

# Dynamic Pricing Agents and Multiagent Learning

Junling Hu

*University of Rochester*

January 15, 2003

**Abstract.** We implemented three different types of pricing agents in a simulated economy. Each type of agent is based on a different learning method. The first method is simple reinforcement learning. The second method is the traditional Q-learning method. The third method is Nash Q-learning method. In each simulation, there are two agents, and a fixed amount of customers. The agent that charges a lower price will attract all the customers. When both firms charge the same price, they receive equal share of the market. Our simulation shows that the two Q-learning methods that take future rewards into account perform better than the simple method that is myopic. Among the two Q-learning methods, Nash Q-learning performs consistently better than the Q-learning method. This suggests the importance of game theoretical modeling in online learning.

**Keywords:** Price agents, multiagent systems

## 1. Introduction

The presence of software agents on the Internet has dramatically changed the way stores compete with each other. With search engine and comparison shopping bots gathering prices from different stores instantly, any price difference among stores will be instantly detected by the customer. Even though there exists some variance among store's prices, such variance has been shrinking. Is there a learning method that allows the seller to consistently perform better than other methods? Another question we are concerned is: Would the economy with pricing agents lead to more or less volatile markets? What kind of price equilibrium would the market converge to under a pricing strategy?

In this paper, we investigate how a software agent can help online stores monitor other stores' prices continuously, and then set corresponding prices. Such agents are called pricebots (Kephart, Hanson, & Greenwald, 2000). An intelligent agent should be able to charge a price reasonable high for the firm, but not too high to lose the customers. Such an agent will keep monitoring other firms' prices, learning about other firms' pricing behavior, and dynamically setting the firm's prices. Through learning about other agents, our agent will help the firm to engage in profitable pricing practices, such as implicit collusion on high prices, instead of in price wars.

In online market, learning is an important part of dynamic pricing strategy. As the environment constantly changes, it is impossible to foresee all possible



situations. With learning, we can put the new data into a new perspective, and change our behavior to adapt to the new environment.

We provide a learning method that is based on a solid game theoretical framework. Unlike other learning methods in the literature that is based on single-agent systems, our learning method explicitly takes other agents into account. This learning method has been proved to converge

We implemented three different types of pricing agents in a simulated economy. Each type of agent is based on a different learning method. The first method is simple reinforcement learning, which learns about an optimal action for one period based on the rewards it receives for that action. The second type is Q-learning method that learns about Q-values which represent long-term optimal values for the agent's own actions. The third type is Nash Q-learning method that learns about Q-values which represents long-term Nash equilibrium values for agent's joint actions.

We let these agents compete with each other in our simulation. In each simulation, there are two agents, and a fixed amount of customers. The agent that charges a lower price will attract all the customers. When both firms charge the same price, they receive equal share of the market.

Our simulation shows that the two Q-learning methods that take future rewards into account perform better than the simple method that is myopic. Among the two Q-learning methods, Nash Q-learning performs consistently better than the Q-learning method. This suggests the importance of game theoretical modeling in online learning.

## 2. Related Work

### 2.1. THE GAME OF PRICE COMPETITION

In economics literature, the game of price competition was first studied by Bertrand, and is now commonly called Bertrand game (Fudenberg & Tirole, 1991). In such a game, stores produce identical products. Price is the only instrument that a store uses to compete with the others. Stores announce their prices simultaneously and noncooperatively. Two stores are noncooperative if each store pursues its own self-interest and has no way to communicate or collude with others. It is also assumed that all stores have the same constant marginal cost, and that the marginal cost is known to every store. Stores can observe other stores' prices once they are announced. Consumers can observe all prices with no costs. Because the products from all stores are perfect substitutes, consumers only buy from the stores with the lowest price. If all stores charge the same prices, then each store takes an equal share of the market.

The unique Nash equilibrium of the one-period game is that both stores charge the same prices equal to the constant marginal cost (Fudenberg &

Tirole, 1991; Tirole, 1988). This is often called “Bertrand paradox”: Although this is a duopolic market, the stores behave like in a perfect competitive market by setting the price equal to the marginal cost and making no profits.

The one period game is an over-simplified model of agent interaction. A more sophisticated model should include multiple periods. That first leads to the repeated game model.

In a finite-period repeated game, a reasonable Nash equilibrium strategy is one such that a store’s pricing strategy maximizes its expected discounted profits given the other store’s strategy at any date and history. By backward induction, the last date of the game is just a one-period game, the equilibrium strategy is that both stores charge the same price equal to the marginal cost. The game in the last but the second date is played just like the last period. Therefore, the equilibrium strategy of the finite multiple period game is to set prices equal to marginal cost every period.

The Internet price competition game is more like an infinitely repeated game. The game is played in much short intervals in e-commerce, we can reasonably assume that every period, the stores act as if it will survive at least several periods. Second, because of the high frequency of play, the number of periods is large even if the calendar period is short. The equilibrium becomes much more complicated in infinite periods. Although competitive pricing can be an equilibrium strategy, there are many more other types of equilibrium strategy, including tacit collusion. In particular, the Folk Theorem for infinitely repeated game says that if players are patient enough, then any positive payoffs that are smaller than monopoly profits can be a subgame perfect equilibrium. (Fudenberg & Tirole, 1991) The main intuition behind the Folk Theorem is that any equilibrium strategy can be supported by a severe penalty threat: if one deviates from the equilibrium strategy, the other will charge competitive price. When discount factor is close to one, the penalty is close to infinity which overweighs any short-term finite gains from the deviation.

Unfortunately the equilibrium theory fails to tell us, if an actual game is played, which equilibrium the game will achieve, and how the players actually play the strategies.

## 2.2. THE DESIGN OF PRICEBOT

Sridharan and Tesauro (Sridharan & Tesauro, 2000) has tested offline performance of single-agent Q-learning in price competition.

## 2.3. THE STOCHASTIC GAME MODEL

We study price competition in the model of stochastic games. Stochastic games include repeated games as special cases. In stochastic games, there is a state variable that indicates the changing environment. Under different states,

agents have different action spaces or encounter different reward functions. Repeated games are stochastic games that has the same state over time.

In stochastic games, the state transition is assumed to follows a Markov process. A formal definition of stochastic games is given as follows (Thusijsman, 1992):

**DEFINITION 1.** *An  $n$ -player stochastic game  $\Gamma$  is a tuple  $\langle S, A^1, \dots, A^n, r^1, \dots, r^n, p \rangle$ , where  $S$  is the state space,  $A^k$  is the action space of player  $k$  for  $k = 1, \dots, n$ ,  $r^k : S \times A^1 \times \dots \times A^n \rightarrow R$  is the payoff function for player  $k$ ,  $p : S \times A^1 \times \dots \times A^n \rightarrow \Delta$  is the transition probability map, where  $\Delta$  is the set of probability distributions over state space  $S$ .*

In a stochastic games, agents choose actions simultaneously at each time  $t$ . The time  $t$  is assumed to be discrete:  $t = 0, 1, 2, \dots$ . The state at time  $t$  is denoted by  $s_t$ . In state  $s_t = s$ , agents independently choose actions  $a^1 \in A^1, a^2 \in A^2, \dots, a^n \in A^n$ , and receive rewards  $r^k(s, a^1, \dots, a^n)$  for  $k = 1, \dots, n$ . The state then transits to the next state  $s_{t+1} = s'$  based on the fixed transition probability  $p(s'|s, a^1, \dots, a^n)$ , which satisfies the constraint

$$\sum_{s' \in S} p(s'|s, a^1, \dots, a^n) = 1. \quad (1)$$

In a *discounted stochastic game*, the objective of each player is to maximize a discounted sum of rewards, where the discount factor  $\beta \in [0, 1)$ . An agent's *strategy* is a plan for playing a game. A strategy  $\pi = (\pi_0, \dots, \pi_t, \dots)$  is defined over the whole course of the game, where  $\pi_t$  is called the *decision rule* at time  $t$ . A decision rule is a function  $\pi_t : \mathbf{H}_t \rightarrow \sigma(A^k)$ , where  $H_t = (s_0, a_0^1, \dots, a_0^n, \dots, s_{t-1}, a_{t-1}^1, \dots, a_{t-1}^n, s_t)$  is the history at time  $t$ , and  $\sigma(A^k)$  is probability distribution over agent  $k$ 's action space.  $\pi$  is called a *stationary strategy* if  $\pi_t = \bar{\pi}$  for all  $t$ , that is the decision rule is independent of time.  $\pi$  is called a *behavior strategy* if  $\pi_t = f(H_t)$ . In a stationary strategy, the decision rule  $\bar{\pi} = (\bar{\pi}(s^1), \dots, \bar{\pi}(s^m))$  assigns a probability distribution over available actions for each state  $s^j, j = 1, \dots, m$ , where  $m$  is the number of states.  $\bar{\pi}(s^j)$  is called a *pure strategy* if it assigns probability 1 to one of the actions.

Let  $\pi^k$  be the strategy of player  $k$ . For a given initial state  $s$ , player  $k$  receives the following value from the game:

$$v^k(s, \pi^1, \pi^2, \dots, \pi^n) = \sum_{t=0}^{\infty} \beta^t E(r_t^k | \pi^1, \pi^2, \dots, \pi^n, s_0 = s) \quad (2)$$

### 3. Simulation of Internet Price Competition

An economic market can be modeled as a stochastic game. To make the story concrete, let us consider a market with  $n$  sellers. Each of the sellers has the capacity to satisfy all market demand. At each time period, the sellers post their prices, and then observe the customer demand. Each seller's profit is calculated as the following:

The seller with the lowest price gets all the demand.

The action in this game is therefore the price a seller chooses.

The market demand curve is a fixed amount of

Our agent is designed to work for a store on the Internet. We assume stores sell books or other standard goods such as CD or electronics. This assumption is to eliminate the case of product differentiation. Assume price is the only thing that matters to the consumers due to the following reasons:

1. All the users use comparison shopping
2. Products are non-differentiable.
3. There is not much difference between my store and other stores in terms of brand-name, customer service, delivery, and so on.

Each store  $i$  makes decision on its price  $p^i$ . We assume all the stores have the same marginal cost  $c$ , and the cost is a constant. The reward function of store  $i$  is its profit function

$$R^i = p^i \cdot q^i - cq^i \quad (3)$$

where  $R^i$  is agent  $i$ 's profit, and  $q^i$  is the demand quantity for store  $i$ . The market demand quantity is determined by the relative prices among stores. Therefore  $q^i$  is a function of  $p^1, \dots, p^n$ .

$$R^i = (p^i - c)q^i(p^1, \dots, p^n) \quad (4)$$

We assume the market is near perfectly competitive: There are many sellers. The one with the lowest price will and one attract all the customers. The total demand is a fixed  $D$ . The game starts from an initial state where all agent's prices are equal:  $p_0^1 = \dots = p_0^n$ . The demand is uniformly distributed among sellers. Thus each seller gets demand equal to  $\frac{D}{n}$ . Starting from time 1, if any agent has a price lower than all the others,

$$p_t^i = \min(p_t^1, \dots, p_t^n) \text{ and } p_t^i \neq p_t^k, \text{ for all } i \neq k,$$

then that agent gets all the demand,  $q_t^i = D$  and  $q_t^k = 0, \forall i \neq k$ .

Assume agents are identical, in terms of cost  $c$  and their power in the market. The game is symmetric in terms of each agent. This is a simplification,

but it would not affect the solution. In fact, we can handle non-symmetric case as long as all the information is perfectly available.

We model the interaction as a stochastic game. Our agent's goal is to maximize the total discounted profits.

$$\max \sum_{t=0}^T \beta^t R_t^i = \sum_{t=0}^T \beta^t p_t^i q_t^i(p_t^1, \dots, p_t^n)$$

The solution to this game is a Nash equilibrium  $(\pi^1, \pi^2, \dots, \pi^n)$ , where each agent's strategy  $\pi^i$  is defined over T periods. It is difficult to find an analytical solution. We will search for computational solutions in this paper.

#### 4. Three types of learning agents

We investigate three types of learning agents in this paper. The first two types have been implemented in E-commerce environment by other researchers (Tesauro & Kephart, 1999) under slightly different context. The third type is a new learning method designed recently by one of us, and is implemented in E-commerce environment for the first time.

##### 4.1. SIMPLE REINFORCEMENT LEARNING AGENT

The first type of agent is a simple reinforcement learning agent that acts as a learning automata (Narendra & Thathachar, 1989). Such an agent does not have long-term memory. It changes its action solely based on the reward it receives from last period. If the reward is positive, then it increase the frequency of choosing previous action. If the reward is negative, it decrease the frequency of choosing that action. Over time, after repeated learning, the agent will be able to converge to a correct action under a stationary environment. By stationary, we refer to an environment that gives the same reward when an agent chooses the same action.

An environment that consists of multiple agent is certainly not stationary. The other agents many change their action strategies and thus lead to different payoffs to our agent even if it chooses the same action. The simple reinforcement learning agent will not be able to learn the right action. We implement such an agent as a benchmark to compare with other learning methods.

##### 4.2. Q-LEARNING AGENT

A Q-learning agent strives to find the optimal policy  $\pi$  that solves a multi-period decision problem. The agent is trying to maximize the total discounted reward

$$v(s, \pi) = \sum_{t=0}^{\infty} \beta^t E(r_t | \pi, s_0 = s). \quad (5)$$

In a single-agent Markov decision process, there exists an optimal policy  $\pi^*$  such that for any  $s \in S$ , the following equation holds:

$$v(s, \pi^*) = \max_a \left\{ r(s, a) + \beta \sum_{s'} p(s'|s, a) v(s', \pi^*) \right\}, \quad (6)$$

where  $v(s, \pi^*)$  is called the *optimal value* for state  $s$ . Equation (6) is also called *Bellman equation*.

The basic idea of Q-learning is that we can define the part of the right-hand side of (6) inside the max operator as

$$Q^*(s, a) = r(s, a) + \beta \sum_{s'} p(s'|s, a) v(s', \pi^*) \quad (7)$$

By this definition,  $Q^*(s, a)$  is the total discounted reward attained by taking action  $a$  in state  $s$  and then following the optimal policy thereafter.

If we know  $Q^*(s, a)$ , then the optimal policy  $\pi^*$  can be found, which is always taking an action that maximizes  $Q^*(s, a)$  under the state  $s$ . The problem is then reduced to finding the function  $Q^*(s, a)$  instead of searching for the optimal function  $v(s, \pi^*)$ .

In Q-learning, the agent starts with arbitrary initial values of  $Q(s, a)$  for all  $s \in S, a \in A$ . At each time  $t$ , the agent chooses an action and observes its reward,  $r_t$ . The agent then updates its Q-values as follows:

$$Q_{t+1}(s, a) = (1 - \alpha_t) Q_t(s, a) + \alpha_t [r_t + \beta \max_b Q_t(s_{t+1}, b)]. \quad (8)$$

where  $\alpha_t \in [0, 1)$  is the learning rate, which decreases over time. Watkins and Dayan (Watkins & Dayan, 1992) proved that sequence (8) converges to  $Q^*(s, a)$  under the assumption that all states and actions have been visited infinitely often.

Q-learning is a model-free learning method, in which we directly learn about the Q-values without learning the state transition probabilities or reward functions. Thus this method is very easy to implement and become useful when we have little knowledge of the environment. However, Q-learning works well in a single-agent environment, given that it satisfies the requirement of stationarity and Markov transition. It does not guarantee to work well in a multiagent environment, which is non-stationary and violates many assumptions of the Q-learning model.

There are many attempts of applying Q-learning to multiagent environments. For E-commerce domain, Tesauro and Kephart (Tesauro & Kephart, 1999) have implemented Q-learning agents that learn about pricing strategies.

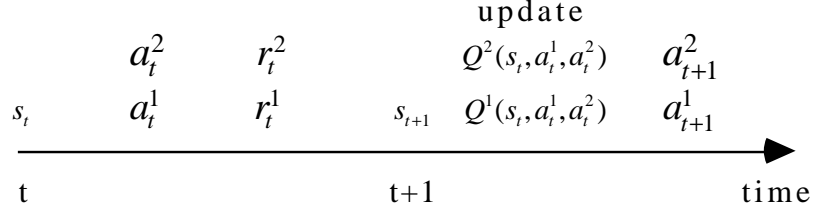


Figure 1. Time line of actions

### 4.3. NASH Q-LEARNING AGENTS

The Nash Q-learning method, constructed by Hu and Wellman (Hu & Wellman, 1998; Hu & Wellman, 2000), extends the framework of Q-learning from single-agent Markov Decision Process to stochastic games. The agent's objective is to maximize the total discounted rewards defined in (??) which is defined over all the agents' Nash equilibrium strategies (policies).

As in single-agent Q-learning, the Nash Q-learning agent updates its Q tables after it observes the state, actions taken by all the agents, and the rewards received by all the agents. It differs from traditional Q-learning in two ways: First, the Q-values in Nash Q-learning depends on the joint actions of all agents, while in Q-learning they depends only on the agent's own action. Second, the Q-values in Nash Q-learning are updated with future Nash equilibrium payoff, while in Q-learning they are updated with the maximum payoff of the agent's own action choices. The Nash updating is justified by our belief that agents always try to model each other, and all agents are rational. Nash equilibrium represents our belief on how the game is going to be played in the long run.

A Q-function can be decomposed into sub-tables. Let a Q-function  $Q^k = (Q^k(s^1), \dots, Q^k(s^m))$  be agent  $k$ 's own Q-function.  $Q^k(s^j)$  is the Q-function under state  $s^j$ , with each element represented by  $Q^k(s^j, a^1, \dots, a^n)$ . The total number of entries in  $Q^k(s^j)$  is  $\prod_{i=1}^n |A^i|$ . Agent  $k$  updates its Q-values according to the following rule:

$$Q_{t+1}^k(s, a^1, \dots, a^n) = (1 - \alpha_t) Q_t^k(s, a^1, \dots, a^n) + \alpha_t [r_t^k + \beta Nash Q_t^k(s_{t+1})] \quad (9)$$

where  $\alpha_t = 0$  for  $(s, a^1, \dots, a^n) \neq (s_t, a_t^1, \dots, a_t^n)$ , and  $Nash Q_t^k(s_{t+1}) = \pi^1(s_{t+1}) \cdots \pi^n(s_{t+1}) \cdot Q_t^k(s_{t+1})$ . The joint strategy  $(\pi^1(s_{t+1}), \dots, \pi^n(s_{t+1}))$  is a Nash equilibrium for the normal-form game  $(Q_t^1(s_{t+1}), \dots, Q_t^n(s_{t+1}))$ .

Note that  $\pi^1(s_{t+1}) \cdots \pi^n(s_{t+1}) \cdot Q_t^k(s_{t+1})$  is a scalar, which is agent  $k$ 's expected Nash equilibrium payoff in state  $s_{t+1}$ . A second thing to be noted is that (9) does not update all the entries in the Q-functions. It only updates the entry corresponding to current state and the actions chosen by the agents. This kind of updating is called asynchronous updating.

In order to calculate the Nash equilibriums strategies  $\pi^1(s_{t+1})$  through  $\pi^n(s_{t+1})$ , agent  $k$  needs to know all  $Q_t^1(s_{t+1}), \dots, Q_t^n(s_{t+1})$ . The information about other agents' Q-functions are not given, and agent  $k$  has to learn about them in the game. Since agent  $k$  has no information about other agents' Q-tables at the beginning except the possible actions and states, it forms conjectures about those Q-tables. For example, the agent can assign  $Q_0^j(s, a^1, \dots, a^n) = 0$  for all  $j$  and all  $s, a^1$  and  $a^2$ . As the game proceeds, agent  $k$  observes other agents' immediate rewards and previous actions. That information can then be used to update agent  $k$ 's conjectures on other agents' Q-functions. Agent  $k$  updates its belief on agent  $j$ 's Q-function, for all  $j \neq k$ , according to the same updating rule as in (9).

## 5. Implementation of learning agents

In our setup, we require that each seller has three action choices: charge a price the same as last period's market price, charge a price a little higher than the last period's price, or a price a little lower than last period. This requirement is a simplification of changing a price in any range relative to last period's price. In fact, our requirement only makes convergence a little slower, but it does not change the qualities results.

We have implemented agents with three different pricing strategies, and each agent is assigned a type associated with each strategy. All of different types of agents charge a price based previous period's market price. The only difference is how to use that information .

### 5.1. SIMPLE REINFORCEMENT LEARNING

The simple learning agent treats the game as a repeated game. It does not model the changing situation. In other words, there is no state in its model. It chooses its price based on the following rules:

$$P_t = \begin{cases} P_{t-1}^* - 0.1 & \text{if } P_{t-1} > P_{t-1}^*; \\ \text{Random}(P_{t-1}^* - 0.1, P_{t-1}^*, P_{t-1}^* + 0.1) & \text{if } P_{t-1} = P_{t-1}^*; \\ P_{t-1}^* + 0.1 & \text{if } P_{t-1} < P_{t-1}^*; \end{cases}$$

### 5.2. Q-LEARNING AGENT

A Q-learning agent is a sophisticated reinforcement learning that directly learns about value function.

$$Q_{t+1}(s, P) = (1 - \alpha_t)Q_t(s, P) + \alpha_t[r_t + \beta \max_b Q_t(s_{t+1}, b)]$$

where state is defined as the minimum price from last periods,  $s_t = P_{t-1}^*$ , where  $P_{t-1}^* = \min\{P_{t-1}^1, \dots, P_{t-1}^1\}$ .

### 5.3. NASH Q-LEARNING AGENT

A Q-learning agent is a sophisticated reinforcement learning that directly learns about value function.

In a 2-player game, agent 1 learns about its Q-function through the following updating process:

$$Q_{t+1}^1(s, P^1, P^2) = (1 - \alpha_t)Q_t^1(s, P^1, P^2) + \alpha_t[r_t + \beta Nash Q_t^1(s_{t+1})].$$

where  $Nash Q_t^1(s_{t+1}) = \pi^1(s_{t+1})Q_t^1(s_{t+1})\pi^2(s_{t+1})$ , and  $(\pi^1(s_{t+1}), \pi^2(s_{t+1}))$  is the Nash equilibrium solution for the normal-form game  $(Q_t^1(s_{t+1}), Q_t^2(s_{t+1}))$ .

## 6. The Simulation

### 6.1. BASIC SETUP

Consider a game that is played over many periods. Each period is one day. Our agent watches for the price change during the day, and make new price choice for the next day. The discount rate  $\beta$  is the inverse of the daily interest rate,  $\beta = \frac{1}{1+i}$ , where  $i = \frac{5\%}{365}$ , and  $\beta = 0.99986$ .

The market demand is modeled as a flat curve. When the price is above 50, there is no demand for the product. Here we can consider 50 as the consumer's reservation price.

In our simulation, the system starts from the case where every agent charges price 20. Starting from time 1, the agents choose their prices. Each agent's marginal cost is the same  $c = 5$ . Price has to be in the range of  $[5, 50]$ . The total demand  $D = 1000$ .

There are two firms in the system, and their profit functions are symmetric, defined as:

$$r^1 = (p^1 - c)q^1(p^1, p^2) \quad (10)$$

$$r^2 = (p^2 - c)q^2(p^1, p^2) \quad (11)$$

If  $p^1 > p^2 = p$  then

$$r^1 = 0$$

$$r^2 = (p - c)D$$

If  $p^1 = p^2 = p$  then

$$r^1 = (p - c)\frac{D}{2}$$

$$r^2 = (p - c)\frac{D}{2}$$

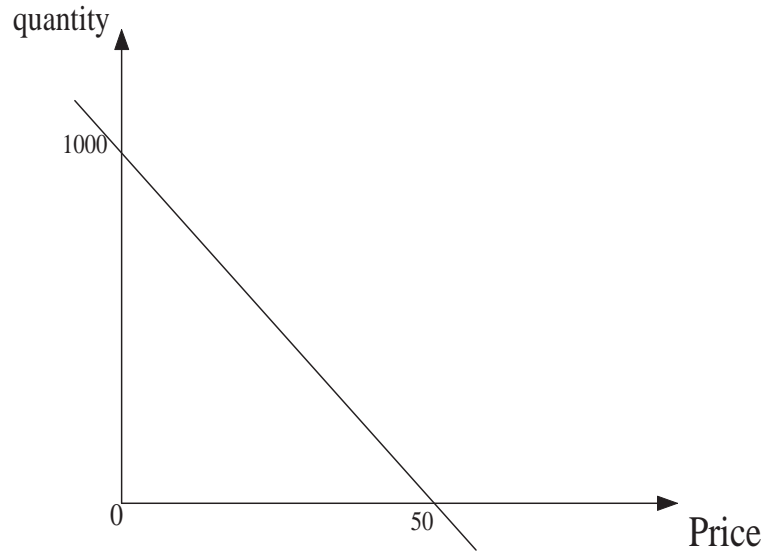


Figure 2. Market demand curve

If  $p = p^1 < p^2$  then

$$r^1 = (p - c)D$$

$$r^2 = 0$$

Table I. A stage game at price  $p$

		Agent 2		
		<i>Up (+0.1)</i>	<i>Stay (0)</i>	<i>Down (-0.1)</i>
Agent 1	<i>Up (+0.1)</i>	$(p-4.9)\frac{D}{2}, (p-4.9)\frac{D}{2}$	$0, (p-5)xD$	$0, (p-5.1)D$
	<i>Stay (0)</i>	$(p-5)D, 0$	$(p-5)\frac{D}{2}, (p-5)\frac{D}{2}$	$0, (p-5.1)D$
	<i>Down (-0.1)</i>	$(p-5.1)D, 0$	$(p-5.1)D, 0$	$(p-5.1)\frac{D}{2}, (p-5.1)\frac{D}{2}$

## 6.2. EXPERIMENTAL RESULTS

We test three cases. In each case, there are two agents. In the first case, both agents are simple reinforcement learning agents. In the second case, both are Q-learning agents. In the third one, both are Nash Q-learning agents. Each case is run for 10000 periods.

We record the agents' discounted accumulated profits at each time period and plot Agent 1' profits in Figure 3. Agent 1 and Agent 2 are symmetric. So we did not show the result of Agent 2 here. The discounted accumulated profit is used as a measure for the agent's performance during learning. We

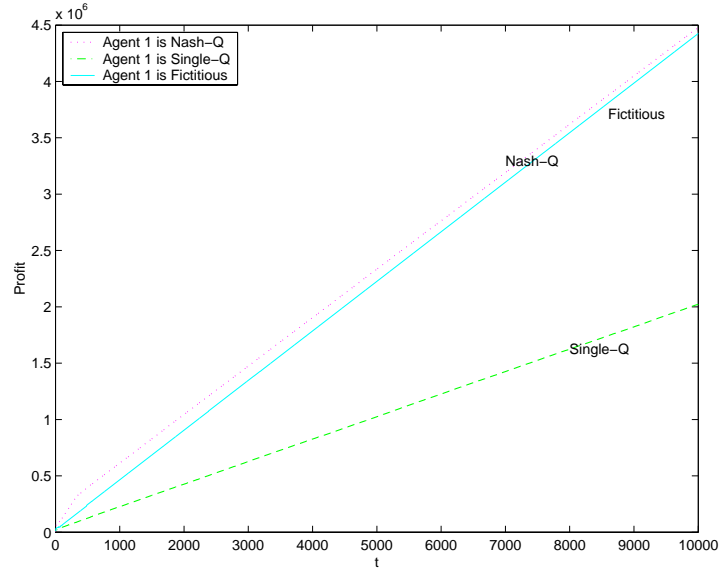


Figure 3. Agent 1's performance when Agent 2 is a Fictitious agent

compare Agent 1's performance under three different types. Figure 3 shows the accumulated profits Agent 1 receives in three different cases.

Our experimental result shows that an agent receives more profits when it uses Nash Q-learning method than any other learning method. This is not surprising since the Nash Q-learning method gives the agent the ability to model others, while taking its own long-term reward into account. The other two learning methods simply ignore the other agent, and thus fail to take the environmental response into account.

What we observe here is different from a previous paper (Wellman & Hu, 1998) which studies the case of conjecture equilibrium in which agents are trapped in a sub-optimal equilibrium in which everyone's belief is confirmed. Our agent in this paper is able to stay away from such a trap because it is able to observe the actions of others after they are chosen. It is also able to observe the market demand function and thus make the correct calculation on its own reward.

In addition to agent performance, we also observe the convergence to the equilibrium price under these different agent types.

Figure 6 shows the price trend when agent 2 is a Fictitious agent. The agents keep undercutting each other's price until they reach the minimum price, which is the marginal cost.

Figure 7 shows the price trend when agent 2 is Nash Q-learning agents. The market price moves rapidly upward until it reaches the highest price 50. The convergence is faster than in Q-learning. This is because agents with

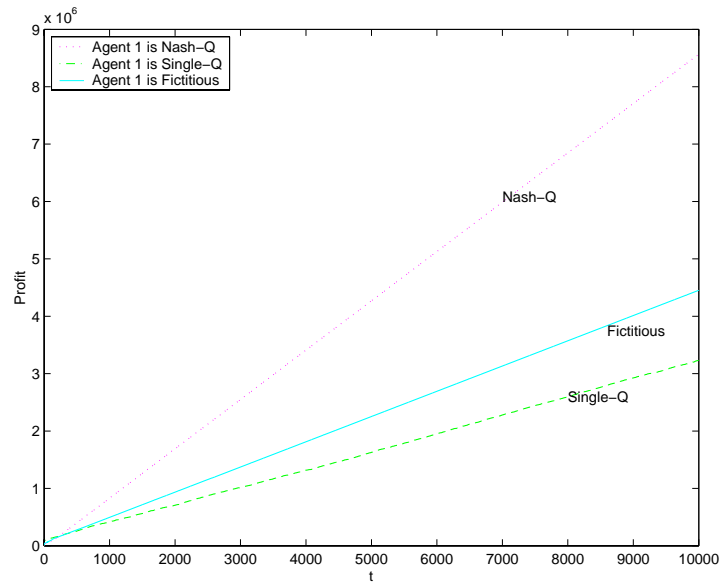


Figure 4. Agent 1's performance when Agent 2 is a Nash agent

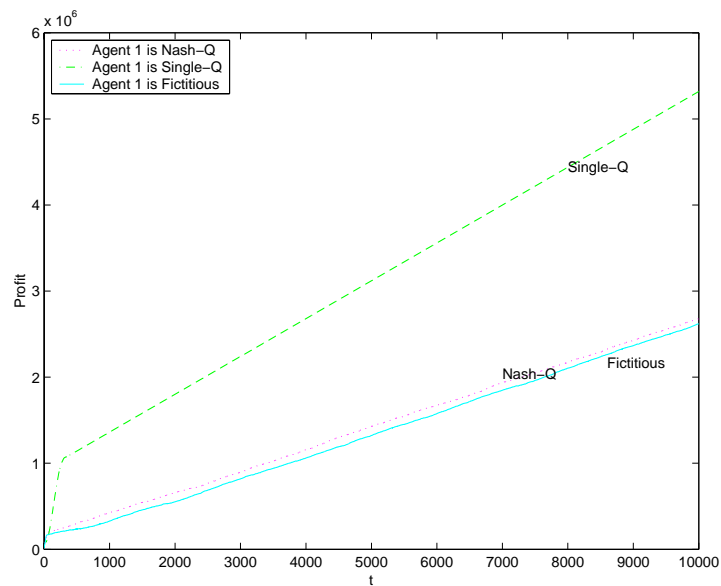


Figure 5. Agent 1's performance when Agent 2 is a Single-Q agent

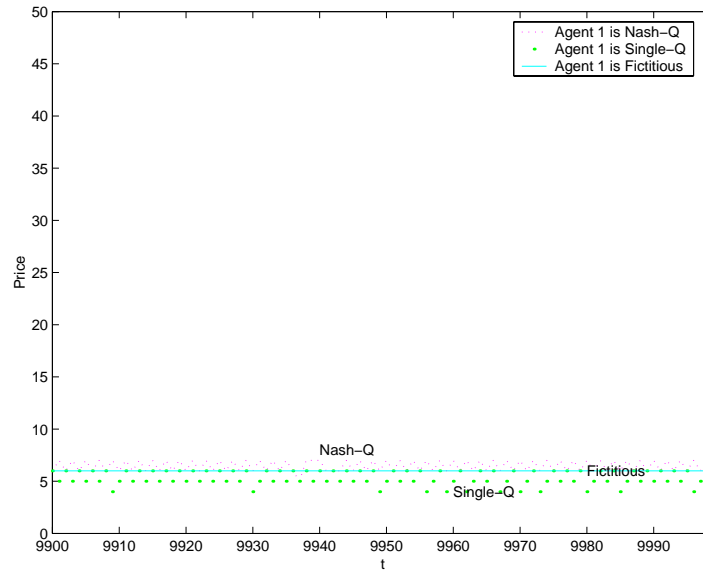


Figure 6. Market price under simple reinforcement learning agents

Nash Q-learning take each other into account. Thus agents are able to move to an equilibrium price that gives both of them better payoffs.

Figure 8 shows the price trend when agent 2 is a single Q-learning agents. The market price starts from the initial price, sometimes goes down a little bit, and always keep the trend of moving up. Eventually, the market price settles down at the highest price 50. This shows that with Q-table to store information about long-term payoff, an agent is able to learn to coordinate with other agents to reaching high payoff, and to avoid the lowest equilibrium price that yields little profit.

## 7. Summary and Future Work

We study three different types of pricing agents in a simulated economy. Each type of agent is based on a different learning method. The first method is simple reinforcement learning, which learns about an optimal action for one period based on the rewards it receives for that action. The second type is Q-learning method that learns about Q-values which represent long-term optimal values for the agent's own actions. The third type is Nash Q-learning method that learns about Q-values which represents long-term Nash equilibrium values for agent's joint actions.

We let these agents compete with each other in our simulation. In each simulation, there are two agents, and a fixed amount of customers. The agent

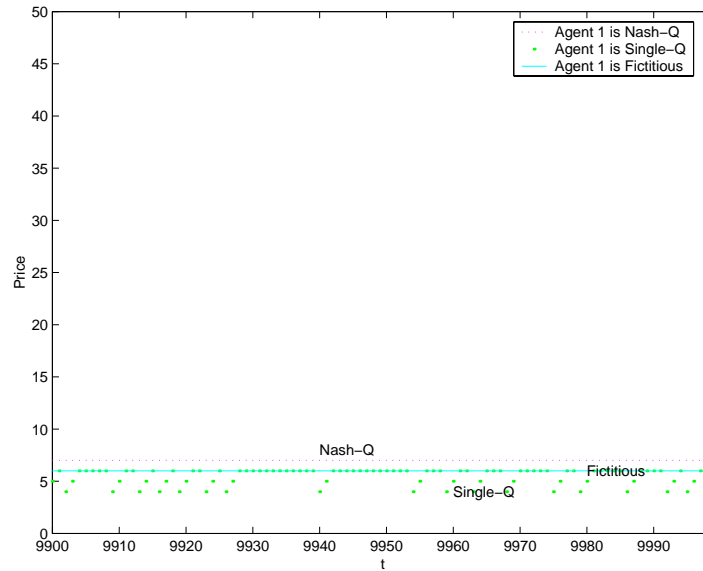


Figure 7. Market price under Q-learning agents

that charges a lower price will attract all the customers. When both firms charge the same price, they receive equal share of the market.

The simple reinforcement learning agent charges the price based on how well it did in previous period. If the agent attracted all customers in last period, then it will increase its price a little bit in this period, hoping to squeezing more profit out of the market. If it had equal share the market with the other agent, it will randomly choose a price at above or below the previous price. The random move is needed to explore new opportunities. If the agent did not get any customer in last period, it will decrease the price a little bit. When market consists of only such simple learning agent, the market price will be driven down to the minimum price that is equal to an agent's marginal cost. Here we assume the agents have the same marginal cost. At that price, no one is making profits. This result is consistent with predictions in economics literature.

The Q-learning agent updates its Q-values by adding the current reward with expected best future total rewards to previous Q-values at each time period. The agent chooses an action that has the highest Q-values. When all agents are Q-learning agents, the market price converges to the highest equilibrium price, which is the reservation price of the customers.<sup>1</sup> Both firms will charge such a price, and make large profits at such prices. Even though Q-learning improves an agent's performance relative to simple reinforcement

<sup>1</sup> A reservation price of a buyer is the maximum price the buyer is willing to pay for a product.

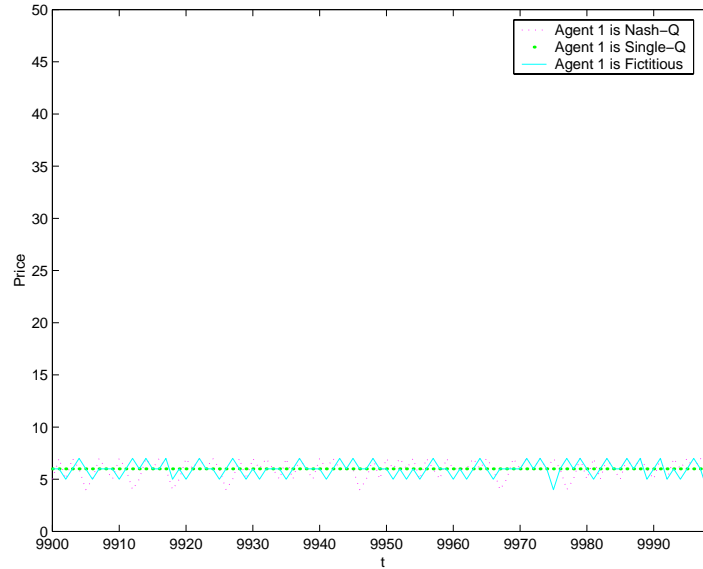


Figure 8. Market price under Nash Q-learning agents

learning, it is not the best learning method available. The fundamental flaw in Q-learning is that it treats other agents as part of an assumed stationary environment, ignoring the fact that other agents are learning and changing behavior at the same time. The existence of other agents makes the environment no longer stationary and calls for more sophisticated modeling. We have introduced Nash Q-learning method in a previous paper (Hu & Wellman, 1998), which modifies Q-learning in several significant ways.

The Nash Q-learning agent learns about Q-values which are functions of agents' joint actions. At each time period, the agent updates its Q-value with the current reward and the expected future Nash equilibrium rewards. The agent chooses an action that is part of a Nash equilibrium strategy. When all agents are Nash Q-learning agents, the market price also converges to the highest equilibrium price, but at faster speed than the Q-learning method. This leads the higher profits of agents than the Q-learning method for many periods. This result can be important when a firm's survival is determined by its profits during learning.

Our simulation shows that the learning methods that take future rewards into account perform better than the method that is myopic. Among the looking ahead methods, the one that takes other agents into account performs better than the one that ignores other agents. This suggests the importance of game theoretical modeling in online learning.

The equilibrium selection problem during learning is sidestepped in our simulation by requiring all agents to choose the same equilibrium for their

actions and Q-table updating. In our future work, we will further explore the impact of equilibrium selection on agents' performance and market price convergence.

## References

- Fudenberg, D., and Tirole, J. 1991. *Game Theory*. Cambridge, MA and London, England: The MIT Press.
- Hu, J., and Wellman, M. P. 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 242–250. Madison, WI: AAAI Press.
- Hu, J., and Wellman, M. P. 2000. Experimental results on Q-learning for general-sum stochastic games. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 407–414. Stanford, CA: Morgan Kaufmann Publishers.
- Kephart, J. O.; Hanson, J. E.; and Greenwald, A. R. 2000. Dynamic pricing by software agents.
- Narendra, K. S., and Thathachar, M. A. 1989. *Learning Automata: an Introduction*. Prentice Hall.
- Sridharan, M., and Tesauro, G. 2000. Multi-agent Q-learning and regression trees for automated pricing decisions. In *Proceedings of the Sventeenth International Conference on Machine Learning*, 927–934. San Francisco, CA: Morgan Kaufmann Publishers.
- Tesauro, G. J., and Kephart, J. O. 1999. Pricing in agent economies using multi-agent Q-learning. In *Proceedings of Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*.
- Thusijnsman, F. 1992. *Optimality and Equilibria in Stochastic Games*. Amsterdam: Centrum voor Wiskunde en Informatica.
- Tirole, J. 1988. *The Theory of Industrial Organization*. Cambridge, MA: The MIT Press.
- Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning* 3:279–292.
- Wellman, M. P., and Hu, J. 1998. Conjectural equilibrium in multiagent learning. *Machine Learning* 33:179–200.