptyset$ **then**
        Find $(\tilde{i},\tilde{j}) \in U_{i,j}$ **such that** $\tilde{i} = \min_{(m,k)\in U_{i,j}} m$.
    end if
end for

for all (i,j) **pairs do**

$$\tilde{\theta}_{i,j} = \begin{cases} \theta_{\tilde{i},\tilde{j}} & \textbf{if } U_{i,j} \neq \emptyset,\\ 0 & \textbf{otherwise}.\end{cases}$$

    if $|D^+_{i,j}| = p_i$ **then**

$$n^*_{i,j} = \theta_{i,j} - \tilde{\theta}_{i,j}. \quad (12)$$

    end if
end for

We denote by $n^*_{i,j}$ the optimal number of i^{th} level clouds for the interval $L_{i,j}$ and λ^*_s the dual variable (Lagrange multiplier) associated with the optimization of (3). To avoid equality between any two elements, we say that d_{t_1} is higher than

TABLE II
λ_t^* AND $D_{i,j}^+$ OF THE EXAMPLE

Time slot t	0	1	2	3	4	5	6	7	8	9	10	11
Demands d_t	**4**	**8**	**6**	**7**	**10**	**2**	**1**	**5**	**3**	**9**	**2**	**4**
r^{th} highest	6	2	4	3	0	9	11	5	8	1	10	7
D^+_{3,j_3^t}		{10,9,8}	{10,9,8,7,6}	{10,9,8,7}	{10}			{10,9,8,7,6,5}		{10,9}		
D^+_{2,j_2^t}		{8}	{8,7,6}	{8,7}	{10}			{10,5}		{9}		
$D^+_{1,t}$		{8}	{6}	{7}	{10}			{5}		{9}		
λ_t^*	0	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	0	0

d_{t_2}, denoted by $d_{t_1} \succ d_{t_2}$, if

$$\begin{cases} d_{t_1} > d_{t_2}, \text{or} \\ d_{t_1} = d_{t_2} \text{ and } t_1 < t_2. \end{cases}$$

With this ordering, one element is strictly higher than the other one. Note that this ordering does not change the solution of the optimization problem.

We define

$$\pi(d_t) = t,$$

which is the time when demand d_t enters into the system. Summarizing Algorithm 1, the processes of finding $n_{i,j}^*$ of the optimization problem (7), is as follows:

1) Sort $(d_0, d_1, ..., d_{T-1})$ in decreasing order. Let $(\sigma_0, ..., \sigma_{T-1})$ be the sorted sequence.
2) Decide $\lambda^*_{\pi(\sigma_r)}$ one by one in the demand decreasing order.
3) Find (i,j) pairs with $n_{i,j}^* = 0$.
4) Decide $n_{i,j}^*$ whose values are not zero according to (9).

After the termination of Algorithm 1, for all (i,j) pairs, we have

$$D_{i,j}^+ = \{d_s \mid \lambda_s^* > 0, s \in L_{i,j}\}, \quad (13)$$

$$\theta_{i,j} = \begin{cases} \min D_{i,j}^+ & \text{if } |D_{i,j}^+| = p_i, \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

Note that $D_{i,j}^+$ specifies demands in $L_{i,j}$ whose λ values are to set 1. The value $\theta_{i,j}$ is closely related to the number $n_{i,j}^*$. Indeed, $\theta_{i,j}$ is the total number of clouds to purchase whose classes are not less than i for the interval $L_{m,k}$'s containing $L_{i,j}$,

$$\theta_{i,j} = \sum_{(m,k): m \geq i, L_{i,j} \subset L_{m,k}} n_{m,k}^* \quad (15)$$

Note that for $i = M$, there is only one single interval $L_{M,0}$. When deciding $n_{i,j}^*$ for $i < M$, we need to consider how many of clouds of class $m > i$ are available and buy the difference between $\theta_{i,j}$ and the total number of clouds of class $m > i$ for $L_{m,k} \supset L_{i,j}$. Hence the number of clouds of class i to purchase is

$$n_{i,j}^* = \begin{cases} \theta_{M,0} & \text{if } i = M, \\ \theta_{i,j} - \max_{(m,k): L_{i,j} \subset L_{m,k}} \theta_{m,k} & \text{if } i < M \end{cases} \quad (16)$$

as in Algorithm 1.

We summarize some properties of λ_s^* and $D_{i,j}^+$, which directly follow from the construction process of λ_s^* and $D_{i,j}^+$ in Algorithm 1.

Lemma 1:

P1) $\lambda_s^* = 1$ or $\lambda_s^* = 0$.
P2) $\lambda_s^* = 0$ if $d_s < \theta_{i,j}$ with $s \in L_{i,j}$ and $\theta_{i,j} > 0$.
P3) $|D_{i,j}^+| \leq p_i$.
P4) $|D_{i,j}^+| = p_i$ if $\theta_{i,j} > 0$.

Lemma 2: For $L_{i_1,j_1} \subsetneq L_{i_2,j_2}$ with $\theta_{i_1,j_1} > 0$, it holds that

$$\theta_{i_1,j_1} \geq \theta_{i_2,j_2}.$$

Proof:
Note that $D_{i_1,j_1}^+ \subset D_{i_2,j_2}^+$ for $L_{i_1,j_1} \subset L_{i_2,j_2}$. Hence $\theta_{i_1,j_1} \geq \theta_{i_2,j_2} \geq 0$ by the definition of $\theta_{i,j}$ (i.e., (14)). ■

By the construction of $D_{i,j}^+$, $\theta_{i,j}$ and $n_{i,j}^*$, Algorithm 1 indeed finds the optimal solution of offline problem (3).

Theorem 1: Algorithm 1 gives the optimal solution of (3).

We give an example describing how Algorithm 1 works. The demand sequence is given in Table II and the prices and leasing periods of our example are

$$(p_1, p_2, p_3) = (1, 3, 6),$$
$$(l_1, l_2, l_3) = (1, 4, 12).$$

Table II illustrates how the values of λ_t^* and $D_{i,j}^+$ are decided sequentially in decreasing order of d_t's (We do not specify $D_{i,j}^+$ for the demands whose $\lambda_t^* = 0$). Recall that λ_t^* and $D_{i,j}^+$ are decided sequentially in decreasing order of d_t. The sorted sequence of d_t's in decreasing order is $(10, 9, 8, 6, 5, 4, 4, 3, 2, 2, 1)$. Note that the numbers of 4 and 2 appear two times. The first 4 is d_0 and the second 4 is d_{11}. Similarly, the first 2 is d_5 and the second 2 is d_{10}. With the sorted sequence of d_t's, λ value and $D_{i,j}^+$'s are decided. Since the highest demand is 10 with time index 4 ($t = 4$), we will set $\lambda_4^* = 1$, $D_{3,0}^+ = \{10\}, D_{2,1}^+ = \{10\}, D_{1,4}^+ = \{10\}$ (note that $j_2^4 = 1$ and $j_1^{10} = 10$). And then we decide λ value for the demand 9, whose time index is 9. That is, λ_9^* for $t = 9$ is decided and corresponding $D_{i,j}^+$'s is updated. And then, λ_1^* for $t = 1$ and corresponding $D_{i,j}^+$'s are updated, and so on. Note

TABLE III
OPTIMAL OFFLINE SOLUTION OF THE EXAMPLE

Time t	0	1	2	3	4	5	6	7	8	9	10	11
d_t	4	8	6	7	10	2	1	5	3	9	2	4
$D^+_{3,0}$	$D^+_{3,0}=\{10,9,8,7,6,5\}$											
$D^+_{2,j}$	$D^+_{2,0}=\{8,7,6\}$				$D^+_{2,1}=\{10,5\}$				$D^+_{2,2}=\{9\}$			
$D^+_{1,t}$	$\emptyset$	8	6	7	10	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	9	$\emptyset$	$\emptyset$
$\theta_{3,0}$	$\theta_{3,0}=5$											
$\theta_{2,j}$	$\theta_{2,0}=6$				$\theta_{2,1}=0$				$\theta_{2,2}=0$			
$\theta_{1,t}$	0	8	6	7	10	0	0	0	0	9	0	0
$n^*_{3,0}$	$n^*_{3,0}$ **= 5**											
$n^*_{2,j}$	$n^*_{2,0}$ **= 1**				$n^*_{2,1}=0$				$n^*_{2,2}=0$			
$n^*_{1,t}$	0	**2**	0	**1**	**5**	0	0	0	0	**4**	0	0

that there are exactly $6(=p_3)$ positive λ_t's and $|D^+_{i,j}| \le p_i$ for all (i,j) in Table II.

Table III gives the values of $\theta_{i,j}$ and $n^*_{i,j}$ decided by Algorithm 1 based on Table II (i.e, λ^*_t's). It can be easily checked that $d_t \ge \sum_{i=1}^{3} n^*_{i,j^t_i}$ and that the optimal cost is $5\times 6+1\times 3+(2+1+5+4)\times 1=45$.

IV. A DETERMINISTIC ONLINE CLOUD SELECTION ALGORITHM

This section considers an online setting. Recall that in online setting, as the demand d_t enters, the constraint $\sum_{i=1}^{M}\sum_{s\in L_{i,j^t_i}(t)} n_i(s) \ge d_t$ is newly created and the decision of $n_i(t)$, the number of clouds of i^{th} class to purchase at time t, is made to meet the constraint for all i where

$$\begin{aligned} L_{i,j}(t) &= L_{i,j} \cap [0,t] \\ &= \Big\{ s \mid kl_i \le s \le \min\big\{t,(k+1)l_i-1\big\}\Big\}. \end{aligned}$$

Moreover, the decisions made prior to time t cannot be revoked. Because of the irrevocable decisions and unpredictability of future demands in advance, the decision made by an online algorithm may turn out to be non-optimal later. Hence the cost determined by an online algorithm is generally bigger than the offline optimal cost of the optimization problem.

In this section, we propose Algorithm 2, an online algorithm with competitive ratio $O(M)$, that determines the number of clouds to buy at each time a new demand enters. We denote by $\mathcal{C}(\underline{d}(t))$ the cost decided by Algorithm 2 by time t with a demand vector $\underline{d}(t)=(d_0,d_1,...,d_t)$. In order to emphasize the present time, i.e., t when a decision is made, we use $\mathcal{C}^*(t), D^+_{i,j}(t), \theta^t_{i,j^t_i}$, and $\lambda^*_s(t)$ instead of $\mathcal{C}^*, D^+_{i,j}, \theta_{i,j^t_i}$, and λ^*_s of the optimization problem of (3) in the previous section. However, in Algorithm 2, we omit the time index t since all the decisions are made at time t. Hence in Algorithm 2, D^+_{i,j^t_i} is for $D^+_{i,j^t_i}(t)$ and θ_{i,j^t_i} is for θ^t_{i,j^t_i}, for example.

The main idea of Algorithm 2 is to purchase minimum $n_i(t)$ for any $i\ge 2$ for which (17) holds

$$n_i(t)+S_{i,j^t_i}(t-1)+\sum_{m>i} S_{m,j^t_m}(t) \;\ge\; \theta^t_{i,j^t_i} \qquad (17)$$

Algorithm 2 An online cloud selection algorithm

Input: $d_t>0$, a **single** demand request.
Output: $n_i(t)$ and $D^+_{i,j^t_i}(t)$ for all m.

Initialization: $j_i^{-1}=-1, \quad \forall i.$

At time t**:** $d_t>0$ enters.
for all i **do**
  $j^t_i=\lfloor \frac{t}{l_i} \rfloor$.
  if $j^t_i \ne j^{t-1}_i$ **then**
    $D^+_{i,j^t_i}=\emptyset$,
    $\theta_{i,j^t_i}=0$,
    $S_{i,j^t_i}=n_i=0$.
  end if
end for

$P=\{i\ge 2 \;\big|\; |D^+_{i,j^t_i}|=p_i\}$.

if $d_t \succ \theta_{i,j^t_i} \quad \forall i\in P$ **then**
  $\theta^{old}_{i,j^t_i}=\theta_{i,j^t_i} \quad \forall i.$

  if $P=\emptyset$ **then**
    $\hat{i}=0$.
    $D^+_{i,j^t_i}=D^+_{i,j^t_i}\cup\{d_t\} \quad \forall i.$
  else
    $\hat{i}=\min P$.
    $D^+_{i,j^t_i}=D^+_{i,j^t_i}+\{d_t\}-\{\theta_{\hat{i},j^t_i}\} \quad \forall i.$
  end if

  1) Update P : $P=\{i\ge 2 \;\big|\; |D^+_{i,j^t_i}|=p_i\}$.

  2) Sort P in decreasing order: $P=\{i_F>..>i_1\}$.

  3) Update θ_{i,j^t_i}:

$$\theta_{i,j^t_i}=\begin{cases} \min D^+_{i,j^t_i} & \text{if } i\in P, \\ d_t & \text{if } i=1 \text{ and } |D^+_{1,t}|=1, \\ 0 & \text{otherwise.} \end{cases}$$

end if

if $d_t>\sum_{i=2}^{M} S_{i,j^t_i}$ **then**

  4) Decide $n_i(t)$:
  For $i\in P\cup\{1\}$, in the decreasing order of i,

$$\begin{cases} n_i &= [\theta_{i,j^t_i}-\sum_{m\ge i} S_{m,j^t_m}]^+, \\ S_{i,j^t_i} &= S_{i,j^t_i}+n_i. \end{cases}$$

  where $[a]^+=\max\{0,a\}$.
end if

where

$$S_{i,j}(t) = \sum_{s \in L_{i,j}(t)} n_i(s).$$

Note that $S_{i,j_i^t}(t-1)$ is the total number of clouds of class i that have been purchased during the time interval $[j_i^t l_i, t-1]$ and $\sum_{m>i} S_{m,j_m^t}(t)$ is the total number of clouds of higher classes than i that are purchased during $[j_i^t l_i, t]$. Hence the sum, $n_i(t)+S_{i,j_i^t}(t-1)+\sum_{m\geq i} S_{m,j_m^t}(t)$, is the total number of clouds whose classes are no lower than i to purchase during the interval $[j_i^t l_i, t]$. Considering (15) or (16) for the offline optimal solution, we can know that (17) is a natural online setting of (15) or (16).

If (17) holds, then the constraint,

$$d_t \leq \sum_{i=1}^{N} \sum_{s \in L_{i,j}(t)} n_i(s),$$

is always satisfied since $n_1(t) = S_{1,t}(t) = \max\{0, d_t - \sum_{i=2}^{M} S_{i,j_i^t(t)}\}$ and $S_{1,j_1^t}(t-1) = S_{1,t}(t-1) = 0$.

Even though the idea of (17) is simple, its implementation is not obvious as seen in Algorithm 2 because of the correlated structures of multiple cloud classes. Note that the additional cloud purchase takes place only when $d_t > \sum_{i\geq 2} S_{i,j_i^t}(t-1)$. To implement the idea (17), we need the information $\theta_{i,j}^t$'s that are decided from $D_{i,j}^+(t)$ with the demand sequence $(d_0, d_1, ..., d_t)$. To get correct $\theta_{i,j}^t$ at time t, we need to update $\theta_{i,j}^t$ at each time d_t enters, regardless of $d_t > \sum_{i\geq 2} S_{i,j_i^t}(t-1)$. For this, we consider when to update $\theta_{i,j}^t$. The update of $\theta_{i,j}^t$ is necessary if there is at least one (i,j) pair such that $\theta_{i,j}^t \neq \theta_{i,j}^{t-1}$. Note that the case $\theta_{i,j}^t \neq \theta_{i,j}^{t-1}$ happens only if $d_t \succ \theta_{i,j_i^t}^{t-1}$, which is line 15 in Algorithm 2. Once such case happens, we update $D_{i,j_i^t}^+$ and get new $\theta_{i,j_i^t}^t$. After update of $\theta_{i,j_i^t}^t$, we decide the number of clouds to purchase.

Note that P at line 26 of Algorithm 2 has the following property that $\theta_{i,j_i^t}^t > \theta_{i,j_i^t}^{t-1}$ for any $i \in P$. Moreover, Algorithm 2 has the following property, Lemma 3, which comes directly from (17).

Lemma 3:

$$d_t > \sum_{i=2}^{M} S_{i,j_i^t}(t-1) \quad \Rightarrow \quad d_t > \theta_{i,j_i^t}^{t-1} \quad \forall\ i \geq 2.$$

The main result of this section is Theorem 2 whose proof is omitted due to space limit (the proof can be found in [15]).

Theorem 2: The competitive ratio of Algorithm 2 is $O(M)$. More specifically, for the one single highest level interval $L_{M,0}(t)$ (i.e, $T = l_M$),

$$\mathcal{C}^*(\underline{d}(t)) \leq \mathcal{C}(\underline{d}(t)) \leq M\mathcal{C}^*(\underline{d}(t))$$

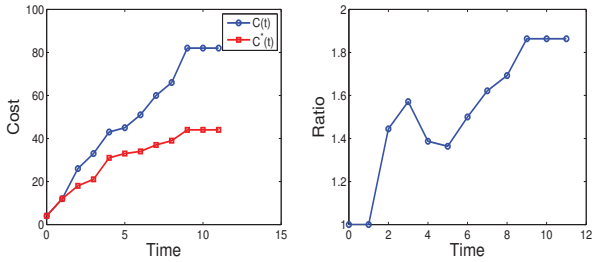
for any demand $\underline{d}(t)$ and any $0 \leq t \leq l_M - 1$.

The analysis for competitive analysis focuses only on the single highest level interval. When restricted to one single highest interval, the performance gap is M, but for the general case $T > l_M$ (for the case of multiple number of highest level intervals) the competitive ratio is $O(M)$. Note that M represents the performance gap for *the worst case* for the single highest interval. Therefore for most cases, the algorithm exhibits much smaller performance gap than M in the highest single interval.

For the special case of on-off demands, (i.e., d_t is either 0 or 1 for all t), our proposed algorithm is identical with the deterministic algorithm for the parking permit problem proposed in [11] that has M performance gap for one single $L_{M,0}$ interval and $O(M)$-competitive ratio for multiple intervals of class M. Since on-off demand is a special case of multiple demands, we can easily know that the competitive ratio of Algorithm 2 cannot be smaller than M (or $O(M)$). However, in many cases, the performance gap of multiple demands is much smaller than the performance gap of on-off demand since the workload with multiple demands is smoother than that of the on-off demand case (i.e., multiplexing effect).

We illustrate $n_i(t)$ for each $t \in [0, T-1]$ decided by Algorithm 2 in Table IV for the same demand vector in Section III. The values of $P, D_{i,j_i^t}^+(t)$, and $n_i(t)$ are specified to help the understanding of Algorithm 2. It can also be easily checked that $d_t \geq \max_{i\geq 2} \theta_{i,j_i^t}^{t-1}$ when $d_t > \sum_{i=2}^{M} S_{i,j_i^t}(t-1)$.

Figure 2(a) shows how the costs of $C(\underline{d}(t))$ and $C^*(\underline{d}(t))$ of the given example change with time t. The ratio of $\frac{C(\underline{d}(T-1))}{C^*(\underline{d}(T-1))}$ at each time t is given in Figure 2(b). Note that the maximum of $\frac{C(\underline{d}(T-1))}{C^*(\underline{d}(T-1))}$ takes places at time $t = 9$ with the value of $\frac{82}{45} = 1.86$, which is much smaller than $M = 3$, as discussed in the above.



(a) $\mathcal{C}^*(\underline{d}(t))$ and $C(\underline{d}(t))$ (b) $\frac{C(\underline{d}(t))}{C^*(\underline{d}(t))}$

Fig. 2. Cost comparison of $\mathcal{C}^*(\underline{d}(t))$ and $C(\underline{d}(t))$

V. A Randomized Online Selection Algorithm

This section proposes a randomized online algorithm that has better performance gap than Algorithm 2, a deterministic online algorithm. One observation of Algorithm 2 is that the algorithm purchases clouds of high classes after realizing that buying high class clouds can reduce the cost. Hence if we purchase more clouds of high classes earlier than Algorithm 2, then the performance gap can be reduced.

For this, we will relax the optimization problem by considering fractional number of clouds instead of integer number

TABLE IV
OUTPUT OF ALGORITHM 2 FOR THE EXAMPLE

Time slot t	0	1	2	3	4	5	6	7	8	9	10	11
Demands d_t	**4**	**8**	**6**	**7**	**10**	**2**	**1**	**5**	**3**	**9**	**2**	**4**
$\sum_{i\neq 1} S_{i,j_i^t}(t-1)$	0	0	0	4	0	0	0	1	2	3	5	5
P (line 13)	$\emptyset$	$\emptyset$	$\emptyset$	{2}	$\emptyset$	$\emptyset$	$\emptyset$	{2,3}	{3}	{3}	{3}	{3}
$\hat{i}$ (line 18)				2				2	3	3		
D^+_{3,j_3^t} (line 20 or 23)	{4}	{8,4}	{8,6,4 }	{8,7,6 }	{10,8,7, 6}	{10,8,7, 6,2}	{10,8,7, 6,2,1}	{10,8,7, 6,5,2}	{10,8,7, 6,5,3}	{10,9,8, 7,6,5}		
D^+_{2,j_2^t}	{4}	{8,4}	{8,6,4}	{8,7,6}	{10}	{10,2}	{10,2,1}	{10,5,2}	{3}	{9,3}		
$D^+_{1,t}$	{4}	{8}	{6}	{7}	{10}	{2}	{1}	{5}	{3}	{9}		
P (line 26)	$\emptyset$	$\emptyset$	{2}	{2}	$\emptyset$	$\emptyset$	{2,3}	{2,3}	{3}	{3}		
$\theta_{3,0}$ (line 29)	0	0	0	0	0	0	1	2	3	5		
θ_{2,j_2^t}	0	0	4	6	0	0	1	2	0	0		
$\theta_{1,t}$	4	8	6	7	10	2	1	5	3	9		
$n_3(t)$ (line 30)	0	0	0	0	0	0	1	1	1	2		
$n_2(t)$	0	0	4	2	0	0	0	0	0	0		
$n_1(t)$	4	8	2	1	10	2	0	3	0	4		

of clouds. Specifically, we consider the following fractional primal optimization problem

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{M}\sum_{j=0}^{j_i^t} x_{i,j}p_i \qquad (18)\\ \text{subject to} \quad & \sum_{i=1}^{M} \frac{x_{i,j_i^s}}{d_s} \geq 1 \quad \text{for all } 0 \leq s \leq t. \end{aligned}$$

Comparing (18) with (3), we find that τ_i has changed to j_i^t, $T-1$ to t and n_{ij} to $x_{i,j}$. We use x_{ij} rather than N_{ij} to emphasize that the value of x_{ij} can be a fractional number. Since we consider an online setting and L_{i,j_i^t} is the interval containing t, τ_i has changed to j_i^t and $T-1$ to t. The important point in this optimization problem is that the variables $x_{i,j}$'s are required to be non-decreasing during the whole execution of an online algorithm, because the purchase of clouds made in the past cannot be revoked.

Using the primal-dual approach proposed by [12], [16], Algorithm 3 is proposed. The main idea of the primal-dual approach is that if $\sum_{i=1}^{M} x_{i,j_i^t} < d_t$, then x_{i,j_i^t} and the dual variable y_t are increased for all i until that $\sum_{i=1}^{M} x_{i,j_i^t} > d_t$. In the algorithm, y_t increases by the amount of d_t for each time x_{i,j_i^t} increases. The increase of the dual variables y_t decides the competitive ratio of the fractional online algorithm.

Theorem 3 states the competitive ratio of Algorithm 3.

Theorem 3: Algorithm 3 is $O(\log(M) + \log(d_{\max}))$-competitive where $d_{\max} = \max d_t$ for a given demand sequence.

Proof:

Algorithm 3 A randomized online cloud selection algorithm

Input: $d_t > 0$, a **single** demand request
Output: $x_{i,j}(t)$ for all (i,j) pairs.

1: **Initialization:**
2: $x_{i,j} = 0$ for all (i,j) pairs.
3: $y_t = 0$ for all t.
4:
5: **At time t: d_t enters**
6: $y_t = 0$
7:
8: **if** $\sum_{i=1}^{M} x_{i,j_i^t} < d_t$ **then**
9: **while** $\sum_{i=1}^{M} x_{i,j_i^t} > d_t$ **do**
10: (a) $x_{i,j_i^t} \leftarrow x_{i,j_i^t}(1+\frac{1}{p_i}) + \frac{1}{Mp_i}$ for all i.
11: (b) $y_t \leftarrow y_t + d_t$.
12: **end while**
13: **end if**

The dual problem of (18) is

$$\begin{aligned} \text{Maximize} \quad & \sum_{s=0}^{t} y_s \qquad (19)\\ \text{subject to} \quad & \sum_{s\in L_{i,j}(t)} \frac{y_s}{d_s} \leq p_i \quad \text{for all } (i,j) \text{ pairs.} \end{aligned}$$

Let $P(t)$ the value of the primal optimization problem (18) and $D(t)$ the value of the dual problem (19) at time t. The process of $x_{i,j_i^t} \leftarrow x_{i,j_i^t}(1+\frac{1}{p_i}) + \frac{1}{Mp_i}$ is called by an augmentation [16]. We will consider the increases of the primal value and the dual value at one single augmentation process, denoted by ΔP and ΔD respectively.

If $\sum_{i=1}^{M} x_{i,j_i^t} < d_t$, then for one single augmentation,

$$\Delta D \quad = \quad d_t,$$

$$
\begin{aligned}
\Delta P &= \sum_{m=1}^{M} \Delta x_{i,k_i^t - p_i} = \sum_{m=1}^{M} \big(x_{i,j_i^t}^{(i-1)} \frac{1}{p_i} + \frac{1}{M}\big) p_i \\
&= \sum_{m=1}^{M} x_{i,j_i^t}^{(i-1)} + \sum_{i=1}^{M} \frac{1}{M} \\
&< d_t + 1 \le 2d_t \qquad (\because d_t \ge \sum_{i=1}^{M} x_{i,j_i^t}) \\
&\le 2\Delta D
\end{aligned}
$$

Suppose that the number of augmentation is h_t for each t. Let $x_{i,j_i^t}^{(0)}$ be the value of x_{i,j_i^t} before any augmentation at time t. Then after h_t augmentations, we have

$$
\begin{aligned}
x_{i,j_i^t} &= x_{i,j_i^t}^{(h_t)} \\
&= x_{i,j_i^t}^{(0)} (1 + \frac{1}{p_i})^{h_t} + \sum_{m=0}^{h_t - 1} \frac{1}{M p_i} \Big(1 + \frac{1}{p_i}\Big)^m \\
&= \sum_{m=0}^{H(t)-1} \frac{1}{M p_i} \Big(1 + \frac{1}{p_i}\Big)^m \\
&= \frac{1}{M} \Big(1 + \frac{1}{p_i}\Big)^{H(t)} \\
&\approx \frac{1}{M} e^{\frac{H(t)}{p_i}} \quad (\because (1+x)h \approx e^{\frac{h}{x}} \ \text{ if } x \le 1)
\end{aligned}
$$

where $H(t) = \sum_{s \in L_{i,j}(t)} h_s$.

Note that $\sum_{i=1}^{M} x_{i,j_i^t} < d_t$ is kept during any augmentation process. Hence

$$
\begin{aligned}
& x_{i,j_i^t}^{(h_t)} = \frac{1}{M} e^{\frac{H(t)}{p_i}} \le d_t \Big(1 + \frac{1}{p_i}\Big) \\
\Rightarrow \quad & H(t) = \sum_{s \in L_{i,j}(t)} h_s \le p_i \log(2 M d_t)
\end{aligned}
$$

Note that $h_t = \frac{y_t}{d_t}$ after finishing all augmentations at t. Hence $\sum_{s \in L_{i,j}(t)} \frac{y_s}{d_s} \le O(\log M + \log d_{\max})$.

$$
\begin{aligned}
P(t) &\le 2 \sum_{s=0}^{t} \Delta D(s) = 2D(t) \\
&\le 2D^*(t) O(\log M + \log d_{\max}) \\
&\le 2P^*(t) O(\log M + \log d_{\max})
\end{aligned}
$$

since $D^*(t) \le P^*(t)$ (see [17]). ■

Since $x_{i,j}$ of Algorithm 3 is a fractional number, we need to round the fractional solution $x_{i,j}$ into an integer value. This can be done in a deterministic way or in a random way [18]. If using randomized rounding such as [11], [12], Algorithm 3 is transformed into a randomized (integer) algorithm.

VI. Summary

In this paper, we consider the decision problem of which cloud classes and how many of clouds are purchased to satisfy multiple unpredictable demands and to minimize overall cost when multiple classes of clouds with diverse leasing periods are available. This problem is a fundamental one for any one who does not have her own infrastructure but leases the resources from (cloud) providers and offer her own service using the leased resources to her customers. We find the optimal solution of the cost minimization problem with the interval model and propose an online deterministic algorithm with $O(M)$-competitive ratio using the optimal solution and a fractional $O(\log M + \log d_{\max})$-competitive algorithm using the primal-dual approach. Our fractional model does not utilize the structure of the offline optimal solution. If we fully utilize the structure of the offline solution, this may result in smaller competitive ratio such as $O(\log M)$ than $O(\log M + \log d_{\max})$. We leave another proposal of a randomized algorithm exploiting the offline optimal solution structure achieving smaller competitive ratio as a future work.

References

[1] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator, "Competitive snoopy caching," in *Proc. Symposium on Foundations of Computer Science*, 1986.

[2] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, "Competitive randomized algorithms for non-uniform problems," in *Proc. ACM SIAM Symposium on Discrete Algorithms*, 1990.

[3] R. Kodialam, "Competitive algorithms for an online rent or buy problem with variable demand," https://www.siam.org/students/siuro/vol7/S01303.pdf, 2014.

[4] L. Ai, X. Wu, L. Huang, L. Huang, P. Tang, and J. Li, "The mulit-shop ski rental problem," in *Proc. ACM SIGMETRICS*, 2014.

[5] A. Khanafer, M. Kodialam, and K. P. N. Puttaswamy, "The constrained ski-rental problem and its application to online cloud cost optimization," in *Proc. IEEE INFOCOM*, 2013.

[6] W. Wang, B. Li, and B. Liang, "To reserve or not to reserve: optimal online multi-insntance acquisiont in iaas clouds," in *Proc. USENIX International Conference on Autonomic Computing (ICAC'13)*, 2013.

[7] X. Hu, A. Ludwig, A. Richa, and S. Schmid, "Competitive strategies for online cloud resource allocation with discounts," in *Proc. IEEE International Conference on Distributed Computing Systems*, 2008.

[8] Z. Lotker, B. Patt-Shamir, and D. Rawitz, "Rent, lease or buy," in *Proc. the 25th International Symposium on the Theoretical Aspects of Computer Science (STACS '08)*, 2008.

[9] H. Fujiwara and K. Iwama, "On the best possible competitive-ratio for multislope ski rental," *Journal of Combinatorial Optimization*, vol. 31, no. 2, pp. 463–490, 2016.

[10] R. Fleischer, "On the bahncard problem," *Theoretical Computer Science*, vol. 268, no. 1, pp. 161–174, 2001.

[11] A. Meyerson, "The parking permit problem," in *Proc. the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS '05)*, 2005.

[12] N. Buchbinder and J. Naor, "The design of competitve online algorithms via a primal-dual approach," *Foundations Trends in Theoretical Computer Science*, vol. 3, no. 2-3, pp. 93–263, 2009.

[13] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge University Press, 1955.

[14] B. M. Anthony and A. Gupta, "Infrastructure leasing problems," in *Proc. International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2007.

[15] Y. Jin, M. Hayashi, and A. Tagami, "Online algorithms for optimal cloud selection with multiple demands," KDDI Research, Inc. Technical Report, 2018.

[16] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naors, "The online set cover problem," in *Proc. Annual ACM Symposium on the Theory of Computing (STOC '03)*, 2003.

[17] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, 2004.

[18] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge University Press, 2011.