Asymmetric Multiagent Reinforcement Learning in Pricing Applications

Ville Könönen and Erkki Oja
Neural Networks Research Centre
Helsinki University of Technology
P.O. Box 5400, FI-02015 HUT, FINLAND
{ville.kononen,erkki.oja}@hut.fi

***Abstract*— Two pricing problems are solved by using asymmetric multiagent reinforcement learning methods in this paper. In the first problem, a flat pricing scenario, there are two competing brokers that sell identical products to customers and compete on the basis of price. The second problem is a hierarchical pricing scenario where a supplier sells products to two competing brokers. In both cases, the methods converged and led to very promising results. We present a brief literature survey of pricing models based on reinforcement learning, introduce the basic concepts of Markov games and solve two pricing problems based on multiagent reinforcement learning.**

I. Introduction

Reinforcement learning methods have recently been established as practical tools for solving Markov decision processes. The main assumption behind these models is that the environment of the learning agent obeys the Markov property, i.e. state transition probabilities depend only on the state of environment and the action selections of the learning agent. For example, in multiagent settings, the Markov property does not always hold and this can lead to suboptimal results. One way to circumvent this problem is to use Markov games which are natural extensions of Markov decision processes to multiagent settings.

The main aim of this paper is to test asymmetric multiagent reinforcement learning methods with two pricing problems. In the first problem, there are two competing brokers that sell identical products to customers and thus compete on the basis of price. This pricing problem was originally proposed by Tesauro and Kephart in [10], where the problem was solved by using a single-agent reinforcement learning method. The proposed method led to very good results in the cases where only one of the agents is learning and the other keeps its pricing strategy fixed. If both agents learn, the proposed method did not always converge. In this paper, we model the problem as a Markov game and solve it by using asymmetric multiagent reinforcement learning. The proposed method converged in every case and the results were very promising. Moreover, we propose and solve a two-level pricing problem with two competing brokers and a supplier that sells products to these brokers.

The idea of using a heuristic approach to model foresight based agent economies was originally proposed by Tesauro and Kephart in [9]. The approach was then extended to utilize Q-learning in [10]. When pricing applications contain a large number of possible prices, the state and action spaces become huge and lookup-table-based reinforcement learning methods become infeasible. To overcome this problem, the Q-function was approximated with different function approximators in [8] and [7].

Our previous contributions in the field of multiagent reinforcement learning include an asymmetric multiagent reinforcement learning model [3]. Additionally, we have proposed numerical methods for multiagent reinforcement learning in [4] and [5].

The paper is organized as follows. In Section II, mathematical preliminaries of game theory and the relevant solution concepts are covered in brief. In Section III, we go briefly through the theory of multiagent reinforcement learning based on Markov games and in Section IV, we present two pricing problems and solve these problems by using asymmetric multiagent reinforcement learning. In the final section, concluding remarks and some suggestions for the further study are presented.

II. Game Theory

This section is mainly concerned with the basic problem settings and definitions of game theory. We start with some preliminary information about mathematical games and then proceed to their solution concepts which are essential for the rest of the paper.

A. Basic Concepts

Mathematical games can be represented in different forms. The most important forms are the *extensive* form and the *strategic* form. Although the extensive form is the most richly structured way to describe game situations, the strategic form is conceptually simpler and it can be derived from the extensive form. In this paper, we use games in strategic form for making decisions at each time step.

Games in strategic form are usually referred to as *matrix games* and particularly in the case of two players, if the payoff matrices for both players are separated, as *bimatrix games*. In general, an N-person matrix game is defined as follows:

Definition 1: A *matrix game* is a tuple $\Gamma = (A^1, \ldots, A^N, r^1, \ldots, r^N)$, where N is the number of players, A^i is the strategy space for player i and

$r^i : A^1 \times A^2 \times \ldots \times A^N \rightarrow \mathbb{R}$ is the payoff function for player i.

In a matrix game, each player i simultaneously implements a strategy $a^i \in A^i$. In addition to pure strategies A^i, we allow the possibility that the player uses a random (mixed) strategy. If we denote the space of probability distributions over a set A by $\Delta(A)$, a randomization by a player over his pure strategies is denoted by $\sigma^i \in \Sigma^i \equiv \Delta(A^i)$.

B. Equilibrium Concepts

In decision problems with only one decision maker, it is adequate to maximize the expected utility of the decision maker. However, in games there are many players and we need to define more elaborated solution concepts. Next we will shortly present two relevant solution concepts of matrix games.

Definition 2: If N is the number of players, the strategies $\sigma_*^i, \ldots, \sigma_*^N$ constitute a *Nash equilibrium* solution of the game if the following inequality holds for all $\sigma^i \in \Sigma^i$ and for all i:

$$r^i(\sigma_*^1, \ldots, \sigma_*^{i-1}, \sigma^i, \sigma_*^{i+1}, \ldots, \sigma_*^N) \leq r^i(\sigma_*^1, \ldots, \sigma_*^N)$$

The idea of the Nash equilibrium solution is that the strategy choice of each player is a best response to his opponents' play and therefore there is no need for deviation from this equilibrium point for any player alone. Thus, the concept of Nash equilibrium solution provides a reasonable solution concept for a matrix game when the roles of the players are symmetric. However, there are decision problems in which one of the players has the ability to enforce his strategy to other players. For solving these kind of optimization problems we have to use a hierarchical equilibrium solution concept, i.e. the *Stackelberg equilibrium* concept. In the two-player case, where one player is acting as the leader (player 1) and the another as the follower (player 2), the leader enforces his strategy to the opponent and the follower reacts rationally to this enforcement.

The basic idea is that the leader enforces his strategy so that he enforces the opponent to select the response that leads to the optimal response for the leader. Algorithmically, in the case of finite bimatrix games where player 1 is the leader and player 2 is the follower, obtaining a Stackelberg solution $(a_s^1, a_s^2(a^1))$ can be seen as the following two-step algorithm:

1) $a_s^2(a^1) = \arg\max_{a^2 \in A^2} r^2(a^1, a^2)$
2) $a_s^1 = \arg\max_{a^1 \in A^1} r^1(a^1, a_s^2(a^1))$

In the step 1, the follower's strategy is expressed as a function of the leader's strategy. In the step 2, the leader maximizes his own utility by selecting the optimal strategy pair. The only requirement is that the follower's response is unique; if this is not the case, some additional restrictions must be set.

Note that saying that the leader is capable to enforce his action choice to the follower does not always mean that the leader has an advantage over the follower. In some games, the leader relinquishes his power by announcing his strategy first.

III. Multiagent Reinforcement Learning

With two or more agents in the environment, the fundamental problem with single-agent Markov Decision Processes (MDPs) is that the approach treats the other agents as a part of the environment and thus ignores the fact that the decisions of the other agents may influence the state of the environment.

One possible solution is to use competitive multiagent MDPs, i.e. *Markov games*. In a Markov game, the process changes its state according to the action choices of all the agents and can thus be seen as a multicontroller MDP. Formally, we define a Markov game as follows:

Definition 3: A *Markov game* (*stochastic game*) is defined as a tuple $(S, A^1, \ldots, A^N, p, r^1, \ldots, r^N)$, where N is the number of agents, S is the set of all states, A^i is the set of all actions for each agent $i \in \{1, N\}$, $p : S \times A^1 \times \ldots \times A^N \rightarrow \Delta(S)$ is the state transition function, $r^i : S \times A^1 \times \ldots \times A^N \rightarrow \mathbb{R}$ is the reward function for agent i. $\Delta(S)$ is the set of probability distributions over the set S.

Again, as in the case of single-agent MDP, we need a policy π^i for each agent i (the policy is assumed to be stationary):

$$\pi^i : S \rightarrow A^i, \forall i \in \{1, N\}. \tag{1}$$

In multiagent systems this policy function is not necessarily deterministic. However, here we assume that the randomization is performed inside the policy function and therefore π^i returns actions directly. The expected value of the discounted utility R^i for the agent i is the following:

$$\begin{aligned} V^i_{\pi^1,\ldots,\pi^N}(s) &= E_{\pi^1,\ldots,\pi^N}[R^i | s_0 = s] \\ &= E_{\pi^1,\ldots,\pi^N}\left[\sum_{t=0}^{\infty} \gamma^t r^i_{t+1} | s_0 = s\right], \end{aligned} \tag{2}$$

where r^i_{t+1} is the immediate reward for agent i after the state transition and γ is a discount factor. Moreover, the value for each state-action pair is

$$\begin{aligned} &Q^i_{\pi^1,\ldots,\pi^N}(s, a^1, \ldots, a^N) \\ &= E_{\pi^1,\ldots,\pi^N}[R^i | s_0 = s, a_0^1 = a^1, \ldots, a_0^N = a^N] \\ &= r^i(s, a^1, \ldots, a^N) \\ &+ \gamma \sum_{s'} p(s' | s, a^1, \ldots, a^N) V^i_{\pi^1,\ldots,\pi^N}(s'). \end{aligned} \tag{3}$$

Contrast to single-agent MDPs, finding the optimal policy π_*^i for each agent i can be seen as a game theoretical problem where the strategies the players can choose are the policies defined in Eq. (1).

A. Solving Markov Games

In the case of multiagent reinforcement learning, it is not enough to maximize the expected utility of individual agents. Instead, our goal is to find an equilibrium policy of the Markov game, e.g. a Nash equilibrium policy. The Nash equilibrium policy is defined as follows:

Definition 4: If N is the number of agents and Π^i is the policy space for the agent i, the policies $\pi_*^1, \ldots, \pi_*^N$ constitute a Nash equilibrium solution of the game if the following inequality holds for all $\pi^i \in \Pi^i$ and for all i in each state $s \in S$:

$$V^i_{\pi^1_*,\ldots,\pi^i,\ldots,\pi^N_*}(s) \leq V^i_{\pi^1_*,\ldots,\pi^N_*}(s)$$

It is noteworthy that Definition 4 coincides with Definition 2 when individual strategies are replaced with policies. The Stackelberg equilibrium concept can be extended for policies in similar fashion. We refer to methods built on Markov games with the Nash equilibrium concept as symmetric methods and to methods that utilize the Stackelberg equilibrium concept as asymmetric methods.

For brevity, learning algorithms are presented next only in the case of two agents and in the asymmetric model the agent one is acting as the leader and the agent two as the follower.

B. Symmetric Learning in Markov Games

As in the case of single agent reinforcement learning, Q-values defined in Eq. (3) can be learned from observations on-line using some iterative algorithm. For example, in the two-agent case, if we use Q-learning, the update rule for the agent 1 is [2]:

$$\begin{aligned} Q^1_{t+1}(s_t, a^1_t, a^2_t) &= (1-\alpha_t) Q^1_t(s_t, a^1_t, a^2_t) \\ &+ \alpha_t[r^1_{t+1} + \gamma \text{Nash}\{Q^1_t(s_{t+1})\}], \end{aligned} \tag{4}$$

where $\text{Nash}\{Q^1_t(s_{t+1})\}$ is the Nash equilibrium outcome of the bimatrix game defined by the payoff function $Q^1_t(s_{t+1})$. The corresponding update rule for the agent 2 is symmetric.

Note that it is guaranteed that every finite matrix game possesses at least one Nash equilibrium in mixed strategies. However, there exists not necessarily Nash equilibrium point in pure strategies and therefore $\text{Nash}\{Q^1_t(s_{t+1})\}$ in Eq. (4) returns the value of a mixed strategy equilibrium.

C. Asymmetric Learning in Markov Games

In the asymmetric case with Q-learning, we get the following update rules for the agents:

$$\begin{aligned} Q^1_{t+1}(s_t, a^1_t, a^2_t) &= (1-\alpha_t) Q^1_t(s_t, a^1_t, a^2_t) + \alpha_t[r^1_{t+1} \\ &+ \gamma \max_{b \in A^1} Q^1_t(s_{t+1}, b, Tb)] \end{aligned} \tag{5}$$

$$\begin{aligned} Q^2_{t+1}(s_t, a^1_t, a^2_t) &= (1-\alpha_t) Q^2_t(s_t, a^1_t, a^2_t) + \alpha_t[r^2_{t+1} \\ &+ \gamma \max_{b \in A^2} Q^2_t(s_{t+1}, g(s_{t+1}, a^c_{t+1}), b)], \end{aligned} \tag{6}$$

where $g(s_t, a^c_t)$ is the leader's enforcement and T is a mapping $T : A^1 \rightarrow A^2$ that conducts the follower's best response to the leader's enforcement.

Above presented algorithms do not define how one selects the current state-action tuple (s_t, a^1_t, a^2_t). For example, it is possible to select states and actions pure randomly. However, it is often more efficient to explore state-action space by calculating an equilibrium solution of the matrix game associated with the current state and then deviate from this solution with some small probability, e.g. by using softmax action selection scheme.

IV. Pricing Problems

In this section, we apply above discussed asymmetric multiagent reinforcement learning method to two pricing problems. In both problems, there are two competing agents (brokers) that sell identical products and compete against each other on the basis of price. At each time step, one of the brokers decides its new price based on the opponent's, i.e. other broker's, current price. After the price has been set, the customer either buys the product at the offered price or buys not the product at all. The objective of the agents is to maximize their profits. We begin the section by modeling the interaction between the two brokers as an asymmetric multiagent reinforcement learning problem. Additionally, we propose a hierarchical pricing problem of three agents, in which one of the agents is acting as a supplier that sells products to the brokers.

A. Flat Pricing Problem

In [10], Tesauro and Kephart modeled the interaction between two brokers as a single-agent reinforcement learning problem in which the goal of the learning agent is to find the pricing strategy that maximizes its long time profits. Reinforcement learning aids the agents to prevent "price wars", i.e. repeated price reductions among the brokers. As a consequence of a price war, the prices would go very low and the overall profits would also be small. Tesauro and Kephart reported very good performance of the approach when one of the brokers keeps its pricing strategy fixed. However, if both brokers try to learn simultaneously, the Markov property assumed in the theory of MDPs does not hold any more and the learning system may encounter serious convergence problems. In this paper, we model the pricing system as a Markov game and test the proposed learning system with two economical models.

In the simple economical model (the Shopbot model [1]), the customer buys the product from the broker with the lowest price. At each time step, after the customer has done his purchase decision, brokers get their immediate profits according to the utility functions defined as follows:

$$u^1(p^1, p^2) = \begin{cases} p^1 - c & \text{if } p^1 \leq p^2 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and

$$u^2(p^1, p^2) = \begin{cases} p^2 - c & \text{if } p^1 > p^2 \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where $p^1, p^2 \in P$ are the current prices of the broker 1 and the broker 2, respectively, and the $c \in [0,1]$ is a fixed marginal cost of the product. In this paper, all prices lie in the unit interval and the parameter $c = 0.2$.

In the second, more complex economical model (the Price-Quality model [6]), there is a quality parameter associated with each broker and the customers make their purchase decisions in a quality-aware manner. Denoting the quality parameter for the broker 1 as q^1 and for the broker 2 as q^2 ($q^1 > q^2$), we get the following utility functions for the brokers [6]:

$$u^1(p^1,p^2) = \begin{cases} (q^1-p^1)(p^1-c(q^1)) & \text{if } 0 \le p^1 \le p^2 \\ & \text{or } p^1 > q^2 \\ (q^1-q^2)(p^1-c(q^1)) & \text{if } p^2 < p^1 < q^2 \end{cases} \tag{9}$$

and

$$u^2(p^1,p^2) = \begin{cases} (q^2-p^2)(p^2-c(q^2)) & \text{if } 0 \le p^2 < p^1 \\ 0 & p^2 \ge p^1, \end{cases} \tag{10}$$

where $c(q^i)$ represents the cost of producing the product i. Note that we assume here that there is an infinite number of customers who all behave as described in [6]. Hence, the above utility functions are simply profit expectations for the brokers. In this work, we use the following linear cost function:

$$c(q^i) = 0.1(1.0 + q^i). \tag{11}$$

Furthermore, we set the quality parameters as follows: $q^1 = 1.0$ and $q^2 = 0.9$. This parameter setting was observed to generate price wars when the agents use simple myopic pricing strategy (i.e. they make their decisions directly based on the above declared utility functions) in [10].

We make the assumption that the brokers do not decide their decisions simultaneously, i.e. there is an ordering among the decision makers. Hence, we model the system with the following Markov game endowed with the asymmetric equilibrium concept:

- The state is the current price of the broker 2.
- The broker 1 is acting as the leader and hence decides its price prior to the broker 2. Hence, as the state is the current price of the broker 2, the utility of the broker 1 depends only on its price selection and the current state.
- The broker 2 is the follower and its utility value depends on the leader's enforcement and its own price selection.

At each time step, the broker 1 calculates a Stackelberg equilibrium point of the matrix game associated to the current state and makes its pricing decision based on this solution. After that, the broker 1 announces its price decision to the broker 2 who, in its turn, maximizes its utility value based on this enforcement. This process is illustrated in Fig. 1.

The corresponding update equations for the brokers 1 and 2 are as follows:

$$\begin{aligned} Q^1_{t+1}(p^2_{t-1},p^1_t,p^2_t) &= (1-\alpha_t)Q^1_t(p^2_{t-1},p^1_t,p^2_t) \\ &+ \alpha_t[u^1(p^1_t,p^2_{t-1}) + \gamma \max_{b\in P} Q^1_t(p^2_t,b,Tb)] \end{aligned} \tag{12}$$

and

$$\begin{aligned} Q^2_{t+1}(p^2_{t-1},p^1_t,p^2_t) &= (1-\alpha_t)Q^2_t(p^2_{t-1},p^1_t,p^2_t) \\ &+ \alpha_t[u^2(p^1_t,p^2_t) + \gamma \max_{b\in P} Q^2_t(p^2_t,g(p^2_t,a^c_t),b)], \end{aligned} \tag{13}$$

where γ is the discount factor, operator Tb conducts the follower's response to the leader's action choice b and $g(\cdot)$ is the leader's enforcement. Note that the learning method does not need any prior model of the opponent, not even the above defined utility functions.

In our test runs, the number of different pricing options was 25 for both agents and the Q-learning was implemented

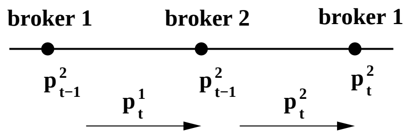

Fig. 1. Timeline of the price decisions in the flat pricing problem. Price symbols below dots describe states and symbols above arrows price decisions.

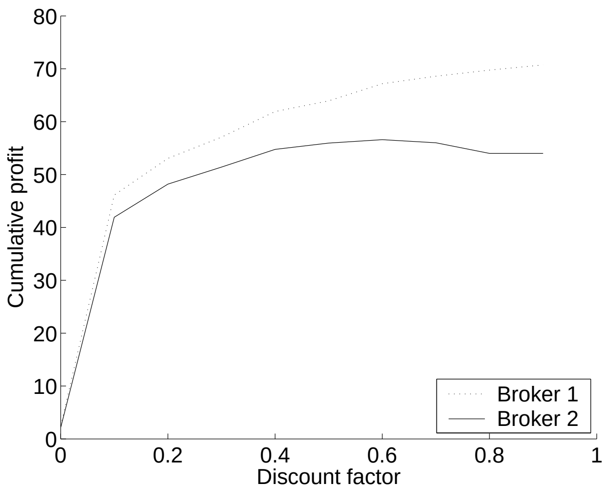

Fig. 2. Averaged profits in the flat pricing model with the Shopbot pricing function. All data points are averages of 1000 test runs each containing 100 pricing decisions for both agents.

by using a simple tabular implementation. During training each state-action tuple was visited 1000 times. Learning rate parameter α was decayed according to the following equation:

$$\alpha = \frac{1.0}{n(p^2_{t-1},p^1_t,p^2_t)}, \tag{14}$$

where $n(\cdot)$ is the number of visits in the state-action tuple. In the testing phase, the initial state (price of the broker 2) was selected randomly and one test run consisted of 100 pricing decisions per broker. In Fig. 2, the cumulative profit (average from 1000 test runs) of each agent is plotted against the discount factor γ in the case of the Shopbot pricing model. Respectively, in Fig. 3 the cumulative profit is shown in the case of the Price-Quality model. In the Shopbot model, the average profit of the broker 1 grows monotonically as the discount factor increases. Also the profit of the broker 2 increases albeit not monotonically. Moreover, the use of small discount factor $\gamma = 0.1$, corresponding to very shallow lookahead, leads to relatively high profits compared to $\gamma = 0.0$. The use of higher discount factors further increases profits but the growth is not so dramatic. In the Price-Quality model, the profits grow steadily as the discount factor is increased.

The convergence of the agents' Q-value tables in the case of Shopbot model is illustrated in Fig. 4, where the Euclidean distance between Q-value vectors from consecutive training rounds is plotted against the round number. Two cases with discount factors 0.3 and 0.9 are plotted for both brokers. It can be seen that the algorithm converged very fast in every

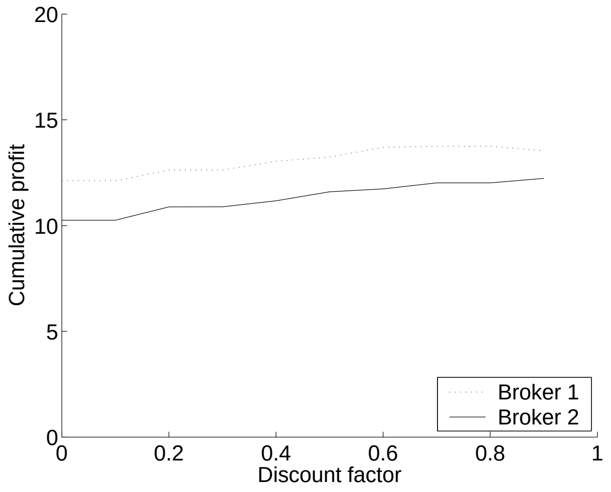

Fig. 3. Averaged profits in the flat pricing model with the Price-Quality pricing function. All data points are averages of 1000 test runs each containing 100 pricing decisions for both agents.

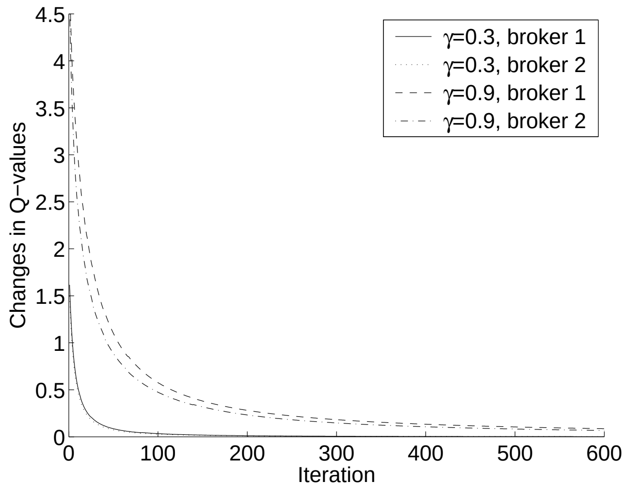

Fig. 4. Convergence of the Q-values in the flat pricing model.

case, although with high γ values the convergence is much slower than with low values of γ. The convergence properties of the algorithm in the case of the Price-Quality model are analogous.

B. Two-Layer Pricing Problem

We now extend the above system to handle two-layer agent hierarchies. In addition to the flat pricing problem setting, there is now a supplier that sells products to both brokers. At each time step, one of the brokers decides its new price based on the opponent's current price (other broker) and the price set by the supplier. The supplier, in its turn, decides its action based on the asymmetric solution concept. The customer buys the product of the lowest price. After the customer's decision, the brokers get their profits according to their immediate utility functions presented in Eqs. (15) and (16). The utility values for the supplier are shown in Eqs. (17) and (18) in the case where the brokers 1 and 2 are charged, respectively.

$$u^1(p^1,p^2,s;l) = \begin{cases} p^1 - s & \text{if } p^1 \le p^2 \text{ and } s < lp^1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$$u^2(p^1,p^2,s;l) = \begin{cases} p^2 - s & \text{if } p^1 > p^2 \text{ and } s < lp^2 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$u^{s1}(p^1,p^2,s;l) = \begin{cases} s - c & \text{if } p^1 \le p^2 \text{ and } s < lp^1 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$u^{s2}(p^1,p^2,s;l) = \begin{cases} s - c & \text{if } p^1 > p^2 \text{ and } s < lp^2 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

In Eqs. (15)–(18), p^1 and p^2 are the prices of the brokers 1 and 2, respectively, s is the price of the supplier and $l \in [0,1]$ is the largest fraction of the broker's price that the broker is willing to pay to the supplier. As in the flat pricing problem, c is a marginal cost of the product. c could also be associated with some quality parameter, perhaps different for each broker. However, in this study, the parameter has a fixed and equal value for each broker.

At each time step, the customer purchases the product from the broker having the lowest price. If the supplier is charging too much from the broker (expected profit of the broker is too low), the broker does not buy the product from the supplier and the utility drops to zero for the supplier and the broker.

In this problem, we use reinforcement learning to aid the agents in anticipating the long-time consequences of their price decisions on both levels of the agent hierarchy. The supplier does not know the fraction l and therefore it is reasonable to apply learning, e.g. reinforcement learning, also for the supplier.

We make the simplifying assumption that the broker 2 keeps its pricing strategy fixed, i.e. it decides its price based on the immediate utility value defined in Eq. (16). Further, the supplier also keeps its pricing strategy fixed with the broker 2. Fig. 5 illustrates this relationship.

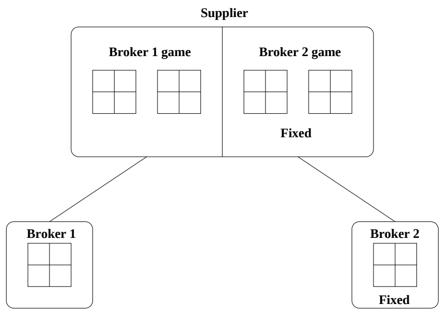

Fig. 5. Supplier-broker relationship.

In the corresponding Markov game, the state is the opponent's (other broker's) last price and the action is the current price decision. The update rules for the supplier and the broker 1 are as follows:

$$\begin{aligned} Q^{s1}_{t+1}(p^2_t, s_t, p^1_t) &= (1-\alpha_t)Q^{s1}_t(p^2_t, s_t, p^1_t) \\ &+ \alpha_t[u^{s1}(p^1_t, p^2_t, s_t; l) + \gamma \max_{b \in P} Q^{s1}_t(p^2_{t+1}, b, Tb)] \end{aligned} \tag{19}$$

and

$$Q^1_{t+1}(p^2_t, s_t, p^1_t) = (1-\alpha_t) Q^1_t(p^2_t, s_t, p^1_t) + \alpha_t[u^1(p^1_t, p^2_t, s_t; l) + \gamma \max_{b \in P} Q^1_t(p^2_{t+1}, g(p^2_{t+1}, a^c_{t+1}), b)], \qquad (20)$$

where p^2_{t+1} is obtained from the fixed game between the supplier and the broker 2. The Q-value tables are initialized by using profit functions (15) and (17).

The parameter l has a value of 0.8 and the producing cost for the supplier is $c = 0.2$ per product. The number of the pricing options was 25 for all agents and the maximum price for the supplier was 0.8. The training phase was conducted as in the case of the flat pricing model. In the testing phase, the initial prices were selected randomly and one test run consisted of 100 pricing decisions per broker. In Fig. 6, the cumulative profit (average from 1000 test runs) of each agent is plotted against the discount factor γ. As we can see from this figure, the average profit of the supplier grows monotonically as the discount factor increases. Moreover, the broker 1 learns a pricing strategy that leads to a moderate growth in the profits compared to the myopic case. The optimizing broker (broker 1) rises its price to the maximum in some situations and therefore has slightly lower profits than the static broker. However, the cumulative profits are much higher also for the broker 1 than in the myopic case.

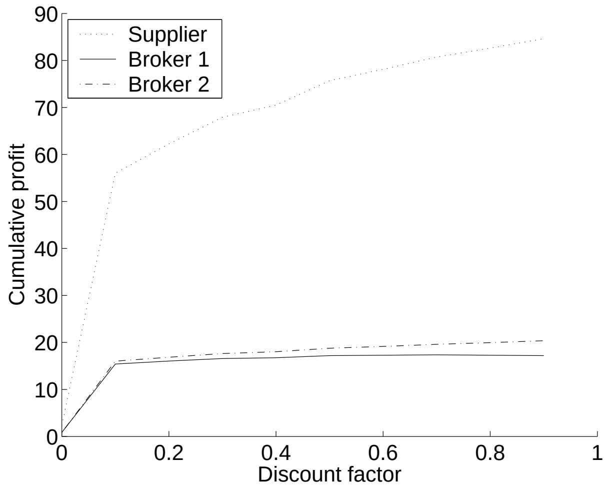

Fig. 6. Averaged profits in the two-layer pricing model. All data points are averages of 1000 test runs each containing 100 pricing decisions for both agents. The maximal possible profit is 100 for the brokers and 200 for the supplier.

The convergence of the agents' Q-value tables is illustrated in Fig. 7. The convergence properties of the algorithm were similar to the flat pricing model.

V. Conclusions and Future Research

Two pricing problems based on asymmetric multiagent reinforcement learning were presented in this paper. The proposed learning methods have stronger convergence properties than single-agent reinforcement learning methods in multiagent

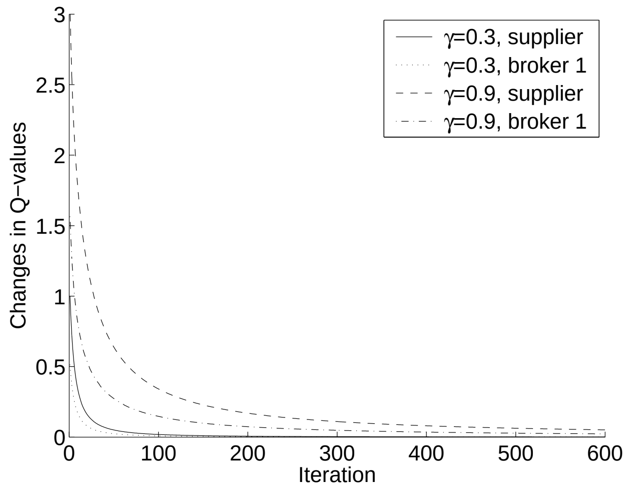

Fig. 7. Convergence of the Q-values in the two-layer pricing model.

environments. The methods converged in every test case and led to very promising results.

Tabular implementations of the multiagent reinforcement learning based pricing models become intractable as the number of pricing options increases. Therefore, we are going to apply numerical methods, both value function based and direct policy gradient methods, to these pricing problems.

References

[1] Amy R. Greenwald and Jeffrey O. Kephart. Shopbots and pricebots. In *Proceedings of the International Conference on Artificial Intelligence (IJCAI'99)*, Stockholm, Sweden, 1999. AAAI Press.

[2] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, Madison, WI, 1998. Morgan Kaufmann Publishers.

[3] Ville J. Könönen. Asymmetric multiagent reinforcement learning. In *Proceedings of the 2003 WIC International Conference on Intelligent Agent Technology (IAT-2003)*, Halifax, Canada, 2003. IEEE Press.

[4] Ville J. Könönen. Gradient based method for symmetric and asymmetric multiagent reinforcement learning. In *Proceedings of the Fourth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2003)*, Hong Kong, China, 2003. Springer-Verlag.

[5] Ville J. Könönen. Policy gradient method for multiagent reinforcement learning. In *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, 2003.

[6] Jakka Sairamesh and Jeffrey O. Kephart. Price dynamics of vertically differentiated information markets. In *Proceedings of the First International Conference on Information and Computational Economics (ICE'98)*, Charleston, SC, 1998. ACM Press.

[7] Manu Sridharan and Gerald Tesauro. Multi-agent Q-learning and regression trees for automated pricing decisions. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, Stanford, CA, 2000. AAAI Press.

[8] Gerald Tesauro. Pricing in agent economies using neural networks and multi-agent Q-learning. In *Sequence Learning: Paradigms, Algorithms, and Applications*, volume 1828 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[9] Gerald Tesauro and Jeffrey O. Kephart. Foresight-based pricing algorithms in an economy of software agents. In *Proceedings of the First International Conference on Information and Computational Economics (ICE'98)*, Charleston, SC, 1998. ACM Press.

[10] Gerald Tesauro and Jeffrey O. Kephart. Pricing in agent economies using multi-agent Q-learning. In *Proceedings of Workshop on Game Theoretic and Decision Theoretic Agents (GTDT'99)*, London, England, 1999.