Pricing Cloud Resource Based on Reinforcement Learning in the Competing Environment

Bing Shi[1,2](✉), Hangxing Zhu[1], Han Yuan[1], Rongjian Shi[1], and Jinwen Wang[1]

[1] School of Computer Science and Technology, Wuhan University of Technology, Wuhan, People's Republic of China
bingshi@whut.edu.cn

[2] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, People's Republic of China

Abstract. Multiple cloud providers compete against each other in order to attract cloud users and make profits in the cloud market. In doing so, each provider needs to charge fees to users in a proper way. In this paper, we will analyze how a cloud provider sets price effectively when competing against other cloud providers. Specifically, we model this problem as a Markov game, and then use minimax-Q and Q learning algorithms to design the pricing policies respectively. Based on this, we run extensive experiments to analyze the effectiveness of minimax-Q and Q learning based pricing policies. We find that although minimax-Q is more suitable in analyzing the competing game with multiple self-interested cloud providers, Q learning based pricing policy performs better in terms of making profits. We also find that minimax-Q learning based pricing policy performs better in terms of keeping cloud users. Our experimental results can provide useful insights on designing practical pricing policies in different situations.

Keywords: Competing cloud providers · Pricing policy
Reinforcement learning · Minimax-Q learning

1 Introduction

During the past few years, the development of cloud computing has achieved significant success in the industry since it can provide economical, scalable, and elastic access to computing resources, thus liberating people from installing, configuring, securing, and updating a variety of hardware and software [1–3]. More and more firms and personal users have been using cloud computing services over Internet, which contribute to the development of the cloud computing market. The global cloud computing market is expected to grow at a 30% compound annual growth rate (CAGR) reaching $270 billion in 2020. To compete for hundreds of billions of dollars, many firms as service providers have been participating in the cloud market [4]. Now, there exist many dominating cloud platforms offering cloud services, such as Microsoft's Azure, IBM's SoftLayer and Amazon's AWS. In the cloud market with multiple cloud providers, cloud users have various choices, and they usually participate in the provider which can satisfy their demands and charge the lowest price to them. Actually, when multiple providers offer similar quality of service [5–7], the price will significantly affect users' choices and thus providers' profits. Therefore, cloud providers need to set effective prices to compete against each other. Furthermore, the competition among providers usually lasts for a long time, i.e. the providers compete against each other repeatedly, and thus they need to maximize the long-term profits. In this paper, we will analyze how a cloud provider designs an appropriate pricing policy to maximize the long-term profit and also remain attractive to cloud users.

There exist some works on designing pricing policies for cloud providers. In [8], a non-cooperative competing model based on game theory has been proposed which computes the equilibrium price for one-shot game and does not consider the long-term profits. In [9,10], the authors assume that there is only one provider, while in today's cloud market multiple providers exist and compete against each other. Then the authors in [11,12] analyze the user behavior with respect to the providers' prices, but ignore the competition among providers. In [13], the authors analyze the pricing policy in the competing environment by assuming that there is only one proactive provider, and other providers just follow the proactive one's pricing policy. Some other works, such as [14,15], consider the competition among providers but does not capture the market dynamics, and their algorithms can only be applied to a very small market with few users.

To the best of our knowledge, few works have considered the situation of multiple providers competing against each other repeatedly. In this paper, we will analyze how the competing cloud provider sets price effectively to maximize the long-term profits in the context with two competing providers.[1] In more detail, we first describe basic settings of cloud users and providers. Specifically, we consider the uncertainty of users choosing cloud providers in the setting, which is consistent with the realistic user behavior. Furthermore, how users choosing cloud providers is affected by the prices, and how cloud providers setting prices is affected by users' choices, and therefore it is a sequential-decision problem. Moreover, this problem involves two self-interested cloud providers, and thus it is a Markov game [16]. In this paper, we model the competition between cloud providers as a Markov game, and then use two typical reinforcement learning algorithms, minimax-Q learning [17] and Q learning [18], to solve this game and design the pricing policy. We then run extensive experiments to evaluate the policies in different situations. We find that although minimax-Q learning, which was specifically designed for Markov games, is more suitable to be applied in this issue, Q learning based pricing policy performs better in terms of making profits. We also find that minimax-Q based pricing policy is better for remaining attractive to cloud users.

[1] Our model can be easily extended to the case with more than two cloud providers.

The structure of the paper is as follows. In Sect. 2, we describe basic settings of cloud users and providers. In Sect. 3, we describe how to use Q learning and minimax-Q learning algorithms to design the pricing policy. We run extensive experiments to evaluate the pricing policies in different situations in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 Basic Settings

In this section, we describe the basic settings of cloud providers and users. We assume that there are N users and two cloud providers, A and B. Cloud providers compete against each other repeatedly, i.e. the competition consists of multiple stages. At the beginning of each stage, each provider publishes its price, and then each user chooses to be served by which provider based on its choice model. According to users' choices, the two providers compute the obtained profits at the current stage, and the competition enters into the next stage.

2.1 Cloud Providers

Cloud providers can make profits by charging fees to users, while they also need to pay for the cost of offering services (e.g. power, hardware, infrastructure maintenance cost and so on). At stage t, provider i should pay for the cost of offering per unit service [19], which is denoted as $c_{i,t}$. We assume that each user only requests one-unit service. Therefore, the amount of requested service at stage t is equal to the number of users. At the beginning of the competition, the initial marginal cost of provider i is $c_{i,0}$. At stage t, the amount of users choosing provider i is $N_{i,t}$, and then at this stage, the marginal cost is:

$$c_{i,t} = c_{i,0}(N_{i,t})^{-\beta}e^{-\theta t} \tag{1}$$

This equation indicates that as more users requiring the service and as time goes, the marginal cost decreases [20]. Specifically, when the provider receives more demands of services, its marginal cost would be decreased because of economics of scale, where β is the parameter for the economics of scale, and $\beta > 0$. Furthermore, the reduction of hardware cost and the development of technology contribute to the temporal decaying factor of the marginal cost, where θ is the parameter of temporal decaying factor, and $\theta > 0$.

We assume that the price is denoted as p, and all allowable prices constitute a finite set P. The price is actually the *action* used in Sect. 3. After providers publishing the prices, users make the choices of providers. We can calculate the immediate reward of each provider, which is the immediate profit made at the current stage t:

$$r_{i,t} = N_{i,t}(p_{i,t} - c_{i,t}) \tag{2}$$

where $p_{i,t}$ is the price set by provider i at stage t.

2.2 Cloud Users

Each user has a marginal value on per-unit requested service, which is denoted as δ. At stage t, after all providers publish the prices, user j can calculate its expected revenue when entering provider i, which is:

$$R_{j,i}^{t} = \delta_j - p_{i,t} \tag{3}$$

Intuitively, based on Eq. 3, cloud users can determine in which provider they can obtain the maximal revenue at the current stage, and then choose that provider. However, in the real world, users keep requiring cloud services, and they usually take into account the prices at previous stages. Specifically, in this paper, we assume that the users will consider the prices at the current stage t and the last stage $t-1$ when choosing the cloud providers. We do not need to consider the prices at all previous stages since in this paper, the providers' prices and the users' choices are affected by each other, and thus the price of the last stage actually implies the dynamic interaction of all previous stages. Therefore, the expected utility that user j can make when entering provider i is:

$$v_{j,i}^{t} = \xi R_{j,i}^{t} + (1-\xi) R_{j,i}^{t-1} \tag{4}$$

where ξ is the weight of the price considered by the user at this stage. Furthermore, in reality, when agents make decisions, their choices are affected by some unobservable factors [21], such as customers' loyalty on some product brand, which is denoted as $\eta_{j,i}$. This part introduces the uncertainty of users' choice. Now the utility that cloud user j makes in provider i at stage t is defined as follows:

$$u_{j,i}^{t} = v_{j,i}^{t} + \eta_{j,i} \tag{5}$$

We assume that the random variable $\eta_{j,i}$ is an independently, identically distributed extreme value, i.e. it follows Gumbel and type I extreme value distribution [21], and the density of $\eta_{j,i}$ is

$$f(\eta_{j,i}) = e^{-\eta_{j,i}} e^{-e^{-\eta_{j,i}}} \tag{6}$$

and the cumulative distribution is

$$F(\eta_{j,i}) = e^{-e^{-\eta_{j,i}}} \tag{7}$$

The probability of user j choosing provider i at stage t, which is denoted as $P_{j,i}^{t}$

$$\begin{aligned} P_{j,i}^{t} &= Prob(u_{j,i}^{t} > u_{j,i'}^{t}, \forall i' \neq i) \\ &= Prob(v_{j,i}^{t} + \eta_{j,i} > v_{j,i'}^{t} + \eta_{j,i'}, \forall i' \neq i) \\ &= Prob(\eta_{j,i'} < \eta_{j,i} + v_{j,i}^{t} - v_{j,i'}^{t}, \forall i' \neq i) \end{aligned} \tag{8}$$

According to (7), $P_{j,i}^{t}$ is

$$P_{j,i}^{t} = e^{-e^{(\eta_{j,i} + v_{j,i}^{t} - v_{j,i'}^{t})}} \tag{9}$$

Since $\eta_{j,i}$ is independent, the cumulative distribution over all $i \neq i'$ is the product of the individual cumulative distributions

$$P^t_{j,i} \mid \eta_{j,i} = \prod e^{-e^{-(\eta_{j,i}+v^t_{j,i}-v^t_{j,i'})}} \tag{10}$$

And $\eta_{j,i}$ is unknown to the providers, so the choice probability is the integral of $P^t_{j,i} \mid \eta_{j,i}$ over all values of $\eta_{j,i}$ weighted by its density

$$P^t_{j,i} = \int (\prod_{i' \neq i} e^{-e^{-(\eta_{j,i}+p_{i,t}-p_{i',t})}}) e^{-\eta_{j,i}} e^{-e^{-\eta_{j,i}}} d\eta_{j,i} \tag{11}$$

The closed-form expression is

$$P^t_{j,i} = \frac{e^{v^t_{j,i}}}{\sum_{i'} e^{v^t_{j,i'}}} \tag{12}$$

which is the probability of user j choosing to be served by provider i at stage t.

3 Reinforcement Learning Algorithms

After describing the basic settings, we now introduce how to design a pricing policy for the cloud provider. How to set an effective price is a decision-making problem, and reinforcement learning algorithms have been widely used to solve similar issues. Specifically, we adopt Q learning algorithm [18] to determine how the provider sets the price. Note that Q learning algorithm is usually used to solve the sequential decision problem involving only one agent, and therefore when using Q learning algorithm, we let the opponent's action be part of the environment. Moreover, since our problem actually involves two providers competing against each other repeatedly, it can be modeled as a Markov game [16]. In such a game, we use minimax-Q learning algorithm [17] to solve this issue[2]. In the following, we introduce how to design the pricing policy based on Q learning and minimax-Q learning algorithms respectively.

At stage t, provider A sets price according to its own and the opponent B's price at the last stage $t-1$, which is denoted as state $s_{t-1} = (p_{A,t-1}, p_{B,t-1})$. Note that the state does not involve the amount of users participating in each provider since the price has implied users' choices and therefore we only use the prices to represent the state. The state space is denoted as $S = P \times P$. For simplicity, in the following, we use $a \in P$ and $b \in P$ to represent the actions of providers A and B respectively. The pricing policies of provider A and B are denote as Π_A

[2] minimax-Q learning was designed to solve Markov game when its stage game is a zero-sum game. In this paper, although the sum of both providers' payoffs is not zero, the gain of one provider (users choosing this provider) is indeed the loss of the other provider (users not choosing that provider). Therefore, it is actually a zero-sum game, and we use minimax-Q learning algorithm to design the pricing policy in this competing environment.

Algorithm 1. Q learning

```
Input: pricing space $P$; $B$'s pricing policy $\Pi_B$
Output: $A$'s pricing policy $\Pi_A$
1: for $\forall s \in S, V(s) = 0, and\ \forall a \in P, Q(s,a) = 0$
2: for $\forall s \in S, \forall a \in P, \Pi_A(s,a) = 1/|P|$
3: repeat
4:     at the current state $s$, given $\epsilon$, generate a random number $rand$ $(0 < rand < 1)$;
       when $rand \leq \epsilon$, $A$ chooses a price $a \in P$ randomly; when $rand > \epsilon$, $A$ chooses a
       price $a \in P$ according to the pricing policy $\Pi_A$
5:     $B$ chooses the price $b$ (according to the pricing policy $\Pi_B$), the next state is
       $s' = (a,b)$
6:     $Q(s,a) = (1-\alpha) * Q(s,a) + \alpha * (r_{i,t} + \gamma * V(s'))$
7:     $\Pi_A(s,\cdot) = argmax_{\Pi'_A(s,\cdot)}(\sum_{a'}(\Pi_A(s,a') * Q(s,a')))$
8:     $V(s) = \sum_a(\Pi_A(s,a) * Q(s,a))$
9: until ($\Pi_A(s,\cdot)$ is converged)
```

Algorithm 2. minimax-Q learning

```
Input: pricing space $P$; $B$'s pricing policy $\Pi_B$
Output: $A$'s pricing policy $\Pi_A$
1: for $\forall s \in S, V(s) = 0, and\ \forall a \in P, Q(s,a,b) = 0$
2: for $\forall s \in S, \forall a \in P, \Pi_A(s,a) \leftarrow 1/|P|$
3: repeat
4:     at the current state $s$, given $\epsilon$, generate a random number $rand$ $(0 < rand < 1)$;
       when $rand \leq \epsilon$, $A$ chooses a price $a \in P$ randomly; when $rand > \epsilon$, $A$ chooses a
       price $a \in P$ according to the pricing policy $\Pi_A$
5:     $B$ chooses the price $b$ (according to the pricing policy $\Pi_B$), the next state is
       $s' = (a,b)$
6:     $Q(s,a,b) = (1-\alpha) * Q(s,a,b) + \alpha * (r_{i,t} + \gamma * V(s'))$
7:     $\Pi_A(s,\cdot) = argmax_{\Pi'_A(s,\cdot)}(min_{b'} \sum_{a'}(\Pi_A(s,a') * Q(s,a',b')))$
8:     $V(s) = min_{b'}(\sum_{a'}(\Pi_A(s',a') * Q(s',a',b')))$
9: until ($\Pi_A(s,\cdot)$ is converged)
```

and Π_B respectively. Based on these notations, Q learning algorithm is shown in Algorithm 1, and minimax-Q learning algorithm is shown in Algorithm 2. In this setting, it is guaranteed that both algorithms will converge [17,18]. The final output Π_A is the designed pricing policy.

4 Experimental Analysis

In this section, we run numerical simulations to analyze the reinforcement learning based pricing policies in different situations. We first describe the parameter setup in the experiments in the following.

Table 1. Experimental parameters

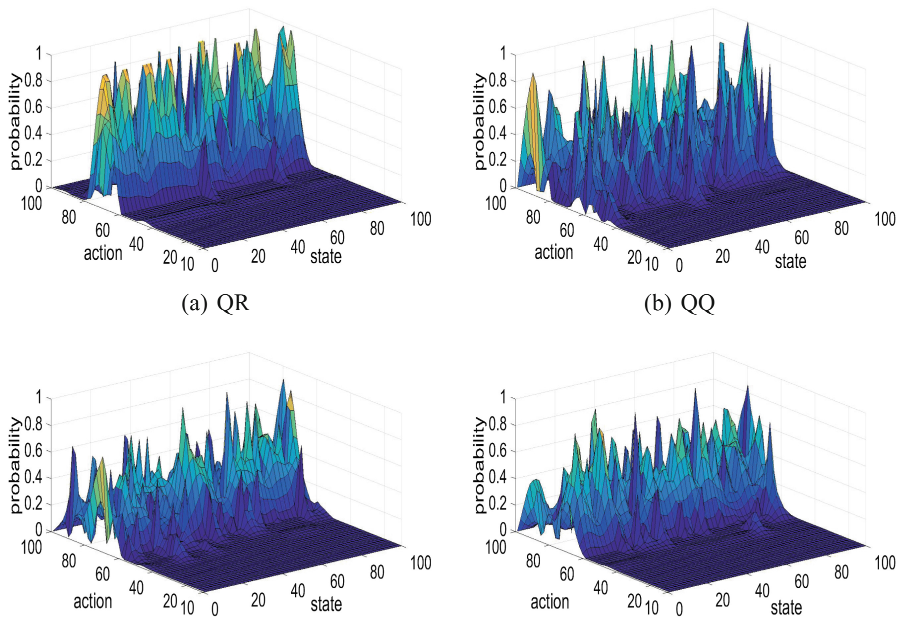
Parameter setup	Description
$c_{\cdot,0} = 5$	cloud provider's marginal cost at the initial stage
$P = \{10, 20, \ldots, 100\}$	the set of allowable prices
$N = 100$	the amount of users in the market
$\delta \in [50, 150]$	the cloud user's marginal value δ follows a uniform distribution supported on [50,150]
$\beta = 0.01$	parameter β in Eq. 1
$\theta = 0.001$	parameter θ in Eq. 1
$\xi = 0.8$	parameter ξ in Eq. 4
$\epsilon = 0.2$	exploration rate in Q learning and minimax-Q learning algorithms
$\gamma = 0.8$	discount factor in Q learning and minimax-Q learning algorithms

4.1 Experimental Parameters

First, we assume that each cloud provider has the same initial marginal cost, i.e. $c_{\cdot,0} = 5$, and the marginal cost is decreased as the demand increases. In addition, we assume that the set of allowable prices P chosen by cloud providers is $\{10, 20, \ldots, 100\}$. Furthermore, we assume that there are $N = 100$ cloud users in total. The marginal values of users δ are independent random variable, and for illustrative purpose, we assume that they are drawn from a uniform distribution with support $[50, 150]$. Other parameters used in the following simulations follow the typical setting in the related literature, and are shown in Table 1.

4.2 Pricing Policy

We first describe the pricing policies trained and output by minimax-Q and Q learning algorithms respectively. We consider the case that the cloud provider takes Q learning and minimax-Q learning algorithms against the opponent choosing actions randomly, and the case that both cloud providers are trained in Q learning and minimax-Q learning algorithms. We name the trained pricing policies as QR, QQ, MR and MM respectively, and for example QQ means that both cloud providers adopt Q learning algorithm and are trained against each other. We show these four pricing policies in Fig. 1. From these figures, we can find the probability of the provider choosing each price (action) at each state. Note that in this paper, the state is a tuple including two providers' prices at the last stage, and in order to show the state in one-dimension *state*-axis, we map state (10, 10) to 1, map state (10, 20) to 2, ..., map (20, 10) to 11, ..., and map (100, 100) to 100. Furthermore, we find that no provider intends to



(a) QR (b) QQ

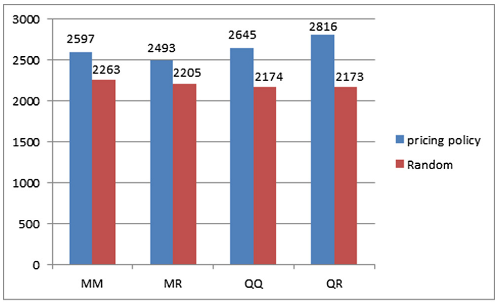
(c) MR (d) MM

Fig. 1. Pricing policies trained in Q and minimax-Q learning algorithms

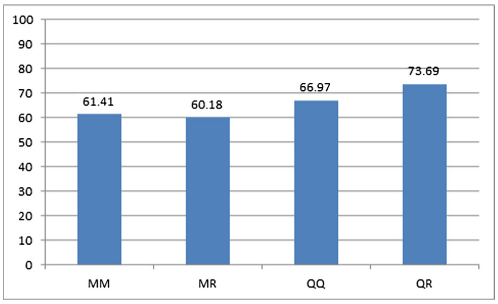
set a minimal price 10 to attract users, since on one hand, a low price is not beneficial for the long-term profit, and on the other hand, cloud users' choices of providers are also affected by some unobservable factors, and thus a provider with a minimal price cannot attract all users, but lose some profits. Furthermore, we find that these pricing policies will not set the highest price since such a high price will drive all users to leave. Moreover, we find that the surface of QQ and QR is sharper than MR and MM's. Specifically, at some state, QR and QQ will choose a deterministic action, but MR and MM have mixed actions. This is because in contrast to Q learning trying to maximizing the profit regardless of the opponent's action, minimax-Q learning needs to randomize the action in order to maximize the profit in the worst case.

4.3 Evaluation

In this section, we run simulations to evaluate the above pricing policies in different situations. Specifically, we use the average profit and the winning percentage as the evaluation metrics.



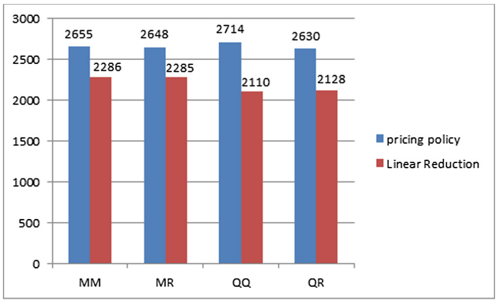
(a) Average profit



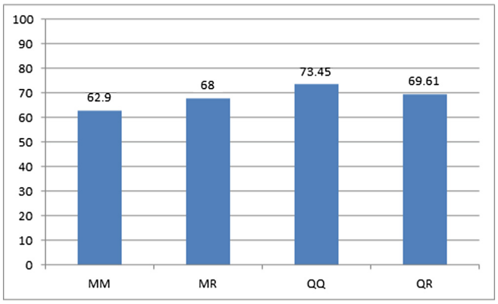
(b) Winning percentage

Fig. 2. MM, MR, QQ, QR vs. *Random*

vs. Random Pricing Policy. We first evaluate these four pricing policies against the opponent adopting a random pricing policy, which chooses each price with equal probability. The reason for doing this is that when participating in the cloud market, some fresh cloud providers often explore the competing environment randomly in order to collect more market information. The competing results are shown in Fig. 2, where Fig. 2(a) is the average profit over 10000 stages, and Fig. 2(b) is the winning percentage of these four policies competing against the random pricing policy. We find that all these four pricing policy can beat the opponent. Not surprisingly, we find that QR is the best one among these four policies when competing against the random pricing policy since QR is trained specifically against the random policy. In contrast, we find that MR is the worst one. This is because when the opponent chooses the price randomly, i.e. not trying to make the other side be the worst, minimax-Q cannot perform well. In fact, we find that QQ and QR perform better than MR and MM. This may indicate that the agent should adopt Q learning when its opponent cannot take action in an intelligent way.



(a) Average profit

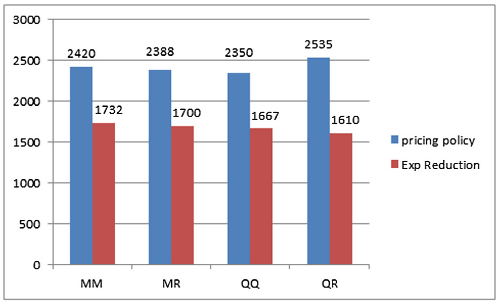


(b) Winning percentage

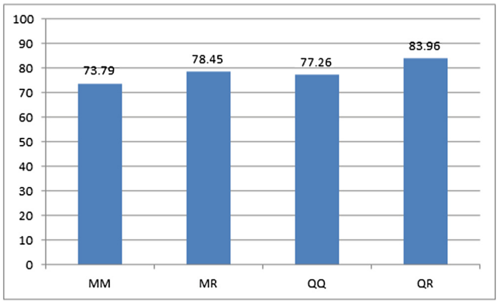
Fig. 3. MM, MR, QQ, QR vs. *Linear Reduction*

vs. Price Reduction Policy. In the real world, cloud providers usually attract cloud users by decreasing the price continuously. For example, when a fresh cloud provider enters the market, it may keep decreasing the price to attract users. We also evaluate these four pricing policies against the cloud provider which keeps reducing the price. Specifically, we consider two typical price reduction policies, Linear Reduction and Exp Reduction. In Linear Reduction policy, the price decreases linearly with respect to time, where at stage t the price is $p_t = p_0 - 0.01t$ (p_0 is the initial price, and we set it as the maximal price, i.e. 100), while in Exp Reduction, the price decreases exponentially with time, where $p_t = p_0 * e^{-0.0003t}$ ($p_0 = 100$ which is the same as before). The results of QQ, QR, MM, MR competing against Linear Reduction policy and Exp Reduction policy are shown in Figs. 3 and 4. We still find that our reinforcement learning-based pricing policies can beat these price reduction policies. Again, we find that MM and MR cannot outperform the reduction policies significantly than that QQ and QR do.

***Q-X* vs. *X*.** In the above, it seems that when competing against the opponent using simple pricing policies (i.e. random or price reduction), Q learning based pricing policy is better. However, after investigating the fundamentals of minimax-Q and Q learning algorithms, we can see that minimax-Q is more suitable in this Markov game with two competing providers. Since this is not proved in the above experiments, in the following we further investigate this issue by



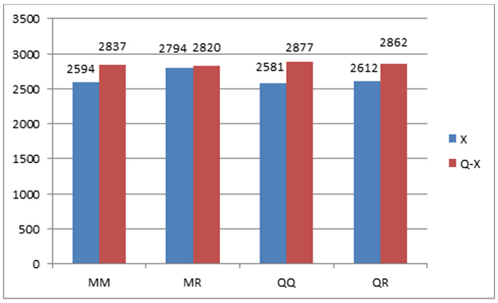
(a) Average profit



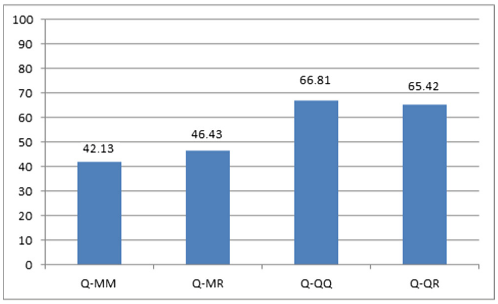
(b) Winning percentage

Fig. 4. MM, MR, QQ, QR vs. *Exp Reduction*

using Q learning algorithm to train the pricing policy against the above four policies, i.e. QQ, QR, MR, MM. We then obtained four new pricing policies, Q-QR, Q-QQ, Q-MR, and Q-MM. Q-QR is a pricing policy based on Q learning competing against QR pricing policy in the above section. We run the following simulations including Q-MM vs MM, Q-MR vs MR, Q-QQ vs QQ and Q-QR vs QR. The results are shown in Fig. 5, where QQ, QR, MR, MM are denoted as X. From Fig. 5(b), in terms of winning percentage, we find that Q-QQ outperforms QQ and Q-QR outperforms QR, i.e. the winning percentage is more than 50%. However, even though Q-MM is trained against MM, it is outperformed by MM, i.e. the winning percentage is less than 50%. The similar result happens for Q-MR. However, from Fig. 5(a), we find that even though Q-MM is outperformed by MM in terms of winning percentage, it obtains more profits than MM. It is similar for Q-MR. This is because minimax-Q based pricing policies try to do the best in the worst case, and therefore its winning percentage can be kept at a good level. However, Q learning based policies try to maximize the profits at all times, and therefore perform better in terms of making profits.



(a) Average profit



(b) Winning percentage

Fig. 5. Q-X vs. X

5 Conclusions

How to set prices effectively is an important issue for the cloud provider, especially in the environment with multiple cloud providers competing against each other. In this paper, we use reinforcement learning algorithms to address this issue. Specifically, we model the issue as a Markov game, and use minimax-Q and Q learning algorithms to design the pricing policies respectively. We then run extensive experiments to analyze the pricing policies. We find that although minimax-Q is more suitable in analyzing the competing game with multiple self-interested agents, Q learning based pricing policy performs better in terms of making profits. We also find that minimax-Q based pricing policy is better for remaining attractive to cloud users. The experimental results can provide useful insights on designing practical pricing policies in different situations.

Acknowledgment. This paper was funded by the National Natural Science Foundation of China (No. 61402344), the Excellent Dissertation Cultivation Funds of Wuhan University of Technology (No. 2017-YS-065) and the Fundamental Research Funds for the Central Universities (WUT:2017II14XZ).

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing: clearing the clouds away from the true potential and obstacles posed by this computing capability. Commun. ACM **53**(4), 50–58 (2010)
2. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandicl, I.: Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **25**(6), 599–616 (2009)
3. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop, pp. 1–10 (2008)
4. Buyya, R., Yeo, C., Venugopal, S.: Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities. In: The 10th IEEE International Conference on High Performance Computing and Communications, pp. 5–13 (2008)
5. Laatikainen, G., Ojala, A., Mazhelis, O.: Cloud services pricing models. In: Herzwurm, G., Margaria, T. (eds.) ICSOB 2013. LNBIP, vol. 150, pp. 117–129. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39336-5_12
6. Sharma, B., Thulasiram, R.K., Thulasiraman, P., Garg, S.K., Buyya, R.: Pricing cloud compute commodities: a novel financial economic model. In: The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 451–457 (2012)
7. Wang, H., Jing, Q., Chen, R., He, B., Qian, Z., Zhou, L.: Distributed systems meet economics: pricing in the cloud. In: The 2nd USENIX Conference on Hot Topics in Cloud Computing, pp. 1–6 (2010)
8. Feng, Y., Li, B., Li, B.: Price competition in an oligopoly market with multiple iaas cloud providers. IEEE Trans. Comput. **63**(1), 59–73 (2014)
9. Kantere, V., Dash, D., Francois, G., Kyriakopoulou, S., Ailamaki, A.: Optimal service pricing for a cloud cache. IEEE Trans. Knowl. Data Eng. **23**(9), 1345–1358 (2011)
10. Xu, H., Li, B.: Maximizing revenue with dynamic cloud pricing: the infinite horizon case. In: IEEE International Conference on Communications, pp. 2929–2933 (2012)
11. Vengerov, D.: A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments. Future Gener. Comput. Syst. **24**(7), 687–693 (2008)
12. Xu, H., Li, B.: Dynamic cloud pricing for revenue maximization. IEEE Trans. Cloud Comput. **1**(2), 158–171 (2013)
13. Xu, B., Qin, T., Qiu, G., Liu, T.Y.: Optimal pricing for the competitive and evolutionary cloud market. In: The 24th International Joint Conference on Artificial Intelligence, pp. 139–145 (2015)
14. Truong-Huu, T., Tham, C.K.: A game-theoretic model for dynamic pricing and competition among cloud providers. In: The 6th International Conference on Utility and Cloud Computing, pp. 235–238 (2013)
15. Truong-Huu, T., Tham, C.K.: A novel model for competition and cooperation among cloud providers. IEEE Trans. Cloud Comput. **2**(3), 251–265 (2014)
16. Wal, J.: Stochastic dynamic programming. Methematisch Centrum (1980)
17. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: 11th International Conference on Machine Learning, pp. 157–163 (1994)
18. Watkins, C.J.C.H., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)

19. Adams, I.F., Long, D.D.E., Miller, E.L., Pasupathy, S., Storer, M.W.: Maximizing efficiency by trading storage for computation. In: The 1st USENIX Conference on Hot Topics in Cloud Computing. vol. 7 (2009)
20. Jung, H., Klein, C.M.: Optimal inventory policies under decreasing cost functions via geometric programming. Eur. J. Oper. Res. **132**(3), 628–642 (2001)
21. Train, K.E.: Discrete Choice Methods with Simulation. Cambridge University Press, Cambridge (2003)