Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Optimal Pricing for the Competitive and Evolutionary Cloud Market

Bolei Xu
The University of Nottingham Ningbo China
Bolei.Xu@nottingham.edu.cn

Tao Qin
Microsoft Research
taoqin@microsoft.com

Guoping Qiu
The University of Nottingham Ningbo China
Guoping.Qiu@nottingham.edu.cn

Tie-Yan Liu
Microsoft Research
tyliu@microsoft.com

Abstract

We study the problem of how to optimize a cloud service provider's pricing policy so as to better compete with other providers. Different from previous work, we take both the evolution of the market and the competition between multiple cloud providers into consideration while optimizing the pricing strategy for the provider. Inspired by the real situations in today's cloud market, we consider a situation in which there is only one provider who actively optimizes his/her pricing policy, while other providers adopt a follow-up policy to match his/her price cut. To compute optimal pricing policy under the above settings, we decompose the optimization problem into two steps: (1) When the market finally becomes saturated, we use Q-learning, a method of reinforcement learning, to derive an optimal pricing policy for the stationary market; (2) Based on the optimal policy for the stationary market, we use backward induction to derive an optimal pricing policy for the situation of competition in an evolutionary market. Numerical simulations demonstrate the effectiveness of our proposed approach.

1 Introduction

In recent years, cloud computing has become very popular and been accepted by both enterprise users and personal users since it can provide economical, scalable, and elastic access to computing resources over Internet [Armbrust *et al.*, 2010]. The global cloud computing market is expected to grow at a 30% CAGR reaching \$270 billion in 2020.[1] Many firms have entered the cloud market as service providers to compete for hundreds of billions of dollars [Buyya *et al.*, 2008]. Given the fierce competition in the cloud market, it is crucial for a cloud provider to find an appropriate pricing policy to maximize the long-run profit and also remain attractive to cloud users.

[1]http://www.marketresearchmedia.com/?p=839, retrieved Nov. 5, 2014.

However, it is a very challenging task for a cloud provider to optimize his/her pricing policy mainly because of two reasons. First, the cloud market is non-stationary. The market grows very fast in recent years before eventually becoming saturated.[2] The optimization of pricing policies needs to consider market dynamics and the long-run profit of the cloud provider. Second, there are usually multiple providers offering very similar cloud services. The competition between providers who offer similar services plays an important role for pricing policy optimization.

In this work, we study the problem of pricing policy optimization in the cloud market by considering both the competition between providers and the evolution of the market. Specifically, we first model an evolutionary cloud computing market where the growth rate of the number of users is changing over different market stages and the demand of individual users is affected by the price. Then we formulate the market competition as a multiple-stage game in which we assume an active provider is competing with other reactive competitors who adopt the follow-up pricing policy. When performing pricing optimization for the active provider under the above settings, to handle the evolution of the cloud market, we decompose the optimization problem into two steps. In the first step, we consider the situation when the market evolves to saturation (stationary market) and apply a well-established method of reinforcement learning — Q-learning, to find out an optimal policy for the provider. In the second step, based on the optimal policy obtained for the stationary market, we use backward induction to find an optimal pricing policy for the situation when the market is still evolving and the environment is non-stationary. We carry out a large set of numerical experiments and show that our proposed approach can help a provider to achieve much better profit against his/her reactive opponents than other simple pricing policies.

Our main contributions are threefold:

(1) To the best of our knowledge, it is the first work to model and analyze an evolutionary cloud computing market. Rather than computing an equilibrium price for a one-shot

[2]In this paper, we assume that the cloud market will eventually become saturated. This is reasonable, since we have observed the same trend in many related markets, e.g., the PC market gradually becomes saturated in recent years.

game as in some existing works, we focus on the optimal pricing policy for a provider to adapt his/her price to different market stages and maximize the long-run profit.

(2) By combining Q-learning with backward induction, we propose a simple yet effective method to find an optimal pricing policy for a provider.

(3) We conduct a series of numerical experiments, which provide important insights into how the competition between cloud providers leads to the market evolution over stages.

2 Related work

How to optimize pricing policy for profit maximization in the cloud market has attracted much research attention recently. [Feng *et al.*, 2013] propose a non-cooperative competition model in a cloud market and target at selecting an optimal price to compete with others. However, they only focus on computing an equilibrium price for one-shot game and do not consider long-run profit maximization.

[Xu and Li, 2013] formulate the revenue maximization problem in cloud computing as a finite-horizon stochastic dynamic program with stochastic demand arrivals and departures. [Kantere *et al.*, 2011] consider the relationship between user demand and price, and solve the revenue maximization problem by a dynamic pricing scheme. However, these two works assume that there is only one provider in the cloud market and ignore the fierce competition among multiple providers in today's cloud industry.

[Vengerov, 2008] adopt a reinforcement learning approach to compute the dynamic pricing policy in the grid computing market in which customers make purchase decisions based on the posted price. [Xu and Li, 2013] extend their previous work [Xu and Li, 2012] to consider the infinite time setting in the cloud market and optimize the dynamic pricing policy also by means of reinforcement learning. However, these works only analyze how the pricing policy would influence demand and disregard the competition among providers.

[Truong-Huu and Tham, 2013] study the problem of long-run profit maximization given the competition among cloud providers and optimize the dynamic pricing policy by means of Markov Decision Process. However, they only consider a stationary market and assume that the number of users in the market is fixed and does not change over market development. Therefore their model does not match today's evolutionary cloud industry.

In our work, we consider both the competition and evolution in the cloud market which are not covered by these aforementioned works. In this regard, our work can describe today's cloud market in a more accurate and practical manner.

3 Problem Modeling

We consider a cloud market with K providers who are competing for users by offering cloud services. For simplicity, we assume there is only one type of service.[3] We formulate the competition among providers as an infinite sequential game:

- At the beginning of each stage, one proactive provider sets his/her price according to some pricing policies. The other providers adopt a follow-up pricing policy to passively update their prices.
- Cloud users purchase services from one of the providers, based on the price and their personal preferences. A user will choose none of the providers if his/her utility is negative for all the providers.

3.1 Market Evolution

Today's cloud market is facing fast growth [Zhang *et al.*, 2010] and thus in this paper we consider two kinds of growth: the increased number of cloud users and the increased demand of each individual cloud user.

Increased number of cloud users

We use the classical logistic growth function [Pearl and Reed, 1920] to model the growth of the number of cloud users:

$$N(t) = \frac{N_0 N_\infty}{N_0 + (N_\infty - N_0)e^{-\kappa t}}, \qquad (1)$$

where N_0 is the initial population, $N(t)$ denotes the number of cloud users in the market at stage t, N_∞ is the saturated population in the market, and κ is the temporal evolution rate of the market.

The above equation indicates that the growth rate of the cloud market is changing over stages. At the initial development stage of the market, users have little understanding and many concerns about cloud computing, and thus the initial population and growth rate are both low. As the market evolves, users become well educated by the market promotion of providers and the word of mouth of early cloud users, and thus users are aware of the benefit that the cloud could bring to them. In this stage, the growth of user number is approximately exponential. Then, the market is gradually saturated and the growth rate slows down, which means there are fewer new arrivals to the market since most users have already been using the cloud service in their daily work and life. Finally, the market becomes stationary and the number of users converges.

Increased demand of each user

We assume the demand of an individual user negatively correlates with the market price. Let $d_{j,t}$ denote the demand of user j at stage t, which is a random variable that follows an exponential distribution with expectation

$$\lambda_t = \frac{\Lambda}{\frac{1}{K}\sum_{i=1}^{K} p_{i,t-1}}, \qquad (2)$$

where $p_{i,t-1}$ is the price set by provider i at stage $t-1$, Λ is a constant parameter. That is, the average price of cloud services will affect the expected demand of an individual user: a higher average price in the previous stage would lead to a lower average user demand and a lower average price would increase average user demand.

When the demand of a cloud user is satisfied, he/she can extract value from the cloud service and need to pay the cloud provider. Let θ_j denote the marginal value that the user can extract from per-unit of the cloud service. We assume that the

[3] Our model and algorithms can be easily generalized to the case with multiple types of services, e.g., different configurations of virtual machines.

marginal values of all the users are independent random variables and follow a uniform distribution [Jiang *et al.*, 2007]. The utility $u^t_{j,i}$ of the user j by choosing provider i is defined as

$$u^t_{j,i} = d_{j,t}(\theta_j - p_{i,t} + \tau_{j,i}), \quad (3)$$

where $\tau_{j,i}$ denotes user j's preference towards provider i and is assumed to be a random variable that follows a gamma distribution [Ma *et al.*, 2011]. When a user enters the cloud market, he/she will choose the provider from whom he/she can obtain the largest positive utility. He/she will not choose any provider if $u^t_{j,i} \leq 0, \forall i$.

3.2 Market Competition

In today's cloud market, it is very rare for any provider to increase the price. For instance, there are five times of price reduction for the AWS EC2 m1.small instance(the oldest on-demand instance supported by Amazon) from the year of 2006 to 2014 and the price of virtual instances in Amazon EC2 never increased from its very beginning to date [4]. Therefore we assume that a provider can only cut down the price or remain the same as in the previous stage.

Provider's utility

While a provider can earn revenue from cloud users, it also needs to pay for the cost of offering cloud services. Let $c_{i,t}$ denote the marginal operational cost, which is a kind of variable cost of maintaining the infrastructure and providing the service (e.g., hardware, power, facilities, etc). It means that a provider has to pay an amount of $c_{i,t}$ to offer a unit service at stage t [Adams *et al.*, 2009]. We model the marginal operational cost as

$$c_{i,t} = c_{i,0}\Big(\sum_{j \in \mathcal{N}_{i,t}} d_{j,t}\Big)^{-\beta} e^{-\eta t}, \quad (4)$$

where $c_{i,0}$ is the initial cost of provider i, $\mathcal{N}_{i,t}$ is the set of users choosing provider i at stage t, and $\beta > 0$ and $\eta > 0$ are two parameters [Jung and Klein, 2001]. The above equation indicates that (1) when the demand received by a provider increases, the marginal operational cost decreases because of economies of scale, and (2) the marginal cost also has a temporal decaying factor which can be interpreted as the consequence of the technology development and the decreasing cost of hardware.

The immediate profit of provider i at stage t can be written as

$$r_{i,t} = \sum_{j \in \mathcal{N}_{i,t}} d_{j,t}(p_{i,t} - c_{i,t}). \quad (5)$$

The long-run profit of provider i at stage t is defined by discounting the future profit into the current stage by a fact $\gamma \in (0, 1)$, as follows:

$$R_{i,t} = \sum_{j=t}^{\infty} \gamma^{j-t} r_{i,t}. \quad (6)$$

[4]Retrieved from Amazon web services blog (http://aws.typepad.com/aws/price-reduction/)

Providers' strategies

The goal of the proactive provider is to maximize the long-run profit through an appropriate pricing policy. Without loss of generality, we assume Provider 1 is the only proactive provider. For the reactive providers, we assume that they do not optimize their utilities and just simply adopt a follow-up policy. Please note that such a follow-up policy is actually very real and is being adopted by many mainstream cloud providers today. For example, in March 2014, after Google's price reduction for its cloud services, Amazon quickly cut the price down in response and Microsoft immediately announced that they will match with Amazon's price cuts [5].

In our model, we study two kinds of follow-up policies for the reactive providers:

- *Absolute follow-up.* Reactive providers follow the absolute price cut of the proactive provider: $p_{i,t(\Delta_{1,t})} = \max(\omega, p_{i,t-1} - \Delta_{1,t})$, where $\Delta_{1,t} = p_{1,t-1} - p_{1,t}$ and ω is the threshold price, which indicates the lowest price a reactive provider is willing to sell.
- *Relative follow-up*: Reactive providers follow the relative price cut of the proactive provider: $p_{i,t(\Theta_{1,t})} = \max(\omega, p_{i,t-1}(1 - \Theta_{1,t}))$, where $\Theta_{1,t} = \frac{p_{1,t-1} - p_{1,t}}{p_{1,t-1}}$ is the relative price cut of the proactive provider at stage t.

4 Pricing Policy Optimization

In this section, we will stand on the viewpoint of the proactive provider, i.e., Provider 1, and study how to use reinforcement learning techniques to optimize his/her pricing policy for long-run profit maximization. Because we are considering an evolutionary market, we cannot directly apply reinforcement learning as done by [Truong-Huu and Tham, 2013] to our problem, and need some novel methods to spin the wheel. In particular, we notice that as the cloud market evolves, it will eventually become mature and stationary. On this basis, we can decompose the solution of an optimal pricing policy into two steps: applying reinforcement learning techniques to find an optimal pricing policy when the market becomes stationary, and then using backward induction to find an optimal pricing policy when the market is still evolving.

4.1 Q-learning for a Stationary Market

At each stage, the proactive provider needs to make a decision about how to change his/her price, based on the latest price of all the providers in the market and the market size. Thus the market environment at one stage can be represented by a tuple of K prices in the previous stage and the number of cloud users at this stage $s = (p^s_1, \cdots, p^s_K, n^s)$. Since both the price and the user number take discrete values, all the possible environments can be represented by a set of discrete states S.

From Eq. (1) we can see that when t is sufficiently large, the number of users in the cloud market will converge to N_∞, indicating that the market has become stationary. In this situation, the optimal pricing problem can be modeled as a Markov

[5]http://www.pcworld.com/article/2138320/microsoft-slashes-azure-prices-introduces-new-basic-tier.html, retrieved Oct. 20, 2014

Decision Process (MDP) and many methods such as value iteration and policy iteration [Sutton and Barto, 1998] can be used to find an optimal pricing policy. Here we adopt the Q-learning technique [Watkins, 1989] because it can enable us to easily extend our approach to solve more realistic problems in the future, especially on calculating optimal policy in the uncertain market environment. For example, when the user demand function is unknown, the reward for each state cannot be recognized beforehand by the provider. In this case, Q-learning is still able to find out an optimal policy as it only needs to know what states exist and what actions are possible in each state, while the value iteration and policy iteration cannot work any longer because they require exact reward information for each state.

Denote $Q(s, a)$ as the discounted long-run expected profit when the proactive provider takes action a in state s. We use a table to represent the $Q(s, a)$ function [Kaelbling *et al.*, 1996], which contains the value for every possible state-action pair. In our scenario, the state s is equivalent to the price of the proactive provider in the previous stage, because (1) the number of users n^s in the state is a constant when the market is stationary, and (2) the price of all the other providers can be simply inferred from the price of the proactive provider due to the follow-up policy. Similarly the action a is equivalent to the price change of the proactive provider in this stage.

Algorithm 1 shows how Q-learning works in our setting. We first initialize the table (line 1). Then we repeat the following update procedure (lines 2-17). For each state s, we randomly take an action a with probability ϵ (lines 4-6) and take the action a maximizing $Q(s, a)$ with probability $1 - \epsilon$ (lines 7-9). After taking action a in state s, the users choose providers (line 10) and the proactive provider receives an immediate reward $r(s, a)$ which is calculated by (5) and the new state is denoted as s'. Then we update $Q(s, a)$ in the table according to the following equations (lines 11-12):

$$\Delta Q(s, a) = \alpha[r(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (7)$$

$$Q(s, a) :\leftarrow Q(s, a) + \Delta Q(s, a) \quad (8)$$

where γ is the discount factor and α is the learning rate [Sutton and Barto, 1998]. We terminate the update procedure if the table converges (lines 14-16).

When we obtain the convergent table, an optimal policy can be easily found by choosing the action with the highest expected profit $Q(s, a)$ in any state s.

4.2 Backward Induction for an Evolutionary Market

After obtaining an optimal pricing policy when the market is stationary and the optimal expected profit for each state-action pair, we adopt backward induction to find an optimal pricing policy when the market is still evolving by leveraging the convergent table Q.

Let T denote a stationary stage of the market, $r(p, a, t)$ denote the immediate profit of the proactive provider at stage t by taking action a (setting new price as a) given the price p of the previous stage $t - 1$, $R(p, t)$ denotes the optimal expected profit that the proactive provider can achieve at stage t given the price p of the previous stage, and $A(p, t)$ denote the optimal action (price) that the proactive provider should take at stage t given the price p of the previous stage.

Algorithm 1 Q-learning for a stationary market

```
Input:
    The number of cloud users $N_\infty$ in the stationary market;
    The set of all possible price $\mathcal{P}$ and the set of all possible actions $\mathcal{A}$;
    Exploration rate $\epsilon$, discount factor $\gamma$, stop criteria $\delta$ ;
Output:
    A convergent table $Q$;
 1: Initialize the table $Q$ to all zeros;
 2: for $j = 1, 2, \cdots$ do
 3:     for each state $s$ do
 4:         $\xi \leftarrow$ rand(0..1)
 5:         if $\xi < \epsilon$ then
 6:             Select $a$ randomly;
 7:         else
 8:             $a \leftarrow \arg\max_a Q(s, a)$;
 9:         end if
10:         Users choose their provider by Eq. (3);
11:         Compute $\Delta Q(s, a)$ according to Eq. (7);
12:         Update $Q(s, a)$ according to Eq. (8);
13:     end for
14:     if $\max_{s,a} \Delta Q(s, a) < \delta$ then
15:         Terminate;
16:     end if
17: end for
```

Algorithm 2 shows how backward induction can find an optimal pricing policy for an evolutionary market. Specifically, by using the convergent table Q outputted by Algorithm 1, we can directly find the optimal action and expected profit at stage T for any given previous price p (lines 1-2). Then we do backward induction (lines 3-8). At each stage t, given the previous price p, we can find the optimal action and expected profit (lines 4-7) by leveraging $A(p, t + 1), \forall p \in \mathcal{P}$ and $R(p, t + 1), \forall p \in \mathcal{P}$, which have already been obtained in the previous iteration.

Algorithm 2 Backward induction for an evolutionary market

```
Input:
    Stationary stage $T$ and the convergent table $Q$ outputted by Algorithm 1;
    The initial price $p_{i,0}$ and initial cost $c_{i,0}$ of each provider $i$ before stage 1;
Output:
    An expected profit table $R(p, t), \forall p \in \mathcal{P}, \forall t \in [T]$ and an optimal action table
    $A(p, t), \forall p \in \mathcal{P}, \forall t \in [T]$;
 1: Set $R(p, T) = \max_{a \in \mathcal{P}, a \leq p} Q(p, a)$;
 2: Set $A(p, T) = \arg\max_{a \in \mathcal{P}, a \leq p} Q(p, a)$;
 3: for $t = T - 1, T - 2, \cdots, 2, 1$ do
 4:     for each $p \in \mathcal{P}$ do
 5:         Set $R(p, t) = \max_{a \in \mathcal{P}, a \leq p} \{r(p, a, t) + \gamma R(a, t + 1)\}$;
 6:         Set $A(p, t) = \arg\max_{a \in \mathcal{P}, a \leq p} \{r(p, a, t) + \gamma R(a, t + 1)\}$;
 7:     end for
 8: end for
```

By combining Algorithm 1 and Algorithm 2, we can find an optimal policy for the proactive provider in the evolutionary market.

5 Numerical Simulations

5.1 The Market with Three Providers

Parameter Settings

For the simulations in this subsection, we consider a market with $K = 3$ providers. Suppose Provider 1 is the proactive one and other two reactive ones are named as Provider

2 and Provider 3 respectively. We simulate Provider 1's performance according to three different initial market scenarios as summarized in Table 1. In Scenario 1, Provider 1 is playing as a market leader with the highest initial price and marginal cost, and the highest preference from users [6]. Similarly, Provider 1 has the second highest value for all the initial parameters (e.g., initial price, initial cost, and the shape of the distribution of users' preference) in Scenario 2 and the lowest value in Scenario 3. In each scenario, we consider Providers 2 and 3 applying either absolute follow-up policy (ABS) or relative follow-up policy (REL) as defined in Section 3.2. We set the threshold price $\omega = 0.1$ in all the scenarios.

	$p_{i,o}$	$c_{i,0}$	k_i
Scenario 1	2.0, 1.5, 1	1.0, 0.8, 0.6	0.9, 0.7, 0.5
Scenario 2	1.5, 2.0, 1	0.8, 1.0, 0.6	0.7, 0.9, 0.5
Scenario 3	1, 1.5, 2.0	0.6, 0.8, 1.0	0.5, 0.7, 0.9

Table 1: Market scenarios

For the simulations, we set the initial population number $N_0 = 100$, the saturated population number is $N_\infty = 10,000$, and the evolution rate $\kappa = 0.07$. Thus the population grows a bit slower from stage 0 to 20, experiences a fast growth from stage 20 to 100, the growth rate is decreasing after stage 100, and the population finally becomes saturated around stage 150.

We assume the users' marginal value θ_j derived from cloud service follows a uniform distribution supported on $[1, 5]$ and the users' demand follows an exponential distribution and set the parameter Λ in Eq. (2) to 2. We set $\beta = 0.01$ and $\eta = 0.02$ for provider's marginal cost function in Eq. (4). For the Q-learning algorithm, we set learning rate $\alpha = \frac{0.1}{1+0.1j}$ which is decreasing with respect to the iteration step j and the stop criteria $\delta = 1$.

Price Evolution

In this subsection, we investigate how the proactive provider changes his/her price in different scenarios of the evolutionary market. Figure 1(a,b) demonstrate how the providers change their prices as the market evolves (we record the price every five time steps). We have the following observations from the figure.

(1) In all the three scenarios, there is a fierce price competition at the initial market development stages. Provider 1 triggers price war at that market stage in order to compete for the new users to expand his/her market share so as to enjoy relatively low marginal cost with the existence of the economies of scale. When the market approaches saturation, Provider 1 gradually weakens the price reduction, because the additional users brought by further price reduction are not able to compensate for the loss of marginal profit.

(2) The difference between these three scenarios lies in that when Provider 1's initial price and preference level are higher, he/she is willing to cut the price down in a more aggressive manner. We can see that in Scenarios 1 and 2, the price is dropping down almost all the time, and the price reduction rate in Scenario 1 is more intense than in Scenario 2. In Scenario 3, price war happens only at the initial market development stages, and then the price keeps constant over the rest market stages. It means that triggering price war is not an effective strategy for the proactive provider to earn positive profit if the preference from users is lower than other competitors. This is because a service provider with a lower preference level will attract fewer users than other providers even if they offer the same price, and as a result the marginal cost of the provider will be higher than his/her competitors. Therefore, the price reduction space for a provider with a lower preference level is relatively smaller than the providers with higher preference levels, otherwise he/she will get a negative profit.

[6]We manipulate the shape parameter k_i of gamma distribution to represent different preference levels from users for Provider i.

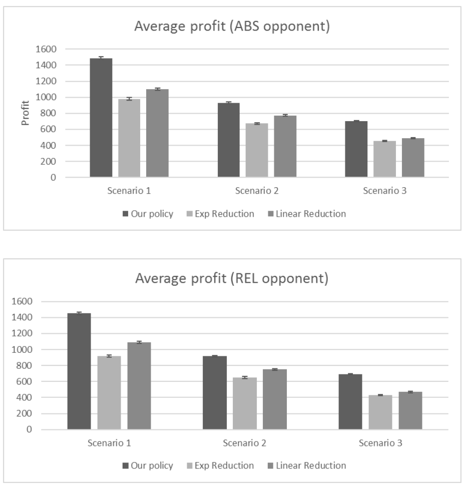
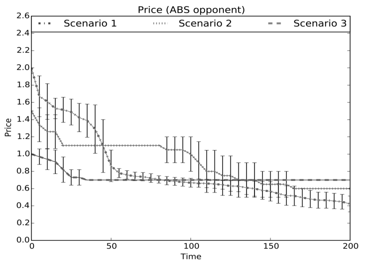


Figure 2: Profit comparison of different price reduction policies

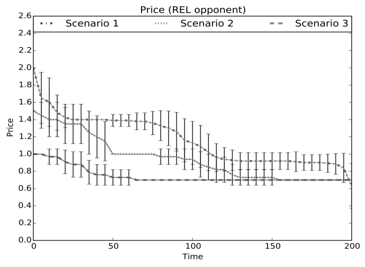
Policy Comparison

To test the performance of our pricing policy calculated by the two algorithms, we need to compare it with some baseline pricing policies. As this paper is the first one to model the evolutionary cloud computing market, there are no existing baselines, and we create two simple price reduction policies to serve as baselines. The first pricing policy asks the proactive provider to exponentially cut his/her price over stages $p_t = p_o e^{-0.01t}$, which means the price drops down quickly at the initial stages, then decreases slower until it approaches the threshold price. We call this policy "Exp Reduction". The second policy asks the proactive provider to drop the price down by 5% for every 10 steps until it approaches the threshold price. We call it "Linear Reduction". Similar to the previous experiments, we set the threshold price as 0.1.

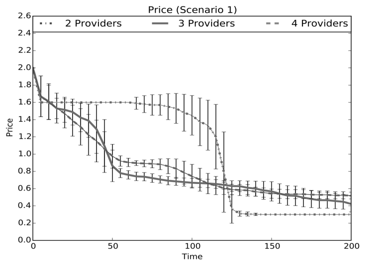
We calculate the discounted cumulative profit $R_{1,0}$ of the proactive provider using Eq. (6) and run the experiment 50 times for each scenario to compare the three policies in Figure 2. According to the figure, using the pricing policy outputted by our algorithms, the proactive provider can achieve the largest profit than using the other two policies for all the scenarios.



(a) ABS policy



(b) REL policy



(c) Multi-providers

Figure 1: Price evolution against reactive opponent

5.2 Different Numbers of Providers

In this subsection, we study that if using our policy, how the price of the proactive provider would be shaped when competing with different numbers of providers. We consider $K = 2, 3, 4$ providers in the market. The initial setting of the market is shown in Table 2. The other settings are the same as in the previous subsection. Figure 1(c) shows the price evolution of the proactive provider averaged over 50 random runs.

	$p_{i,o}$	$c_{i,0}$	k_i
$K = 2$	2, 1.5	1, 0.8	0.9, 0.7
$K = 3$	2, 1.5, 1	1, 0.8, 0.6	0.9, 0.7, 0.5
$K = 4$	2, 1.5, 1, 0.5	1, 0.8, 0.6, 0.4	0.9, 0.7, 0.5, 0.3

Table 2: Initial settings

It can be seen that the price reduction becomes more frequent with the increasing number of providers. When competing with only one competitor, there is only one explicit price war that cuts the price down aggressively when the market is almost saturated and the user base in the market becomes large. When there are two opponents in the market, the explicit price war happens at the initial market development stages (around stage 50), and the price reduction continues over market stages. When playing against three opponents, the price is dropping all the time, but the per-stage drop is not as large as that of against fewer opponents.

It should be noticed that a duopoly competition leads to the lowest final price at time step 200. As there are only two providers in the market, both of the providers will attract more demand and the final marginal cost will be lower than with more providers; consequently, the proactive provider has a larger space to cut down his/her price. In fact, this observation is quite interesting. It means that the competition with more than 2 providers does not lead to better welfare to cloud users in the saturation stage; instead, more providers will lead to market fragment. As a result, the marginal cost will be larger for each provider and consequently the price will also be higher. Similar results are also observed in [Feldman *et al.*, 2013], in which the authors find that competition among two firms can significantly increase the users' social welfare in comparison with the monopoly case, but a further increase in competition triggered by additional firms may be hazardous for the society.

6 Discussions and Future Works

In this paper, we have modeled the evolutionary cloud computing market and studied how to maximize the long-run profit of a proactive provider when competing with other reactive providers. To handle the non-stationary market, we have proposed two algorithms to find an optimal pricing policy: The Q-learning algorithm works for the situation when the market has evolved to a stationary state, and the backward induction algorithm works for the evolving and non-stationary market environment. Overall speaking, our analysis has simulated possible price evolutions in different scenarios. These results on one hand could explain the observed trend in today's cloud computing market, e.g., the fierce price competition at the initial market development stages. They, on the other hand, also provide insights on the appropriate actions that the proactive provider should take in different situations.

As this is the very first work to address the profit maximization problem in an evolutionary cloud market, we make use of some assumptions to simplify the model and the analysis. However, it is not difficult to weaken these assumptions so as to increase the practical impact of our work, as discussed below.

In the current analysis, we have assumed specific forms of the market growth function and the user demand function. However, our approach is actually generally applicable to any kinds of market growth functions and user demand functions, since our algorithms only use them to compute the immediate reward $r(s, a)$, which is only related to the number of users and the number of units of cloud services they demand at each stage. This flexibility indicates a strong practical value of our approach. When the cloud provider cumulates more knowledge about the market, he/she will have a more accurate understanding of the market growth function and the user demand function. Actually in the literature, machine learning methods have been widely adopted to predict market growth [Bose and Mahapatra, 2001] and user preference [Lai *et al.*, 2012] based on historical data, and have achieved very promising results. In this case, the output of our approach will become more precise and provide more practically useful guideline for the cloud provider to make appropriate decisions, when more historical data are available.

Furthermore, we have assumed there is only one proactive provider in the market. While it well describes the current cloud market, it is interesting to see whether our approach can go beyond the assumption and solve the optimal pricing problem in a market with multiple proactive providers. Actually we can achieve this by extending the problem into a stochastic game and leveraging the methods designed for multi-agent reinforcement learning, which are consisting of n-provider Markov decision process [Hu *et al.*, 1998]. Due to space restrictions, we leave all the aforementioned interesting extensions to the future work.

Acknowledgments

This work is partially supported by Ningbo Science and Technology Bureau (Project No 2012B10055 and 2013D10008) and by the International Doctoral Innovation Centre (IDIC) at the University of Nottingham Ningbo China.

References

[Adams *et al.*, 2009] Ian F Adams, Darrell DE Long, Ethan L Miller, Shankar Pasupathy, and Mark W Storer. Maximizing efficiency by trading storage for computation. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, pages 17–17. USENIX Association, 2009.

[Armbrust *et al.*, 2010] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[Bose and Mahapatra, 2001] Indranil Bose and Radha K Mahapatra. Business data mininga machine learning perspective. *Information & management*, 39(3):211–225, 2001.

[Buyya *et al.*, 2008] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. Ieee, 2008.

[Feldman *et al.*, 2013] Moran Feldman, Reshef Meir, and Moshe Tennenholtz. Competition in the presence of social networks: How many service providers maximize welfare? In *Web and Internet Economics*, pages 174–187. Springer, 2013.

[Feng *et al.*, 2013] Yuan Feng, Baochun Li, and Bo Li. Price competition in an oligopoly cloud market with multiple iaas cloud providers. *IEEE Transactions on Computers*, page 1, 2013.

[Hu *et al.*, 1998] Junling Hu, Michael P Wellman, et al. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, pages 242–250. Citeseer, 1998.

[Jiang *et al.*, 2007] Bao-Jun Jiang, Pei-yu Chen, and Tridas Mukhopadhyay. Software licensing: pay-per-use versus perpetual. 2007.

[Jung and Klein, 2001] Hoon Jung and Cerry M Klein. Optimal inventory policies under decreasing cost functions via geometric programming. *European Journal of Operational Research*, 132(3):628–642, 2001.

[Kaelbling *et al.*, 1996] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Kantere *et al.*, 2011] Verena Kantere, Debabrata Dash, Gregory Francois, Sofia Kyriakopoulou, and Anastasia Ailamaki. Optimal service pricing for a cloud cache. *Knowledge and Data Engineering, IEEE Transactions on*, 23(9):1345–1358, 2011.

[Lai *et al.*, 2012] Siwei Lai, Yang Liu, Huxiang Gu, Liheng Xu, Kang Liu, Shiming Xiang, Jun Zhao, Rui Diao, Liang Xiang, Hang Li, et al. Hybrid recommendation models for binary user preference prediction problem. In *KDD Cup*, pages 137–151, 2012.

[Ma *et al.*, 2011] Hao Ma, Chao Liu, Irwin King, and Michael R Lyu. Probabilistic factor models for web site recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 265–274. ACM, 2011.

[Pearl and Reed, 1920] Raymond Pearl and Lowell J Reed. On the rate of growth of the population of the united states since 1790 and its mathematical representation. *Proceedings of the National Academy of Sciences of the United States of America*, 6(6):275, 1920.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[Truong-Huu and Tham, 2013] Tram Truong-Huu and Chen-Khong Tham. A game-theoretic model for dynamic pricing and competition among cloud providers. In *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, pages 235–238. IEEE, 2013.

[Vengerov, 2008] David Vengerov. A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments. *Future Generation Computer Systems*, 24(7):687–693, 2008.

[Watkins, 1989] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

[Xu and Li, 2012] Hong Xu and Baochun Li. Maximizing revenue with dynamic cloud pricing: The infinite horizon case. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2929–2933. IEEE, 2012.

[Xu and Li, 2013] Hong Xu and Baochun Li. Dynamic cloud pricing for revenue maximization. *IEEE Transactions on Cloud Computing*, page 1, 2013.

[Zhang *et al.*, 2010] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.