To Sell or Not To Sell: Trading Your Reserved Instances in Amazon EC2 Marketplace

Shengsong Yang[1], Li Pan[1*], Qingyang Wang[2], Shijun Liu[1*]
[1]School of Computer Science and Technology, Shandong University
Jinan 250101, China
[2]School of Electrical Engineering and Computer Science, Louisiana State University
Baton Rouge LA 70803, USA
Email: yss_sdu@sina.com, panli@sdu.edu.cn, qwang26@lsu.edu, lsj@sdu.edu.cn

***Abstract*—Recently, Amazon EC2 offers a reserved instance marketplace, where cloud users can sell their idle reserved instances varying in contract lengths and pricing options for avoiding the waste of their unused reservations. However, without knowing the future demands, it is hard for users to determine how to sell instances optimally, for it would incur more cost if new demands arrive after selling their reserved instances. For dealing with this problem, in this paper we first propose three online selling algorithms to guide cloud users in making decisions whether or not to sell their reservations in Amazon EC2 marketplace while guaranteeing competitive ratios. We prove theoretically that the three proposed online algorithms can guarantee bounded competitive ratios, whose values are specific to the type of reserved instances under consideration. Specifically, for all standard instances (Linux, US East) for 1-year terms in Amazon EC2, compared with a benchmark optimal offline algorithm, our algorithm $A_{3T/4}$ can achieve a ratio of $2-\alpha-a/4$ in managing instance purchasing cost, where α is the entitled discount due to reservation and a is the selling discount specified by the user who sells its reservations. Finally, through extensive experiments based on workload data collected from real-world applications, we validate the effectiveness of our online instance selling algorithms by showing that it can bring significant cost savings to cloud users compared with always keeping their reservations in Amazon EC2 reserved instance marketplace.**

***Index Terms*—cloud; IaaS; online algorithm; competitive analysis; cost management.**

I. Introduction

Nowadays, there are more and more users turning to public cloud platforms for acquiring Infrastructure-as-a-Service (IaaS) offerings. In the first half of 2017, the worldwide public cloud market revenue reached $63.2 billion with an increase of 28.6%. At the same time, IaaS costs begins to dominate a large population in an enterprise's IT investments. Thus cost management has become an increasingly urgent concern. Currently, IaaS markets such as Amazon EC2 [1], ElasticHosts [2] and Google Cloud Platform [3], mainly support two pricing schemes, which are the subscription scheme for using reserved instances and the pay-as-you-go scheme for using on-demand ones. When a user launches on-demand instances, the user is only charged for the incurred instance-hours. When a user launches reserved instances, the user first needs to *prepay* a certain upfront fee, and then the user can get a cheaper hourly rate than corresponding on-demand instances [4]. Table I gives an example of pricing rules for reserved and on-demand instances provided by Amazon cloud platform.

TABLE I
Pricing of the d2.xlarge instance (US East (Ohio), Linux) in Amazon EC2, as of Jan. 1, 2018

Payment Option	Upfront	Monthly	Effective Hourly
No Upfront	$0	$293.46	$0.402
Partial Upfront	$1506	$125.56	$0.344
All Upfront	$2952	$0	$0.337
On-Demand		$0.69 per Hour	

From the perspective of cost management, when workloads are stable, users would like to reserve instances. But for sporadic workloads, choosing on-demand instances instead generally costs less. Thus future demand curves are needed to be predicted for making instance purchase decisions. However, to a cloud user, it is difficult to predict future workloads accurately. For dealing with these issues, there exist online algorithms to guide users in purchasing instances. For example, one online algorithm has been proposed to guide users when they should reserve instances instead of purchasing on-demand instances and how many instances they should reserve [5]. Although these online algorithms have a satisfactory performance in guiding users in purchasing instances economically, there may still exist the waste of reservations, which happens when few demands arrive after reserving an instance, or the reservations still have large remaining period when users' jobs are finished. In this situation, the best choice for users is to sell these reserved instances to cover their upfront fee.

Recently, Amazon EC2 has launched a reserved instance marketplace, where cloud users can sell their unused reserved instances to limit losses. But without any prior knowledge of future demands, it is hard for a cloud user to make the decision whether to sell its reserved instances or not. It would incur more loss if new demands arrive after selling reserved instances, because for fulfilling these demands the user needs to purchase new on-demand instances or reserve new instances which will incur extra prepaid expenses. Thus, they need to carefully answer two important questions for each reserved instance: (1) whether should I sell this reserved instance,

*To whom correspondence should be addressed.

and (2) when should I sell this reserved instance? Currently, there is no corresponding algorithm to guide users in selling reservations in Amazon EC2 reserved instance marketplace.

In this paper, for addressing the above issues, we propose to use online selling algorithms to guide cloud users in making a decision whether or not to sell their reservations in Amazon EC2 reserved instance marketplace to guarantee competitive ratios, while users do not need to predict future demands. Specifically, we design three online algorithms $A_{3T/4}$, $A_{T/2}$ and $A_{T/4}$ for making decisions whether to sell a reserved instance at the time spot *3T/4*, *T/2* and *T/4* respectively, which denote three fourths, one half and one fourth of the reservation period T^*. We prove theoretically that the proposed three online algorithms can guarantee bounded competitive ratios, whose values are specific to the type of reserved instances under consideration. Specifically, for all standard instances (Linux, US East) for 1-year terms in Amazon EC2, compared with a benchmark optimal offline algorithm, our algorithm $A_{3T/4}$ can achieve a ratio of $2 - \alpha - a/4$ in managing instance purchasing cost, where α is the entitled discount due to reservation and $a \in [0, 1]$ is the selling discount specified by the cloud user who sells its reservations. Finally, through extensive experiments based on workload data collected from real-world applications, we validate the effectiveness of our online instance selling algorithms and show that it can save costs significantly compared with always keeping their reservations in Amazon EC2 reserved instance marketplace.

The rest of the paper is organized as follows. In Section II we discuss the related works briefly. Section III introduces the cost management problem for IaaS users. Section IV presents an online selling algorithm. In Section V, we propose two additional online algorithms. Section VI gives extensive experimental evaluations. Finally, Section VII states conclusions and future work directions.

II. Related Work

Based on two main pricing models including the pay-as-you-go model and the subscription model in public IaaS markets such as Amazon EC2, recently there have been studies [6] aiming to achieve cost savings by combining the two instance purchase options efficiently. For example, online algorithms [5] [7] have been proposed to guide users in determining to purchase on-demand instances or reserve instances. Besides, there are also great efforts in investigating cost-saving strategies relying on historic workloads [8] to make long-term predictions of future workloads [9] [10]. However, such predictions have practical limitations. For example, prediction models generally assume that workloads are relatively stable, which is not always the true situation in practice. Thus in some situations the prediction model as well as the corresponding cost-saving strategies may perform poorly.

For online algorithms [11] [12] [13], the approach of competitive analysis is often used to evaluate their performance with the term of competitive ratio [14] [15] [16]. The recently proposed online reservation algorithms can achieve satisfactory competitive ratios. For example, Wang et al. proposed an online reservation algorithm with a competitive ratio in managing IaaS costs. There are also some previous studies on analyzing competitive ratios by considering the worst case. For example, for paging problem, Cohen et al. introduced an online algorithm and analyzed its competitive ratio by considering the worse case in [15] and a new instance pricing model relying on a reselling model was proposed by Zhang et al. [17] using worse case analysis. In our work, we also analyze the competitive ratios of our proposed online algorithms by considering the worse cases.

There are some efforts on investigating more flexible reservation models for public IaaS markets [18]. Zhang et al. proposed a new pricing model for flexible instance reservation, in which a user can put forward its preferred reservation length and its required number of instances, while in today's IaaS cloud markets, users can only choose from predefined reservation packages.

There are also online algorithms [19] [17] assuming that users can sell out instances hours of their idle reserved instances to other users in a pay-per-use way and they achieve satisfactory competitive ratios. But they rely on a specific instance reselling model which is currently not supported by public IaaS cloud providers.

Based on the above discussion, the main contribution of this work lies in that we first propose online selling algorithms to guide cloud users in making a decision whether or not to sell their reservations in Amazon EC2 marketplace. Cloud users can achieve significant cost savings by choosing our online algorithms to sell their reservations in Amazon marketplace, with the cost bounded by the competitive ratios guaranteed in our online instance selling algorithms.

III. Cost Management for Cloud Users

In this section, we start off by discussing the pricing models in Amazon EC2 cloud platform, as well as the selling models for reserved instances. Then we review the actual cost calculation methods and online algorithms.

A. On-demand and Reservation IaaS Models

We first discuss the two main pricing schemes in current public IaaS cloud markets, which are the on-demand purchasing model and the reservation model.

For on-demand instances, users make payment for using instances by hour without paying any expenses in advance. For a given instance, using p to denote its hourly rate, if a user uses this instance t hours, the cost is $p * t$. Taking t2.nano in Amazon EC2 as an example, its hourly rate is \$0.0059. If using this instance for 1000 hours, the user needs to pay \$5.9.

Another pricing option is the subscription model for reserved instances. A user first needs to pay a certain upfront fee, then the user can get a cheaper hourly rate than that of corresponding on-demand instances. We use R to denote the upfront fee, and use $\alpha * p$ to denote its hourly rate, where α is

*T denotes the reservation period. For instance, when providing reserved instances, Amazon has 1-year and 3-year options, meaning T is 1 or 3 years.

the discount due to reservation. When a user uses this instance t hours, the cost is $R+\alpha*p*t$. For example, the upfront fee of a t2.nano instance is \$18, and the discount hourly rate is 0.002 within 1 year of its reservation period. In this case, the discount because of reservation is $\alpha = 0.002/0.0059 = 0.34$. Using a t2.nano instance under a contract of 1-year terms for 1000 hours will cost the user \$20.

B. The Selling Rules in Amazon EC2

According to the selling rules in Amazon EC2 reserved instance marketplace [20], a user can sell the remaining period of its unused reserved instances to other cloud users. Thus this user becomes a *seller* and from now on we call the user who buys reservations from this seller as the *buyer*. The seller charges an upfront fee for each of its reserved instance to sell. Once a reserved instance is sold out, the seller cannot use this instance anymore. The buyer pays the upfront fee to obtain the ownership of this instance and then the buyer can enjoy the cheaper hourly rate in the instance's remaining reservation period. The Amazon EC2 platform charges a service fee of 12 percent of the total upfront fee of each reserved instance sold in its marketplace, and the seller keeps the left 88 percent of the total upfront fee paid by the buyer.

When there are multiple users selling their reservations in Amazon EC2, the selling sequence is related to the upfront fee and the remaining period of the reservations for sale. For example, the marketplace sells the reserved instance with the lowest upfront fee at first to the buyer. If the buyer's request is not fulfilled, the marketplace will sell the reserved instance with the next lowest upfront fee. So to attract users and sell faster, the seller can set a discount of its required upfront fee compared the original one set by Amazon EC2. For example, the original upfront fee of the t2.nano instance in Amazon EC2 is \$18. If the user wants to sell the remaining second half reservation cycle of this instance, its upfront fee should be at most \$9. Then the seller sets 20 percent off the upfront fee to make its instance more attractive to buyers, so its upfront fee becomes \$7.2. The buyer needs to pay \$7.2, and after deducting 12 percent of the upfront fee, the seller eventually receives $\$7.2 * (1 - 0.12) = \$6.336$.

Once a deal is completed between a seller and a buyer, the seller will not have a cheaper hourly rate during this instance's remaining period. If the seller wants to use this type of instance again, it must reserve another one or pay on-demand hourly fee to use on-demand instances instead. In this situation, selling its reservations incurs extra cost to the seller, compared with just keeping their reservations. Thus the seller needs to carefully decide whether and when it should sell its reserved instances.

C. The Online Instance Selling Problem

Now we formalize the above instance selling problems. We define the time $t = 0, 1, 2...$ in hours, in accordance with Amazon EC2's hourly billing policy. At each time t, we assume that there are d_t demands arrived, n_t reserved instances newly reserved, and r_t reserved instances that can provide services. The calculation methods of n_t and r_t are as follows. At time t, when an instance is newly reserved, the value of n_t is increased by one. Then from time t to time $t + T$, the value of r_t is increased by one, representing that this reserved instance can provide service in the cycle from time t to time $t + T$. Here T denotes the reservation period.

At each time t, demand d_t means that the user needs to prepare d_t instances to provide services. The relationship between d_t and r_t falls into either of the following two situations. When d_t is larger than r_t, it means that the active reserved instances are not adequate for fulfilling all demands at time t. Thus the user needs to purchase another $d_t - r_t$ on-demand instances. Let o_t be the number of new on-demand instances purchased at time t. When the value of d_t is smaller than r_t, it means there are $r_t - d_t$ reserved instances remaining idle. We can conclude that $o_t + r_t \geq d_t$ at each time t. It ensures that all d_t instance requests are fulfilled at time t, by o_t on-demand instances and r_t active reserved ones. When selling out a reserved instance, the value of r_t is reduced by one starting from the moment that the instance is sold to the time that the reservation period is out of range.

For a user at time t, the actual cost includes the cost $o_t * p$ for buying on-demand instances, the upfront fee $n_t * R$ of the instances newly reserved and the hourly fees $r_t * \alpha * p$ of the active reserved instances. Using s_t to denote the number of instances sold at time t, the income gained from selling reservations at time t is $s_t * a * rp * R$, where $a \in [0, 1]$ is the selling discount specified by the user, and rp means remaining period of the reserved instance for sale. Thus, for this user the actual cost at time t is as follows:

$$C_t = o_t * p + n_t * R + r_t * \alpha * p - s_t * a * rp * R \quad (1)$$

Although the reserved instance marketplace in Amazon EC2 allows users to sell their instances with varying remaining reservation periods, it becomes highly difficult for users to make decisions if each instance's selling time is uncertain. As an exploratory work, in this paper we investigate the situations where the selling decisions are made at the time spot *3T/4*, *T/2* or *T/4*, where *T* denotes the reservation period. We leave the discussion of online algorithms for making selling decisions at an arbitrary time spot as our future work.

D. Measure of Competitiveness

Generally, the approach of competitive analysis [14] [15] is adopted for measuring an online algorithm's performance. The competitive ratio of an online algorithm represents the bound of the ratio between it and the optimal offline algorithm. For the instance reservations selling problem, it represents the ratio of the instance cost achieved by the online algorithm to the instance cost achieved by the optimal offline algorithm. The optimal offline algorithm is the one that achieves the lowest cost when all future demand is known.

Definition 1 (*c-Competitive*): An online instance selling algorithm A is $c-competitive$. It means that for all possible sequences of demands set $d = \{d_1, d_2...d_T\}$ and reservations set $r = \{r_1, r_2...r_T\}$, we have

$$C_A(d, r) \leq c * C_{OPT}(d, r) \quad (2)$$

where $C_A(d,r)$ is the cost of online selling algorithm A given input demands set d and reservations set r, $C_{OPT}(d,r)$ is the cost achieved by the optimal offline algorithm, and c is a constant. Thus, the smaller the value of c, the better performance our online instance selling algorithm achieves. Our research is to make our online instance selling algorithm approach the optimal solution more closely.

IV. An Online Selling Algorithm

We now present an online algorithm $A_{3T/4}$ and analyze its competitive ratio through theoretical analysis. We show that $A_{3T/4}$ can achieve a competitive ratio of $2-\alpha-a/4$ compared with an optimal offline algorithm for all standard instances for 1-year terms in Amazon EC2.

A. The Optimal Offline Selling Algorithm

We first discuss the optimal offline selling algorithm as a benchmark for measuring the performance of the online selling algorithm. We first define some states. For a reserved instance, let $t_1 = 1$ be the start time this instance is reserved and $t_2 = T$ be the end time this reserved instance is out of work. We assume that this reserved instance is sold at time t, $t \in [1,T]$. Let $\varepsilon = t/T$, $\varepsilon \in [0,1]$. The selling income is $(1-\varepsilon) * a * R$ while a reserved instance is sold at the time t, where $a \in [0,1]$ is the selling discount specified by the seller, and R is the original upfront fee for this reserved instance.

For a reserved instance, using x to denote the number of demands before it is sold, and y to denote the number of demands after selling it, their values are calculated as:

$$x = \sum_{t=1}^{\varepsilon * T} d_t, \qquad y = \sum_{t=\varepsilon * T+1}^{T} d_t \tag{3}$$

with T denoting the reservation period, and $\varepsilon * T$ denoting the time spot when this reserved instance is sold.

Based on the definition of x and y, we can conclude that in the optimal offline selling algorithm, the cost used for buying on-demand instances for satisfying these x demands, reduced by the income earned from selling the reserved instance, is still larger than the cost used for this reserved instance to satisfy these x demands. Conversely, the cost used for buying on-demand instances for satisfying these y demands, reduced by the income earned from selling the reserved instance, is less than the cost of these y demands satisfied by this reserved instance. These conclusions can be expressed as:

$$\varepsilon * R + \alpha * p * x < \varepsilon * R - a * \varepsilon * R + p * x \tag{4}$$

$$(1-\varepsilon) * R + \alpha * p * y > (1-\varepsilon) * R - a * (1-\varepsilon) * R + p * y \tag{5}$$

The left part of Eqs. (4) and (5) is the cost incurred by using this reserved instance to satisfy these demands. The right part is the cost in purchasing on-demand instances for satisfying those demands after selling this instance. From Eq. (4) and Eq. (5), we can get

$$x > \frac{\varepsilon * a * R}{p * (1-\alpha)} \tag{6}$$

$$y < \frac{(1-\varepsilon) * a * R}{p * (1-\alpha)} \tag{7}$$

The value ranges of x and y will be used later in analyzing our online algorithms' competitive ratio.

B. An Online Selling Algorithm $A_{3T/4}$

For a reserved instance *i*, our online selling algorithm $A_{3T/4}$ decides whether to sell it or not based on the number of demands x in the three fourths of its reservation period, i.e., $3T/4$. The reason is that the size of x determines its utilization. When the cost of purchasing on-demand instances for satisfying these x demands, reduced by the income earned by selling *i*, is equal to the cost of these x demands satisfied by *i* in the three fourths of its reservation period, we have the following equation:

$$\frac{3}{4} * R + \alpha * p * x = \frac{3}{4} * R - a * \frac{3}{4} * R + p * x \tag{8}$$

From Eq. (8), we can calculate

$$x = \frac{3 * a * R}{4 * p * (1-\alpha)} \tag{9}$$

Let $\beta = \frac{3*a*R}{4*p*(1-\alpha)}$. β is a break-even point in the online selling algorithm, which is vital for deciding whether to sell an instance *i* or not. We can conclude that when the number of demands is less than β during the three fourths of its reservation period, the best choice is selling instance *i* and purchasing new on-demand instances to satisfy these demands. However, in the three fourths of its reservation period, we didn't sell instance *i*. To compensate for this "mistake", at time $3T/4$ we sell it to compensate its losses in the remaining period $T/4$. Correspondingly, when the number of demands is larger than β, we should use instance *i* to satisfy these demands for achieving less cost.

The previous discussion was based on one unit of reserved instance. Our online selling algorithm should work on the situations where cloud users may hold multiple reserved instances. An important problem for a cloud user with multiple instances is how to calculate each instance's working time when these reserved instances are undifferentiated at each time. Consider that we only reserve new instances when the previous reserved instances cannot fulfill all demands. We can see that the instances newly reserved are used to provide services to the demands newly happened. Thus, the working sequence we proposed is that the reserved instance with less remaining period should be selected first when demands arrive at each time. It means that we assign higher priorities to those reserved instances with less remaining period when considering to satisfy the demands newly arrived. This also can increase the utilization of each reserved instance.

At time t, if there exists a reserved instance in its *3T/4*, we check its working time during the the three fourths of its reservation period, and sell this reserved instance if its working time is less than the break-even point. In **Algorithm 1** we present our online selling algorithm $A_{3T/4}$ in details.

In order to illustrate the online selling algorithm more intuitively, we consider an example as shown in Fig. 1. There

Algorithm 1 An Online Selling Algorithm $A_{3T/4}$

Input: the set of demands d, the set of new reserved instances n, and the set of active reservations r.

Output: the set of updated active reservations r, and the set of selling instances s.

Let $inst$ be the reserved instance which is being decided to sell or not. Let l be the number of the reserved instances whose remaining time is larger than $inst$.
At each time t, loop as follows:
if $n_{t-3T/4} == 0$ **then**
  There is no need to make decisions at this moment.
else
  for $i = 1$ to $n_{t-3T/4}$ **do**
    $l = 0$; $f = 0$;
    for $j = t - 3T/4 + 1$ to t **do**
      Update the value of l at each time: $l = l + n_j$;
      if $r_j - d_j - i + 1 > l$ **then**
        $inst$ is free at this moment: $f \leftarrow f + 1$;
      end if
    end for
    Get the working time w of $inst$ according to its free time f: $w = \frac{3*T}{4} - f$;
    if $w < \beta$ **then**
      Sell this instance: $s_t \leftarrow s_t + 1$;
      for $k = t + 1$ to $t + T/4$ **do**
        Update the number of active reserved instances in the future: $r_k \leftarrow r_k - 1$;
      end for
      for $k = t - 3T/4 + 1$ to t **do**
        Update the historical information to indicate that this instance has been processed: $r_k \leftarrow r_k - 1$;
      end for
    end if
  end for
end if
$t \leftarrow t + 1$, repeat from 2.

are two new instances named $inst_1$ and $inst_2$ which are reserved at time $t-3T/4+1$. After the time spot $t-3T/4+1$, a total of two instances named $inst_3$ and $inst_4$ are reserved. It means that there are two instances whose remaining period is less than $inst_1$ and $inst_2$. At the beginning of time spot $t-2$, the number of reservation minus the number of demands is three at this moment. So three reserved instances is free, including $inst_4$, $inst_3$, and one instance of $inst_1$ and $inst_2$ based on the instance working sequence. At time t, if we decide to sell one instance of $inst_1$ and $inst_2$, the number of the active reserved instances will be updated during the time spot $t-3T/4+1$ to $t+T/4$, which is shown by the dotted line.

C. Performance Analysis: $(2-\alpha-a/4)$-Competitiveness

We now use *OPT* to represent the optimal offline selling algorithm, and analyze the performance of our online selling algorithm $A_{3T/4}$ by using *OPT* as a benchmark.

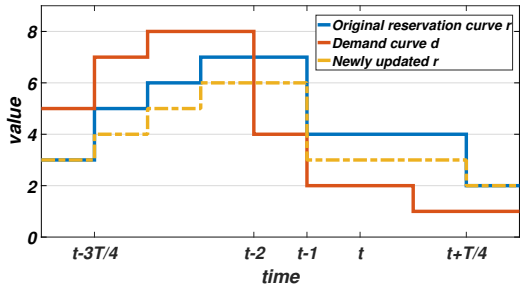

Fig. 1. An example to illustrate Algorithm $A_{3T/4}$. A reserved instance is selling at the time spot t. The dotted line denotes the changes of the reservation curve r after selling this instance.

Proposition 1: Algorithm $A_{3T/4}$ is $(2-\alpha-a/4)$-competitive. Formally, for each reserved instance that is up to sell, we have

$$C_{A_{3T/4}} \leq (2-\alpha-a/4) * C_{OPT} \tag{10}$$

where $C_{A_{3T/4}}$ is the actual cost achieved by the online selling algorithm $A_{3T/4}$ and C_{OPT} is the actual cost achieved by the optimal offline selling algorithm *OPT*.

Proof: For a reserved instance, we decide whether to sell it or not at the time spot *3T/4*, so we have $\varepsilon \in [3/4, 1]$, which represents the selling moment of the optimal offline selling algorithm. Let x_0 be the number of demands during the three fourths of the reservation period, x_1 be the number of demands during *3T/4* to $\varepsilon * T$, and x_2 be the number of demands during $\varepsilon * T$ to T. Based on the analysis of the optimal offline algorithm, we can arrive at the following conclusions.

$$x_1 > \frac{(\varepsilon - 3/4) * a * R}{p * (1-\alpha)} \tag{11}$$

$$x_2 < \frac{(1-\varepsilon) * a * R}{p * (1-\alpha)} \tag{12}$$

Because of Eq. (11), we can calculate the bound of the cost achieved by *OPT* as follows:

$$\begin{aligned} C_{OPT} &= R + \alpha * p * (x_0 + x_1) - (1-\varepsilon) * a * R \\ &\quad + p * x_2 \\ &> R + \alpha * p * x_0 + \frac{\alpha * (\varepsilon - 3/4) * a * R}{1-\alpha} \\ &\quad - (1-\varepsilon) * a * R + p * x_2 \\ &= (1-a) * R + \frac{\varepsilon * a * R}{1-\alpha} - \frac{3 * \alpha * a * R}{4 * (1-\alpha)} \\ &\quad + \alpha * p * x_0 + p * x_2 \end{aligned} \tag{13}$$

According to our online selling algorithm $A_{3T/4}$, if $x_0 < \beta$, we will sell this reserved instance at time *3T/4* to compensate its losses in the remaining period *T/4*. Conversely, if $x_0 > \beta$, we will continue to use this reserved instance in the remaining period *T/4*. Then we analyze the situation in two cases.

Case 1: In this case, we consider

$$x_0 < \frac{3 * a * R}{4 * p * (1-\alpha)} \tag{14}$$

The actual cost incurred by $A_{3T/4}$ is calculated as follows:

$$C_{A_{3T/4}} = R + \alpha * p * x_0 - \frac{a * R}{4} + p * (x_1 + x_2) \quad (15)$$

$C_{A_{3T/4}} - C_{OPT}$ represents the gap between $A_{3T/4}$ and the optimal offline selling algorithm *OPT*, as:

$$\begin{aligned} & C_{A_{3T/4}} - C_{OPT} \\ &= R + \alpha * p * x_0 - \frac{a * R}{4} + p * (x_1 + x_2) - R \\ &\quad - \alpha * p * (x_0 + x_1) + (1 - \varepsilon) * a * R - p * x_2 \\ &= \frac{3 * a * R}{4} - \varepsilon * a * R + (1 - \alpha) * p * x_1 \end{aligned} \quad (16)$$

According to the Eq. (11), we know that x_1 and ε are positively related. Besides, we can define

$$x_1 = \gamma_1 * \frac{(\varepsilon - 3/4) * a * R}{p * (1 - \alpha)} \quad (17)$$

where $\gamma_1 > 1$. Then we can get

$$\begin{aligned} & C_{A_{3T/4}} - C_{OPT} \\ &= \frac{3 * a * R}{4} - \varepsilon * a * R + (1 - \alpha) * p * x_1 \\ &= \frac{3 * a * R}{4} - \varepsilon * a * R + \gamma_1 * (\varepsilon - 3/4) * a * R \\ &= (1 - \gamma_1) * \frac{3 * a * R}{4} + (\gamma_1 - 1) * \varepsilon * a * R \end{aligned} \quad (18)$$

The coefficient of ε is $\gamma_1 - 1$, which is a positive number because $\gamma_1 > 1$. We can conclude that $C_{A_{3T/4}} - C_{OPT}$ and ε are positively related. The gap of the actual cost between $A_{3T/4}$ and *OPT* reaches its maximum when $\varepsilon = 1$. In this case, the cost performance of $A_{3T/4}$ is lowest, and thus the competitive ratio reaches a maximum.

$$\begin{aligned} \frac{C_{A_{3T/4}}}{C_{OPT}} &= \frac{R + \alpha * p * x_0 - \frac{a*R}{4} + p * (x_1 + x_2)}{R + \alpha * p * (x_0 + x_1) - (1 - \varepsilon) * a * R + p * x_2} \\ &< \frac{R + \alpha * p * x_0 - \frac{a*R}{4} + p * x_1}{R + \alpha * p * (x_0 + x_1)} && (19) \\ &< \frac{R - \frac{a*R}{4} + (1 - \alpha) * p * x_1}{R} && (20) \\ &< \frac{R - \frac{a*R}{4} + (1 - \alpha) * \frac{\theta * R}{4}}{R} && (21) \\ &= 1 + \frac{\theta * (1 - \alpha)}{4} - \frac{a}{4} && (22) \end{aligned}$$

where $\theta = C/R$. R represents the upfront fee of the reserved instance. C is the largest cost incurred by on-demand instances in the whole reservation period, in which the demands are lasting during the whole reservation period. Here, Eq. (19) holds because when $\varepsilon = 1$, $x_2 = 0$. Eq. (20) holds true because the value of competitive ratio is a fraction larger than 1, and for such a fraction, if a same value is subtracted from its denominator and numerator at the same time, its value increases. Eq. (21) holds because the maximum of $p * x_1$ is $\theta * R/4$ according the definition of θ.

Formally, for all standard instances (Linux, US East) for 1-year terms in Amazon EC2, we have made a statistic to see the value of θ. We can conclude that $\theta \in (1, 4)$. Thus we can get

$$\frac{C_{A_{3T/4}}}{C_{OPT}} < 2 - \alpha - \frac{a}{4} \quad (23)$$

Case 2: In this case, we consider

$$x_0 > \frac{3 * a * R}{4 * p * (1 - \alpha)} \quad (24)$$

The actual cost incurred by $A_{3T/4}$ is calculated as follows

$$C_{A_{3T/4}} = R + \alpha * p * (x_0 + x_1 + x_2) \quad (25)$$

Then the gap between $A_{3T/4}$ and the optimal offline selling algorithm *OPT* is:

$$\begin{aligned} & C_{A_{3T/4}} - C_{OPT} \\ &= R + \alpha * p * (x_0 + x_1 + x_2) - R - \alpha * p * (x_0 + x_1) \\ &\quad + (1 - \varepsilon) * a * R - p * x_2 \\ &= a * R - \varepsilon * a * R - (1 - \alpha) * p * x_2 \end{aligned} \quad (26)$$

According to Eq. (12), we know that x_2 is inversely proportional to ε. Besides, we can define

$$x_2 = \gamma_2 * \frac{(1 - \varepsilon) * a * R}{p * (1 - \alpha)} \quad (27)$$

where $\gamma_2 < 1$. Then we can get

$$\begin{aligned} & C_{A_{3T/4}} - C_{OPT} \\ &= a * R - \varepsilon * a * R - (1 - \alpha) * p * x_2 \\ &= a * R - \varepsilon * a * R - \gamma_2 * (1 - \varepsilon) * a * R \\ &= (1 - \gamma_2) * a * R + (\gamma_2 - 1) * \varepsilon * a * R \end{aligned} \quad (28)$$

The coefficient of ε is $\gamma_2 - 1$, which is a negative number because $\gamma_2 < 1$. We can conclude that $C_{A_{3T/4}} - C_{OPT}$ and ε are negatively related. The gap of the actual cost between $A_{3T/4}$ and *OPT* reaches its maximum when $\varepsilon = 3/4$. In this case, the actual cost incurred by $A_{3T/4}$ is lowest, and thus the competitive ratio reaches a maximum.

$$\begin{aligned} \frac{C_{A_{3T/4}}}{C_{OPT}} &= \frac{R + \alpha * p * (x_0 + x_1 + x_2)}{R + \alpha * p * (x_0 + x_1) - (1 - \varepsilon) * a * R + p * x_2} \\ &< \frac{R + \alpha * p * (x_0 + x_2)}{R + \alpha * p * x_0 - \frac{a*R}{4} + p * x_2} && (29) \\ &< \frac{R}{R - \frac{a*R}{4} + (1 - \alpha) * p * x_2} && (30) \\ &< \frac{4}{4 - a} && (31) \end{aligned}$$

Here, Eq. (31) holds because the denominator reduced by $(1-\alpha)*p*x_2$. We can see that when $\alpha + a/4 + 4/(4-a) \leq 2$, algorithm $A_{T/2}$ is $2 - \alpha - a/4$-competitive. When $\alpha + a/4 + 4/(4 - a) > 2$, algorithm $A_{T/2}$ is $4/(4 - a)$-competitive.

We have counted the discount ratio α for all standard instances (Linux, US East) for 1-year terms. Based on our statistical calculation, $\alpha < 0.36$. Thus in any cases we can conclude that $\alpha + a/4 + 4/(4 - a) < 2$, where $a \in [0, 1]$. We have

$$\frac{C_{A_{3T/4}}}{C_{OPT}} < \frac{4}{4 - a} < 2 - \alpha - \frac{a}{4} \quad (32)$$

So we have the competitive ratio of online algorithm $A_{T/2}$ as $2-\alpha-a/4$ for all standard instances (Linux, US East) for 1-year terms in Amazon EC2.

V. Two Additional Online Selling Algorithms

In this section, we propose two additional extended online selling algorithms whose selling time spots are different from algorithm $A_{T/2}$. In the first extended online selling algorithm, we assume that the remaining period of the reserved instance for sale is half of the reservation period. It means that we make the decision of whether to sell a reserved instance or not at the time spot *T/2*. Similarly, the time spot of the second extended online selling algorithm is *T/4*.

Let $A_{T/2}$ ($A_{T/4}$) denote the first (second) extended online selling algorithm and let $O_{T/2}$ ($O_{T/4}$) denote the optimal offline selling algorithm corresponding to $A_{T/2}$ ($A_{T/4}$). With the change of the time spot for making a decision, the definition of some parameters also need to make changes. We take $A_{T/2}$ as an example for illustration.

In $A_{T/2}$, the break-even spot becomes $\beta = \frac{a*R}{2*p*(1-\alpha)}$, and the remaining period of each reserved instance up to sell is *T/2*. **Algorithm 2** presents our online algorithm $A_{T/2}$ in detail.

Algorithm 2 A Deterministic Online Selling Algorithm $A_{T/2}$

```
At each time t, loop as follows:
if n_{t-T/2} == 0 then
    There is no need to make decisions at this moment.
else
    for i = 1 to n_{t-T/2} do
        l = 0; f = 0;
        for j = t - T/2 + 1 to t do
            Update the value of l at each time: l = l + n_j;
            if r_j - d_j - i + 1 > l then
                inst is free at this moment: f ← f + 1;
            end if
        end for
        Get the working time of the inst:
        w = T/2 - f;
        if w < β then
            Sell this instance: s_t ← s_t + 1;
            for k = t + 1 to t + T/2 do
                Update the number of r: r_k ← r_k − 1;
            end for
            for k = t − T/2 + 1 to t do
                Update the historical information: r_k ← r_k − 1;
            end for
        end if
    end for
end if
t ← t + 1, repeat from 1.
```

The implementation process of $A_{T/4}$, is closely similar to that of $A_{T/2}$, except the definition of some parameters as: in $A_{T/4}$, the break-even spot is $\beta = \frac{a*R}{4*p*(1-\alpha)}$, and the remaining period of each reserved instance up to sell is *3T/4*.

For all standard instances (Linux, US East) for 1-year terms in Amazon EC2, we discuss the cost performance of online algorithms $A_{T/2}$ and $A_{T/4}$ in the following cases.

Proposition 2a: Algorithm $A_{T/2}$ is $(3-2*\alpha-a/2)$-competitive, when $\alpha + a/4 + 1/(2-a) \leq 3/2$. Formally, for each reserved instance that is up to sell, we have

$$C_{A_{T/2}} \leq (3-2*\alpha-a/2)*C_{OPT} \tag{33}$$

Proof: In this case, the gap of the actual cost between $A_{T/2}$ and *OPT* reaches its maximum when $\varepsilon = 1$. In other words, the cost performance of $A_{T/2}$ is lowest, and thus the competitive ratio reaches a maximum.

$$\begin{aligned}
\frac{C_{A_{T/2}}}{C_{OPT}} &= \frac{R+\alpha*p*x_0-\frac{a*R}{2}+p*(x_1+x_2)}{R+\alpha*p*(x_0+x_1)-(1-\varepsilon)*a*R+p*x_2} \\
&< \frac{R+\alpha*p*x_0-\frac{a*R}{2}+p*x_1}{R+\alpha*p*(x_0+x_1)} && (34)\\
&< \frac{R-\frac{a*R}{2}+(1-\alpha)*p*x_1}{R} && (35)\\
&< \frac{R-\frac{a*R}{2}+(1-\alpha)*\frac{\theta*R}{2}}{R} && (36)\\
&= 1+\frac{\theta*(1-\alpha)}{2}-\frac{a}{2} && (37)
\end{aligned}$$

where $\theta = C/R$. Here, Eq. (34) holds because $x_2 = 0$ when $\varepsilon = 1$. Eq. (36) holds because the maximum of $p*x_1$ is $\theta*R/2$ according the definition of θ.

Formally, for all standard instances (Linux, US East) for 1-year terms in Amazon EC2, we can conclude that $\theta \in (1,4)$. Thus we can get

$$\frac{C_{A_{T/2}}}{C_{OPT}} < 3-2*\alpha-\frac{a}{2} \tag{38}$$

Proposition 2b: Algorithm $A_{T/2}$ is $(2/(2-a))$-competitive, when $\alpha+a/4+1/(2-a) > 3/2$. Formally, for each reserved instance that is up to sell, we have

$$C_{A_{T/2}} \leq (2/(2-a))*C_{OPT} \tag{39}$$

Proof: In this case, we can conclude that the gap of the actual cost between $A_{T/2}$ and *OPT* reaches its maximum when $\varepsilon = 1/2$. Thus, the cost performance of $A_{T/2}$ is lowest, and the competitive ratio reaches a maximum.

$$\begin{aligned}
\frac{C_{A_{T/2}}}{C_{OPT}} &= \frac{R+\alpha*p*(x_0+x_2)}{R+\alpha*p*x_0-\frac{a*R}{2}+p*x_2} \\
&< \frac{R}{R-\frac{a*R}{2}+(1-\alpha)*p*x_2} && (40)\\
&< \frac{2}{2-a} && (41)
\end{aligned}$$

Here, we can conclude that $x_1 = 0$ when $\varepsilon = 1/2$. Eq. (40) holds with the same situation seen in Eq. (20) discussed earlier in Section IV. Eq. (41) holds because the denominator reduced by $(1-\alpha)*p*x_2$.

Proposition 3a: Algorithm $A_{T/4}$ is $(4-3*\alpha-3*a/4)$-competitive, when $\alpha+a/4+4/(12-9*a) \leq 4/3$. Formally, for each reserved instance that is up to sell, we have

$$C_{A_{T/4}} \leq (4-3*\alpha-3*a/4)*C_{OPT} \tag{42}$$

Proof: In this case, the gap of the actual cost between $A_{T/4}$ and OPT reaches its maximum when $\varepsilon = 1$. In other words, the cost performance of $A_{T/4}$ is lowest, and thus the competitive ratio reaches a maximum.

$$\begin{aligned}
\frac{C_{A_{T/2}}}{C_{OPT}} &= \frac{R+\alpha*p*x_0-\frac{3*a*R}{4}+p*(x_1+x_2)}{R+\alpha*p*(x_0+x_1)-(1-\varepsilon)*a*R+p*x_2} \\
&< \frac{R+\alpha*p*x_0-\frac{3*a*R}{4}+p*x_1}{R+\alpha*p*(x_0+x_1)} && (43)\\
&< \frac{R-\frac{3*a*R}{4}+(1-\alpha)*p*x_1}{R} && (44)\\
&< \frac{R-\frac{3*a*R}{4}+(1-\alpha)*\frac{3*\theta*R}{4}}{R} && (45)\\
&= 1+\frac{3*\theta*(1-\alpha)}{4}-\frac{3*a}{4} && (46)
\end{aligned}$$

where $\theta = C/R$. Here, Eq. (43) holds because $x_2 = 0$ when $\varepsilon = 1$. Eq. (45) holds because the maximum of $p*x_1$ is $3*\theta*R/4$ in this case.

Formally, for all standard instances (Linux, US East) for 1-year terms in Amazon EC2, we can conclude that $\theta \in (1,4)$. Thus we can get

$$\frac{C_{A_{T/4}}}{C_{OPT}} < 4-3*\alpha-\frac{3*a}{4} \tag{47}$$

Proposition 3b: Algorithm $A_{T/4}$ is $(4/(4-3*a))$-competitive, when $\alpha+a/4+4/(12-9*a) > 4/3$. Formally, for each reserved instance that is up to sell, we have

$$C_{A_{T/4}} \leq (4/(4-3*a))*C_{OPT} \tag{48}$$

Proof: In this case, we can conclude that the gap of the actual cost between $A_{T/4}$ and OPT reaches its maximum when $\varepsilon = 1/4$. Thus, the cost performance of $A_{T/4}$ is lowest, and the competitive ratio reaches a maximum.

$$\begin{aligned}
\frac{C_{A_{T/2}}}{C_{OPT}} &= \frac{R+\alpha*p*(x_0+x_2)}{R+\alpha*p*x_0-\frac{3*a*R}{4}+p*x_2} \\
&< \frac{R}{R-\frac{3*a*R}{4}+(1-\alpha)*p*x_2} && (49)\\
&< \frac{4}{4-3*a} && (50)
\end{aligned}$$

From the above discussion, we can see that the competitive ratios of $A_{T/2}$ and $A_{T/4}$ are not very good compared with $A_{3T/4}$. But we know that when the demands are stable, the earlier we make the decisions whether to sell a reserved instance or not, the better performance our online instance selling algorithm can achieve. In the following experimental section, we show that $A_{T/2}$ and $A_{T/4}$ generally perform better if the demands fluctuate little.

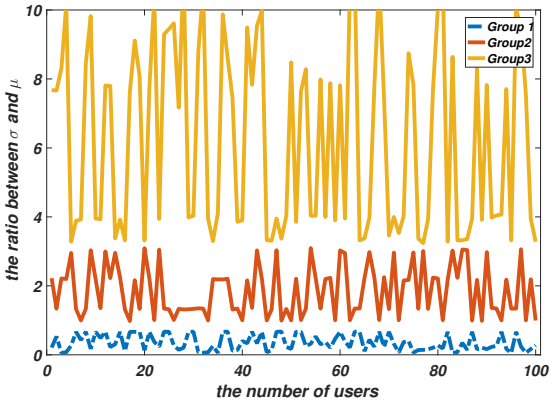

Fig. 2. The fluctuation statistics of demands (the ratio between the standard deviation σ and the mean μ) in each user group.

VI. Experimental Evaluation

Through theoretical competitive analysis, we prove our proposed online instance selling algorithms can achieve guaranteed competitive ratios. In this section, we validate their effectiveness through extensive experiments using real-world workload data.

A. Experimental Settings

The long-term user demand data we used in this work mainly includes two datasets. One contains 36 EC2 usage log files [21], and the other consists of Google cluster-usage traces [22], which contain 40 GB of workload resource requirements of more than 900 users.

The example of instance we adopted is d2.xlarge (Linux, US East) in Amazon EC2. Its upfront fee is \$1506, and on-demand fee is \$0.69. The discount α of this instance is 0.25. The reservation period is one year in our experiment.

For preprocessing the Google cluster-usage traces, we saw that the number of instances a user needs is proportional to the resources required including CPU, memory, disk and so on. Thus we used the requested number of resources recorded in Google cluster-usage traces to represent the number of instances required. As mentioned before, the actual cost performance of each online selling algorithm is greatly influenced by the fluctuation of demands. As shown in Fig. 3, we classified the 300 users selected from the original dataset into three groups according to the level of fluctuations in demands, which is measured by the standard deviation σ and the mean μ. In the first group, there are 100 users whose demands are relatively stable, with the value of $\sigma/\mu < 1$. The second group includes 100 users whose demands fluctuate slightly, with the value of $1 < \sigma/\mu < 3$. The third group includes 100 users and their demands are highly fluctuating, with the value of $\sigma/\mu > 3$. We conduct experiments with each user group to validate the effectiveness of our proposed algorithms.

After completing the preliminary preprocessing of the dataset, we need to do another important step: imitating users' behaviors to reserve instances. Our experimentation needs the dataset with the value of demands and new reserved instances

TABLE II
THE ACTUAL COST OF ONLINE ALGORITHMS FOR A USER WHOSE DEMANDS ARE HIGHLY FLUCTUATING.

	$A_{3T/4}$	$A_{T/2}$	$A_{T/4}$	*Keep-Reserved*
Cost	9.36e+04	9.40e+04	9.45e+04	9.58e+04

at each time, which is hard to gain in public IaaS clouds. To imitate users' reservation behaviors more comprehensively, we chose four online purchasing algorithms. The first is *All-reserved*, in which a user chooses reserved instances to serve all workloads, to imitate the user's reservation behavior when the demands are relatively stable. The second is random reservation, which takes a random number that is not greater than the demands' quantity as the targeted number of active reserved instances at each time. The third is an online purchasing algorithm proposed by Wang et al. [5], which help users to determine whether to purchase reserved instances or choose on-demand ones. The final algorithm is a variant of the online purchasing algorithm. To make reservations more active in this algorithm, the break-even point β is smaller than that in the third algorithm.

B. Evaluations of Online Selling Algorithms

We separately compared the three proposed online selling algorithms with two benchmark online algorithms. The first is *All-selling*, which sells all the remaining period of each reserved instance at the time point for making decisions. The second algorithm is *Keep-reserved*, in which all reserved instances are not sold. All the three proposed online selling algorithms, as well as the corresponding benchmark algorithms, were conducted for each group in the preprocessed dataset. The costs achieved by all these algorithms were normalized to *Keep-reserved*.

Cost Performance: We separately summarized the performance of three online selling algorithms including $A_{3T/4}$, $A_{T/2}$, and $A_{T/4}$ based on the dataset for all demand fluctuation levels in Fig. 3. We can see in Fig. 3a that the online selling algorithm's cost performance is better than both *All-selling* and *Keep-reserved* algorithms, but not by much. The reason is that the remaining period of each instance sold is *T/4*, which is too short to compensate the cost greatly. When switching from *Keep-reserved* to the proposed $A_{3T/4}$, more than 60% users reduce their costs. Only 1% users incur slightly more costs than before, and the growth rate of the highest cost is no more than 1%. Compared with the benchmarks, the gap of $A_{T/2}$ is bigger than that of $A_{3T/4}$. As shown in Fig. 3b, when switching from *Keep-reserved* to the proposed $A_{T/4}$, more than 70% users reduce their costs. About 40% users save more than 20% cost, which is better than that of $A_{3T/4}$. But there are about 3% users incurring slightly more costs than before. In Fig. 3c, cost savings of $A_{T/4}$ are guaranteed, with more than 75% users reduce their costs, and more than 40% users save more than 30%. However, there are about 5% users incurring more costs than before, which is worse than $A_{3T/4}$ and $A_{T/4}$.

TABLE III
AVERAGE COST PERFORMANCE OF EACH ALGORITHM (NORMALIZED TO KEEP-RESERVED).

	Group 1	Group 2	Group 3	All users
$A_{3T/4}$	0.9387	0.9154	0.9300	0.9279
$A_{T/2}$	0.8797	0.8329	0.8966	0.8643
$A_{T/4}$	0.8199	0.7583	0.8620	0.8032

Next we compared the performance of all the three proposed online selling algorithms at different demand fluctuation levels, and the results are shown in Fig. 4. Just as we have discussed in previous section, when users' demands are relatively stable in Fig. 4a and less fluctuating in Fig. 4b, the cost performance of $A_{T/4}$ is better than that of $A_{3T/4}$ and $A_{T/2}$, with the reason that there are larger remaining period of each reserved instance to compensate the cost. Besides, as we can see from Fig. 4c, when the demands are highly fluctuating, the cost performance of $A_{T/4}$ is also better than $A_{3T/4}$ and $A_{T/2}$ on average. However, when it comes to the extreme cases, as shown in Table II, $A_{3T/4}$ performs best for users whose demands are highly fluctuating. Thus, when users know their demand patterns in advance, they could achieve the least actual cost by choosing proper online selling algorithms. Of course, from the experimental results, we can see that even without any prior knowledge of future demand patterns, cloud users still can achieve a satisfactory actual cost by choosing any one of our online selling algorithms.

Then we summarized the average cost performance of all algorithms for each group of users in Table III. We can see that cost savings of our online selling algorithms are almost guaranteed. Thus, we can conclude that our three online selling algorithms can save cost significantly in all cases discussed in this paper.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we first propose online selling algorithms to guide cloud users in making a decision whether or not to sell its reservations in Amazon EC2 reserved instance marketplace without knowing future demands while guaranteeing competitive ratios. Our three online algorithms $A_{3T/4}$, $A_{T/2}$ and $A_{T/4}$ make decisions whether or not to sell a reserved instance at the time spot *3T/4*, *T/2* and *T/4* respectively, where *T* denotes the reservation period. We prove theoretically that the three proposed online algorithms can achieve guaranteed competitive ratios and their values are specific to the type of reserved instances under consideration. Specifically, for all standard instances (Linux, US East) for 1-year terms in Amazon EC2, our algorithm $A_{3T/4}$ can achieve a ratio of $2 - \alpha - a/4$ in managing instance purchasing cost, compared with an optimal offline algorithm. Through extensive experiments based on workload data collected from real-world applications, we show that our online algorithms can achieve significant cost savings to cloud users compared with always keeping their reservations in Amazon EC2 reserved instance marketplace.

In the future work we would like to design a randomized online selling algorithm, which guides users in selling their

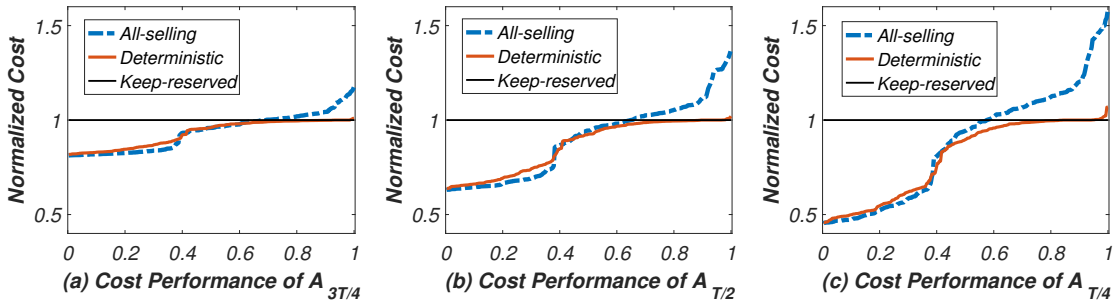

Fig. 3. The cost performance of online algorithms. All costs are normalized to Keep-Reserved.

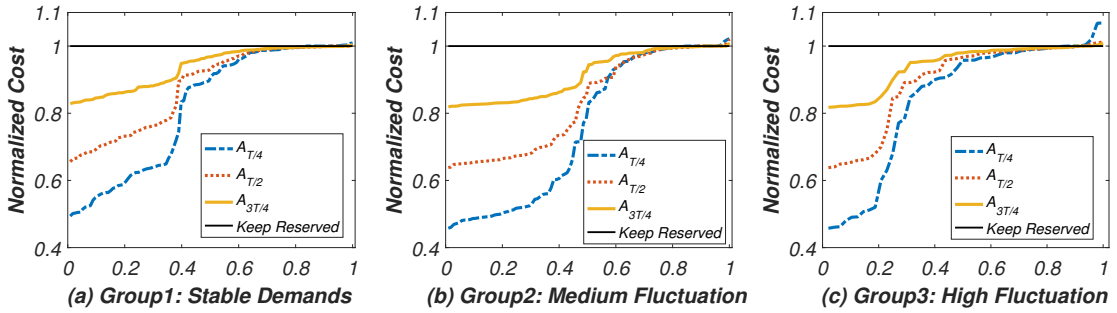

Fig. 4. The cost performance of online algorithms in different groups.

reservations at an arbitrary time spot during their reservation periods. Although it is a more challenging work, we speculate that the randomized online selling algorithm will achieve a better possible competitive ratio.

ACKNOWLEDGMENT

The authors would like to acknowledge the support provided by the National Key Research and Development Program of China (2017YFA0700601), the National Science Foundation by CISE's CNS (1566443), the Louisiana Board of Regents under grant LEQSF (2015-18)-RD-A-11, the Key Research and Development Program of Shandong Province (2018GGX101019, 2017CXGC0605, 2017CXGC0604), the Young Scholars Program of Shandong University, and the grants from Fujitsu.

REFERENCES

[1] Amazon EC2. [Online]. Available: http://amazonaws-china.com/cn/ec2/
[2] ElasticHosts. [Online]. Available: http://www.elasticchosts.com/
[3] Google Cloud Platform. [Online]. Available: http://cloud.google.com
[4] Amazon EC2 Pricing. [Online]. Available: http://amazonaws-china.com/cn/ec2/pricing/
[5] W. Wang, B. Li, and B. Liang, "To reserve or not to reserve: Optimal online multi-instance acquisition in IaaS clouds," in *Proc. Int. Conf. Autonomic Computing (ICAC 2013)*, 2013, pp. 13–22.
[6] AWS Case Studies. [Online]. Available: http://aws.amazon.com/solutions/case-studies
[7] C. Wang, W. Ma, T. Qin, F. Yang, T. Y. Liu, X. Chen, and X. Hu, "New mechanism for reservation in cloud computing," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, 2015, pp. 1765–1766.
[8] Y. J. Hong, J. Xue, and M. Thottethodi, "Dynamic server provisioning to minimize cost in an IaaS cloud," *ACM Sigmetrics Performance Evaluation Review*, vol. 39, no. 1, pp. 339–340, 2011.
[9] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, "Dynamic provisioning of multi-tier internet applications," in *Proc. Int. Conf. Autonomic Computing (ICAC 2005)*, 2005, pp. 217–228.
[10] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Capacity management and demand prediction for next generation data centers," in *Proc. IEEE Int. Conf. Web Services (ICWS 2009)*, 2009, pp. 43–50.
[11] L. Zhang, Z. Li, C. Wu, and M. Chen, "Online algorithms for uploading deferrable big data to the cloud," in *Proc. IEEE INFOCOM*, 2014, pp. 2022–2030.
[12] S. Albers, "Online algorithms," *ACM Computing Surveys*, vol. 31, no. 3es, p. 4, 2006.
[13] Y. Azar and E. Khaitsin, "Prompt mechanism for ad placement over time," in *Proc. Int. Conf. Algorithmic Game Theory*, 2011, pp. 19–30.
[14] A. Borodin and E. Y. Ran, *Online computation and competitive analysis*. Cambridge University Press, 1998.
[15] E. Cohen, H. Kaplan, and U. Zwick, "Competitive analysis of the LRFU paging algorithm," *Algorithmica*, vol. 33, no. 4, pp. 511–516, 2002.
[16] A. R. Karlin, M. S. Manasse, L. A. Mcgeoch, and S. Owicki, "Competitive randomized algorithms for non-uniform problems," *Algorithmica*, vol. 11, no. 6, pp. 542–571, 1994.
[17] S. Zhang, D. Yuan, L. Pan, S. Liu, L. Cui, and X. Meng, "Selling reserved instances through pay-as-you-go model in cloud computing," in *Proc. IEEE Int. Conf. Web Services (ICWS 2017)*, 2017, pp. 130–137.
[18] J. Zhang, W. Ma, T. Qin, X. Sun, and T. Y. Liu, "Randomized mechanisms for selling reserved instances in cloud," in *Proc. AAAI Conf. Artificial Intelligence (AAAI-17)*, 2017, pp. 750–756.
[19] W. Wang, D. Niu, B. Liang, and B. Li, "Dynamic cloud instance acquisition via IaaS cloud brokerage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1580–1593, 2015.
[20] Amazon EC2 Reserved Instance Marketplace. [Online]. Available: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ri-market-general.html
[21] Amozon EC2 File Log. [Online]. Available: http://pages.cs.wisc.edu/~keqhe/cloudmeasure_datasets.html
[22] Google Cluster-Usage Traces. [Online]. Available: http://code.google.com/p/googleclusterdata/