Web Intelligence and Agent Systems: An international journal 2 (2004) 105–121
IOS Press

Asymmetric multiagent reinforcement learning

Ville Könönen
Neural Networks Research Centre, Helsinki University of Technology, P.O. Box 5400, FI-02015 HUT, Finland
Tel.: +358 9 451 5024; Fax: +358 9 451 3277; E-mail: ville.kononen@hut.fi

Abstract. A novel model for asymmetric multiagent reinforcement learning is introduced in this paper. The model addresses the problem where the information states of the agents involved in the learning task are not equal; some agents (leaders) have information how their opponents (followers) will select their actions and based on this information leaders encourage followers to select actions that lead to improved payoffs for the leaders. This kind of configuration arises e.g. in semi-centralized multiagent systems with an external global utility associated to the system. We present a brief literature survey of multiagent reinforcement learning based on Markov games and then propose an asymmetric learning model that utilizes the theory of Markov games. Additionally, we construct a practical learning method based on the proposed learning model and study its convergence properties. Finally, we test our model with a simple example problem and a larger two-layer pricing application.

Keywords: Multiagent reinforcement learning, Markov games, Nash equilibrium, Stackelberg equilibrium, convergence

1. Introduction

Reinforcement learning methods have attained lots of attention in recent years [12]. Although these methods and procedures were earlier considered to be too ambitious and to lack a firm foundation, they have now been established as practical methods for solving Markov decision processes. However, the requirement for reinforcement learning methods to work is that the problem domain where the methods are applied obeys the Markov property. In many real-world problems this property is not fully satisfied but many reinforcement learning methods can still handle these situations relatively well. Especially, in the case of two or more decision makers in the same system the Markov property does not hold and more advanced methods should be used instead. One possible solution is to use competitive Markov decision processes since there exists a suitable theoretical framework for these processes and some learning methods have also been proposed.

Earlier work utilizing Markov games in multiagent reinforcement learning deals with a symmetric learning model, i.e. each agent in the system has equal information in the action selection task. In this paper we introduce an asymmetric learning model where information states are not equal; some agents (leaders) try to encourage agents with less information (followers) to select actions that lead to improved payoffs for the leaders. This approach has several key benefits over the symmetric learning model:

1. Opposite to the symmetric learning model, the value of the equilibrium point is unique for the leader in each state. This uniqueness implies strong convergence properties of the model.
2. Asymmetric solution always exists in pure strategies. This leads to very fast equilibrium point evaluation. However, it is also possible to calculate mixed strategy equilibria by using bi-level optimization techniques (see [2]).
3. Many problem instances are inherently hierarchical. This is true e.g. in semi-centralized multiagent systems. Additionally, space and computational requirements of the asymmetric learning model are often lower than in the symmetric case.

Although it is natural to apply asymmetric learning techniques with inherently hierarchical problems, it is also possible to simply enforce some ordering among

learners and use the asymmetric learning model in the place of the symmetric learning model in nonhierarchical problems. In many cases, e.g. in coordination problems and team games, this approach can still lead to very good results.

Multiagent reinforcement learning methods have been discussed earlier by many authors. Existing methods can be roughly divided into three distinct groups: 1) methods utilizing direct gradients of agents' value functions, 2) methods that estimate the value functions and then use this estimate to compute an equilibrium of the process and 3) methods that use direct policy gradients.

Early methods for multiagent reinforcement learning include e.g. [6] and [24]. The method presented in [6] uses a simplified version of Q-learning to estimate agents' value functions. This method can fail to converge in some difficult coordination problems and some improvements aiming to overcome these problems were published in [14] and [13]. Moreover, a set of performance comparisons between single-agent reinforcement learning and multiagent reinforcement learning were performed in [6]. In [24], a simple gradient-based method is used to optimize agents' value functions directly so that it is always a best response to opponents' changing strategies. In all of these papers, the methods are tested with repeated games and deterministic reward values. In [15], Kapetanakis et al. propose the technique that converge almost always in fully stochastic environments, i.e. when rewards are stochastic.

A more recent study falling into the third category is [5], in which uses a policy gradient method originally proposed by Sutton, McAllester, Singh and Mansour in [26], in multiagent context. The policy gradient method tries to find the optimal policy from a restricted class of parametrized policies. However, this method leans on the Markov property of the environment and is not thus directly suitable for multiagent domains. Bowling and Veloso solve this problem by using the WoLF principle [4] to adjust the learning rate so that the convergence is guaranteed (albeit only with very simple problems). Another study on policy gradients is [22], in which the VAPS framework originally proposed by Baird and Moore in [1] for single-agent domains is expanded for cooperative games.

The first learning method for multistate Markov games was proposed by Littman in [19]. He introduced a Q-learning method for Markov games with two players and a zero-sum payoff structure. This method is guaranteed to converge from arbitrary initial values to the optimal value functions. However, the zero-sum payoff structure can be a very restrictive requirement in some systems and thus Hu and Wellman extended this algorithm to general-sum Markov games in [11]. Unfortunately, their method is guaranteed to converge only under very restrictive conditions. Littman proposed a new method in [20], which relaxes these limitations by adding some additional (a priori) information about the roles of the agents in the system. Wang and Sandholm proposed a method that is guaranteed to converge with any team Markov game to the optimal Nash equilibrium in [30]. Conitzer and Sandholm presented an algorithm that converges to a Nash equilibrium in self-play and learn to play optimally against stationary opponents in [7].

It still remains as an open question if there exist computationally efficient methods for computing Nash equilibria of finite games. To overcome this problem, Greenwald and Hall proposed a multiagent reinforcement learning method that uses the correlated equilibrium concept in place of the Nash equilibrium in [10]. Correlated equilibrium points can be calculated using linear programming and thus the method remains tractable also with larger problem instances. Some complexity results about Nash equilibria can be found in [8].

A totally different approach to multiagent reinforcement learning is the COllective INtelligence (COIN) architecture proposed by Wolpert and Tumer [34]. The COIN architecture can been seen rather as a framework for adjusting single-agent reinforcement learning algorithms for multiagent domains than a standalone method for multiagent reinforcement learning. The main idea of COIN is to define reward signals for each agent according to some global fitness measure. Due to the overall structure of the COIN, the method is very scalable and remains robust as the problem size scales up. The COIN framework is also used in many large-scale realworld applications, see e.g. [35] and [33].

Our previous contributions in the field of multiagent reinforcement learning include an asymmetric multiagent reinforcement learning model [16]. The publication serves as a stepping stone for this more advanced work. Additionally, we have proposed numerical methods for multiagent reinforcement learning in [17] and [18].

We begin the paper by introducing the background and basic solution concepts of game theory. Then we briefly go through the theory behind Markov decision processes and introduce some learning methods applied to multiagent reinforcement learning problem. Finally,

we propose our asymmetric learning model for multiagent reinforcement learning, study its convergence properties and test it with example problems that have reasonable large state-action spaces.

2. 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.

2.1. 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.

Examples of both extensive form and strategic form games can be seen in Figs 1 and 2. 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 the player i and $r^i : A^1 \times A^2 \times \ldots \times A^N \rightarrow \boldsymbol{R}$ is the payoff function for the player i.

An example of a game in extensive form. Nodes 1.1 and 2.2 are decision nodes of the player 1 and 2, respectively. Each arch connected to a decision node (marked with a) is denoting the decision of the corresponding player. Dashed boxes are information states for the corresponding player, e.g. player 2 does not observe the actual strategy choice of the player 1. The ith number in a leaf node is the resulting payoff for the player i.

An example of a game in strategic form. The rows in the matrix represent strategies for the player 1 and columns for the player 2. In each entry of the matrix, the ith number is the payoff for the player i.

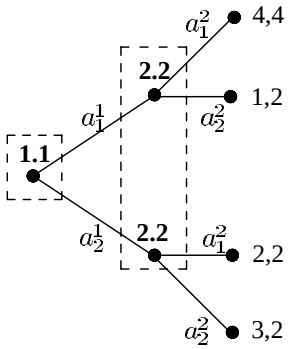

Fig. 1. An example of a game in extensive form. Nodes 1.1 and 2.2 are decision nodes of the player 1 and 2, respectively. Each arch connected to a decision node (marked with a) is denoting the decision of the corresponding player. Dashed boxes are information states for the corresponding player, e.g. player 2 does not observe the actual strategy choice of the player 1. The ith number in a leaf node is the resulting payoff for the player i.

	a_1^2	a_2^2
a_1^1	**2,2**	**4,0**
a_2^1	**1,0**	**3,1**

Fig. 2. An example of a game in strategic form. The rows in the matrix represent strategies for the player 1 and columns for the player 2. In each entry of the matrix, the ith number is the payoff for the player i.

In a matrix game, each player i simultaneously chooses 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 $\sigma^i \in \Sigma^i \equiv \Delta(A^i)$.

2.2. 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 $\sigma_*^1, \ldots, \sigma_*^N$ constitute a *Nash equilibrium* solution

of the game if the following inequality holds for all $\sigma^i \in \Sigma^i$ and for all i:

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

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. two-player *Stackelberg equilibrium* concept, where one player is acting as the leader (player 1) and the other as the follower (player 2). The leader enforces his strategy to the opponent and the follower is reacting rationally to this enforcement.

The basic idea is that the leader selects 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 the player 1 is the leader and the 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 step 1, the follower's strategy is expressed as a function of the leader's strategy. In 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.

As an example of the Stackelberg equilibrium solution concept, let us consider the game in Fig. 3. If we now stipulate that the player 1 is the leader and the player 2 is the follower, the player 1 can enforce his strategy to the player 2. Thus, before the player 1 enforces his strategy to the player 2, he has to consider the possible responses of the player 2 and then enforce the strategy which is the most favorable to him. In this example game, if the player 1 chooses the strategy a_1^1, then the best response of the player 2 is a_1^2. If the player 1 chooses a_2^1, then the follower's response is a_2^2 and if the player 1 chooses a_3^1, the player 2 responses using strategy a_3^2. Clearly, the player 1 maximizes his own utility by choosing strategy a_1^1 and thus the Stackelberg equilibrium solution is (a_1^1, a_1^2).

	a_1^2	a_2^2	a_3^2
a_1^1	**0,1**	**-2,-1**	$-\frac{3}{2}, \frac{2}{3}$
a_2^1	**-1,-2**	**-1,0**	**-3,-1**
a_3^1	**1,0**	**-2,-1**	$-2, \frac{1}{2}$

Fig. 3. An example matrix game. Its unique Nash equilibrium in pure strategies is (a_2^1, a_2^2). If the player 1 is the leader, the Stackelberg equilibrium solution is (a_1^1, a_1^2). Correspondingly, if the player 2 is the leader, (a_1^1, a_3^2) is the Stackelberg equilibrium solution [3].

An example matrix game. Its unique Nash equilibrium in pure strategies is (a_2^1, a_2^2). If the player 1 is the leader, the Stackelberg equilibrium solution is (a_1^1, a_1^2). Correspondingly, if the player 2 is the leader, (a_1^1, a_3^2) is the Stackelberg equilibrium solution [3].

3. Single-agent reinforcement learning

In this section, we briefly introduce the mathematical theory of noncompetitive Markov decision processes. In addition, practical solution methods for these processes are discussed at the end of this section.

3.1. Markov decision process

A fundamental concept in a *Markov Decision Process* is an *agent* that interacts with the environment in the manner illustrated in Fig. 4. The environment evolves (changes its state) probabilistically and for each state there is a set of possible actions that the agent may take. Every time the agent takes an action, a certain cost is incurred.

Formally, we define the Markov decision process as follows:

Definition 3. A *Markov Decision Process* (MDP) is a tuple (S, A, p, r), where S is the set of all states, A is the set of all actions, $p : S \times A \to \Delta(S)$ is the state transition function and $r : S \times A \to \boldsymbol{R}$ is the reward function. $\Delta(S)$ is the set of probability distributions over the set S.

Additionally, we need a *policy*, i.e. a rule stating what to do, given the knowledge of the current state of the environment. The policy is defined as a function from states to actions:

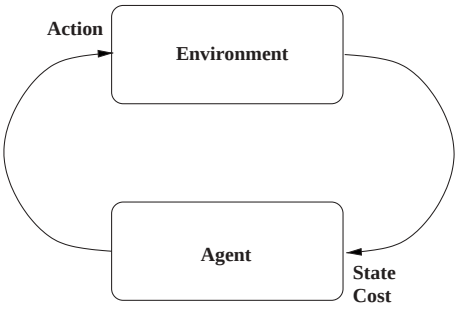

Fig. 4. An overview of the learning system.

$$\pi : S_t \rightarrow A_t, \tag{1}$$

where t refers to the discrete time step. The policy is *stationary* if there are no time dependents, i.e.

$$\pi : S \rightarrow A. \tag{2}$$

In this paper, we are only interested about stationary policies. The goal of the agent is to find the policy π_* that maximizes his expected discounted utility:

$$\begin{aligned} V_\pi(s) &= E_\pi[R|s_0 = s] \\ &= E_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right], \end{aligned} \tag{3}$$

where r_t is an immediate reward at time step t and γ is a discount factor. Moreover, the value for each state-action pair is:

$$\begin{aligned} Q_\pi(s,a) &= E_\pi[R|s_0 = s, a_0 = a] \\ &= r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_\pi(s'). \end{aligned} \tag{4}$$

Finding the optimal policy π_* can be seen as an optimization problem, which can be solved e.g. using dynamic programming algorithms.

3.2. Solving MDPs

Using dynamic programming requires solving the following equation for all states $s \in S$:

$$V_{\pi_*}(s) = \max_{a \in A(s)} Q_{\pi_*}(s,a). \tag{5}$$

These equations, *Bellman optimality equations*, form a basis for reinforcement learning algorithms. There are two basic methods for calculating the optimal policy, *policy iteration* and *value iteration*. In the policy iteration algorithm, the current policy is evaluated and then improved using greedy optimization based on the evaluation step. The value iteration algorithm is based on successive approximations of the value function and there is no need for repeated computation of the exact value function.

In both algorithms, the exact model of the environment should be known a priori. In many situations, however, we do not have the model available. Fortunately, it is possible to approximate the model from individual samples on-line. These methods are called temporal difference methods and can be divided to off-policy and on-policy methods based on whether they are using the same policy they are optimizing for learning or not. An example of on-policy methods is SARSA which has the update rule [23]:

$$\begin{aligned} Q_{t+1}(s_t,a_t) = (1-\alpha_t) Q_t(s_t,a_t) \\ + \alpha_t[r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1})], \end{aligned} \tag{6}$$

where the action selection in the state s_{t+1} occurs according to the current policy. An example of off-policy methods is Q-learning. Its update rule is [32]:

$$\begin{aligned} Q_{t+1}(s_t,a_t) = (1-\alpha_t) Q_t(s_t,a_t) \\ + \alpha_t[r_{t+1} + \gamma \max_{b \in A} Q_t(s_{t+1}, b)]. \end{aligned} \tag{7}$$

When an off-policy method like Q-learning is used, some other policy than the one being optimized can be used for exploring the state space. Therefore off-policy methods are more widely used in real applications.

4. Multiagent reinforcement learning

Until now, we have only discussed the case where there is only one agent in the environment. In this section we extend the theory of MDPs to the case of multiple decision makers in the same environment. At the end of the section, a number of solving and learning methods for this extended model are briefly discussed.

4.1. Markov games

With multiple agents in the environment, the fundamental problem of single-agent 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 Markov decision processes, 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 Markov decision process. Formally, we define a Markov game as follows:

Definition 4. 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, $r^i : S \times A^1 \times \ldots \times A^N \rightarrow \boldsymbol{R}$ is the reward function for the 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 policies are assumed to be stationary):

$$\pi^i : S \rightarrow A^i, \forall i \in \{1, N\}. \tag{8}$$

The expected discounted utility of agent i is the following:

$$\begin{aligned} V^i_{\pi^1,\ldots,\pi^N}(s) &= E_{\pi^1,\ldots,\pi^N}[R^i | s_0 = s] \\ &= E_{\pi^1,\ldots,\pi^N}\left[\sum_{t=0}^{\infty} \gamma^t r^i_{t+1} | s_0 = s\right], \end{aligned} \tag{9}$$

where r^i_t is the immediate reward at time step t for agent i and γ is a discount factor. Moreover, the value for each state-action pair is

$$\begin{aligned} &Q^i_{\pi^1,\ldots,\pi^N}(s, a^1, \ldots, a^N) \\ &= E_{\pi^1,\ldots,\pi^N}[R^i | s_0 = s, a^1_0 = a^1, \ldots, a^N_0 = a^N] \\ &= r^i(s, a^1, \ldots, a^N) \\ &\quad + \gamma \sum_{s'} p(s' | s, a^1, \ldots, a^N) V^i_{\pi^1,\ldots,\pi^N}(s'). \end{aligned} \tag{10}$$

Contrast to the single-agent MDP, 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. (8).

4.2. Solving markov games

In the case of multiagent reinforcement learning, it is not enough to maximize the expected utilities 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 5. 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:

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

It is noteworthy that Definition 5 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 build on Markov games with the Nash equilibrium concept as symmetric methods and to methods that utilize the Stackelberg equilibrium concept as asymmetric methods.

If the exact model, i.e. rewards and state transition probabilities, is known a priori, it is possible to solve the game using standard mathematical optimization methods. However, only a few special cases of Markov games can be solved with linear programming and, in general, more complex methods are needed.

4.3. Symmetric learning in markov games

A Markov game can be seen as a set of matrix games associated with each state $s \in S$. In these matrix games, payoffs for each player i are equal to function $Q^i_{\pi^1,\ldots,\pi^N}$. The learning methods utilize the fact that if the optimal value functions V^i are known, it is possible to obtain a solution of a Markov game by solving the matrix games associated to each state s. Of course, the value functions are not normally known and they must be estimated during the learning process.

As in the case of single agent reinforcement learning, Q-values defined in Eq. (10) 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 [11]:

$$\begin{aligned} Q^1_{t+1}(s_t, a^1_t, a^2_t) &= (1 - \alpha_t) Q^1_t(s_t, a^1_t, a^2_t) \\ &\quad + \alpha_t [r^1_{t+1} + \gamma \mathrm{Nash}\{Q^1_t(s_{t+1})\}], \end{aligned} \tag{11}$$

where $\mathrm{Nash}\{Q^1_t(s_{t+1})\}$ is a 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 [11]:

$$\begin{aligned} Q^2_{t+1}(s_t, a^1_t, a^2_t) &= (1 - \alpha_t) Q^2_t(s_t, a^1_t, a^2_t) \\ &\quad + \alpha_t [r^2_{t+1} + \gamma \mathrm{Nash}\{Q^2_t(s_{t+1})\}]. \end{aligned} \tag{12}$$

Note that it is guaranteed that every finite matrix game possesses at least one Nash equilibrium in mixed strategies. However, there is not necessarily Nash equilibrium point in pure strategies and therefore $\mathrm{Nash}\{\cdot\}$ in Eqs (11) and (12) returns the value of a mixed strategy equilibrium.

The major drawback of the generic Nash-operator is that the operator is not unique. Non-uniqueness induces two major problems:

1. It is not possible to provide general convergence proofs. This is a straight consequence of the non-uniqueness of the Nash operator; there are several Nash solutions and hence the value of the operator is not well-defined.
2. All agents should select the same equilibrium solution. This can be very considerable problem, since there can exists a vast number of different equilibria in a Markov game. It is possible to solve this problem, however, by using some refinements of the Nash equilibrium concept (reduce the space of the equilibrium solutions) or, in general, by allowing some messaging between agents. Learning methods that set additional requirements for the equilibria are proposed e.g. in [30], [31] and [29].

If the payoff structures of the matrix games associated with the states are zero-sum, it is possible to solve the games by using the MaxMin operator, which is unique. However, the zero-sum property could be too restrictive for many problem domains.

Moreover, calculating Nash equilibrium solution(s) of the matrix game at each step of the algorithm can be a very complex task. Greenwald and Hall overcome this problem by using correlated equilibrium concept in place of Nash equilibrium [10]. Correlated equilibria of a matrix game can be determined efficiently by using linear programming.

5. Asymmetric learning in markov games

In this section, we introduce a mathematical model for *asymmetric multiagent reinforcement learning*. We start by extending the Stackelberg equilibrium concept for Markov games and then proceed to actual solving and learning methods for these processes. Corresponding definitions and theorems for matrix games can be found in [3]. For brevity, all mathematics in this section is presented in the case of two agents, the agent one acting as the leader and the agent two as the follower. However, the extension to the case of arbitrary number of agents is quite straightforward and various agent hierarchies can be easily constructed.

The asymmetric learning model relaxes the problems of the symmetric model presented in the previous section by allowing communication between agents (leader's enforcements). However, if both agents accept their roles and keep a copy of the opponent's Q-function, there is no need for "real" communication between agents; both agents are capable of calculating an equilibrium solution alone.

5.1. Mathematical model

Let us begin with the following definition and assumption that lead to the Stackelberg equilibrium concept for Markov games:

Definition 6. Let Π^1 and Π^2 be (finite) policy spaces for the agents 1 and 2, respectively. Further, let $R(\pi^1) \in \Pi^2, \forall \pi^1 \in \Pi^1$ be the set of follower's (agent 2) response policies to leader's enforcement policies.

Assumption 1. The set of follower's (agent 2) response policies $R(\pi^1) \in \Pi^2, \forall \pi^1$ is a singleton, i.e. in every state $s \in S$ the follower's response is unique.

If Assumption 1 does not hold, some extra restrictions should be made. For example, the leader can be risk averse in the sense that he secures his possible losses against the choices of the follower or there can exist a social convention that stipulates the action selection method used by the follower.

Based on Definition 6 and Assumption 1, there is a mapping $T : \Pi^1 \rightarrow \Pi^2$ so that if $\pi^2 \in R(\pi^1)$ then $\pi^2 = T\pi^1$. In other words, the mapping T conducts the follower's best response to the leader's enforcement. Using the mapping T we can define the leader's Stackelberg policy π_S^1 as follows:

Definition 7. Let Π^1 be the finite policy space for agent 1 (leader). Then a Stackelberg equilibrium policy $\pi_S^1 \in \Pi^1$ satisfies the following inequality for all $\pi^1 \in \Pi^1$ in all $s \in S$:

$$V^1_{\pi^1, T\pi^1}(s) \leqslant V^1_{\pi^1_S, T\pi^1_S}(s)$$

Clearly, the follower's response is:

$$\pi^2_S = T\pi^1_S. \tag{13}$$

In the asymmetric learning model, the leader enforces his action first. Hence, the follower's policy is depended on this enforcement, i.e.

$$\pi^2 : S \times A^1 \rightarrow A^2. \tag{14}$$

The following theorem (adaptation for the Stackelberg equilibrium concept from [9]) leads to the actual procedure for determining a Stackelberg equilibrium policy for Markov games:

Theorem 1. The following two assertions are equivalent:

1. The pair (π^1_S, π^2_S) is a Stackelberg equilibrium point in the discounted Markov game with equilibrium payoffs $(V^1_{\pi^1_S, \pi^2_S}(s), V^2_{\pi^1_S, \pi^2_S}(s)), \forall s \in S$.

2. $\forall s \in S$, the pair $(\pi^1_S(s), \pi^2_S(s, \pi^1_S(s)))$ constitutes a Stackelberg equilibrium point in the static bimatrix game $(Q^1_{\pi^1_S,\pi^2_S}(s, a^1, a^2), Q^2_{\pi^1_S,\pi^2_S}(s, a^1, a^2)), a^1 \in A^1, a^2 \in A^2$.

proof: For the agent 1 in the arbitrary state $s \in S$, the state-action function $Q^1_{\pi^1_S,\pi^2_S}$ can be written as follows:

$$Q^1_{\pi^1_S,\pi^2_S}(s, a^1, a^2) = r^1(s, a^1, a^2) + \gamma \sum_{s'} p(s'|s, a^1, a^2) V^1_{\pi^1_S,\pi^2_S}(s').$$

Now, if Assertion 2 holds, it follows that

$$Q^1_{\pi^1_S,\pi^2_S}(s, a^1, \pi^2_S(s, a^1)) \leqslant V^1_{\pi^1_S,\pi^2_S}(s), \forall a^1 \in A^1$$

with equality for $\pi^1_S(s)$. Therefore it also holds that

$$V^1_{\pi^1,\pi^2_S}(s) \leqslant V^1_{\pi^1_S,\pi^2_S}(s), \forall \pi^1 \in \Pi^1, \forall s \in S.$$

Similarly, for the agent 2:

$$V^2_{\pi^1_S,\pi^2}(s) \leqslant V^2_{\pi^1_S,\pi^2_S}(s), \forall \pi^2 \in \Pi^2, \forall s \in S.$$

For the agent 1, if Assertion 1 holds, the policy π^1_S is a best response to the policy π^2_S. Then it follows, in the arbitrary state $s \in S$, for the MDP associated to the policy π^2_S:

$$\pi^1_S(s) = \arg\max_{a^1 \in A^1} [r^1(s, a^1, \pi^2_S(s, a^1)) + \gamma \sum_{s'} p(s'|s, a^1, \pi^2_S(s, a^1)) V^1_{\pi^1_S,\pi^2_S}(s')].$$

Clearly, this defines a best response to the opponent's action choice $\pi^2_S(s, a^1)$ in the bimatrix game associated to the state s. The proof for the agent 2 is symmetric.

5.2. Obtaining asymmetric solution of markov game

As stated in Theorem 1, a Markov game can be seen as a set of matrix games associated with each state $s \in S$. If the value functions of both the leader and the follower are known, we can obtain an asymmetric solution of the Markov game by solving the matrix game associated with each state s using the Stackelberg equilibrium solution concept. The following three stage protocol solves a Stackelberg equilibrium solution in a state $s \in S$:

1. Determination of the cooperation strategies $a^c = (a^{1c}, a^{2c})$ by finding the maximum element of the matrix game $Q^1_{\pi_1,\pi_2}$ in the state s:

$$\arg \max_{\substack{a^1 \in A^1 \\ a^2 \in A^2}} Q^1_{\pi^1,\pi^2}(s, a^1, a^2). \quad (15)$$

2. Determination of the leader's enforcement (and action, $a^1_S = g(s, a^c)$):

$$g(s, a^c) = \arg \min_{a^1 \in A^1} \|f(Q^2_{\pi^1,\pi^2}(s, a^1, a^2)), a^c\|. \quad (16)$$

3. Determination of the follower's response a^2_S:

$$a^2_S = \arg \max_{a^2 \in A^2} Q^2_{\pi^1,\pi^2}(s, g(s, a^c), a^2). \quad (17)$$

In the protocol, $\|a, a^c\|, a \in A^2$ is a distance measure, defined in the Q-value space of the leader, measuring the distance between the Q-value corresponding a particular action and the Q-value associated to the cooperation strategies (maximal possible payoff for the leader), i.e.

$$\|x, a^c\| = |Q^1_{\pi^1,\pi^2}(s, a^1, x) - Q^1_{\pi^1,\pi^2}(s, a^{1c}, a^{2c})|. \quad (18)$$

The function f is used to select actions by the player 2; e.g. in the case of of greedy action selection $f = \arg\max_{a^2 \in A^2}$. In practical implementations of the protocol, e.g. when the protocol is applied to action selection during learning, the minimization in step 2 can be replaced with the *softmin* function and the maximization in step 3 with the *softmax* function for ensuring the proper exploration of the state-action space.

5.3. Practical off-policy learning algorithm

Actual learning of the payoffs $Q^1_{\pi^1_S,\pi^2_S}$ and $Q^2_{\pi^1_S,\pi^2_S}$ can be done by using any suitable method from the field of reinforcement learning. In this paper we present the equations for asymmetric multiagent Q-learning. If the agent 1 is the leader and the agent 2 is the follower, update rules for the Q-values are as follows:

$$Q^1_{t+1}(s_t, a^1_t, a^2_t) = (1 - \alpha_t) Q^1_t(s_t, a^1_t, a^2_t) + \alpha_t [r^1_{t+1} + \gamma \max_{b \in A^1} Q^1_t(s_{t+1}, b, Tb)], \quad (19)$$

$$Q^2_{t+1}(s_t, a^1_t, a^2_t) = (1 - \alpha_t) Q^2_t(s_t, a^1_t, a^2_t) + \alpha_t [r^2_{t+1} + \gamma \max_{b \in A^2} Q^2_t(s_{t+1}, g(s_{t+1}, a^c_{t+1}), b)]. \quad (20)$$

Learning steps for the leader and the follower are shown in Algorithms 1 and 2, respectively. Note that only the leader needs a copy of the opponent agent. These algorithms do not dictate the actual action selection procedure; the only requirement is that every state-action tuple is visited infinitely often when teach-

ing is continued infinitely long. For example, it is possible to select actions by sampling from the uniform distribution. However, it is more efficient to calculate an asymmetric solution of the matrix game associated with the current state and differ from this solution with some small probability by using e.g. softmax action selection.

5.4. Convergence properties of the off-policy method

In this section we study the convergence of the above presented learning method. We build our study on two theorems published by Szepesvári and Littman in [27]. For brevity, all mathematics in this section is presented in the case of two agents.

We begin by introducing the following lemma and corollary originally proposed by Szepesvári and Littman in [27]:

Lemma 1. [Conditional Averaging Lemma] Let $0 \leqslant \alpha_t$ and ω_t be random variables. Assume that the following hold with probability 1: $E[\omega_t|\alpha_t \neq 0] = A$, $E[\omega_t^2] < B < \infty$, $\sum_{t=1}^{\infty} \alpha_t = \infty$ and $\sum_{t=1}^{\infty} \alpha_t^2 < C < \infty$ for some $B, C > 0$. Then, the process

$$Q_{t+1} = (1 - \alpha_t)Q_t + \alpha_t \omega_t$$

converges to A with probability 1.

Corollary 1. If X is an arbitrary set and $P_t : \boldsymbol{Q}(\mathsf{X}) \rightarrow \boldsymbol{Q}(\mathsf{X})$ is an operator that performs a mapping from the Q-value space $\boldsymbol{Q}$ to itself, the following process converges to the fixed point Q^* ($x \in \mathsf{X}$):

$$Q_{t+1}(x) = (1 - f_t(x))Q_t(x) + f_t(x)[P_t Q_t](x)$$

if the following assumptions hold:

1. The following process converges to Q^* with probability 1:
$$Q_{t+1}(x) = (1 - f_t(x))Q_t(x) + f_t(x)[P_t Q^*](x)$$
2. There exists a number $0 < a < 1$ and a sequence $\lambda_t \geqslant 0$ converging to zero with probability 1 such that $\|P_t Q - P_t Q^*\| \leqslant a\|Q - Q^*\| + \lambda_t$ holds for all $Q \in \boldsymbol{Q}$.
3. $0 \leqslant f_t(x) \leqslant 1$, $t \geqslant 0$ and $\sum_{t=1}^{n} f_t(x)$ converges to infinity uniformly in x as $n \rightarrow \infty$.

Note that in [27], the expectations in Lemma 1 are conditioned with the history space F_t (increasing sequence of σ-fields). However, the properties concerning these σ-fields are satisfied trivially with asynchronous reinforcement learning methods and hence, for brevity, we neglect these conditions in our inspections.

The crucial point is to note that both the leader's and the follower's learning processes are in the form of the process in Corollary 1. In this work we identify X by the set of state-action tuples (s, a^1, a^2). Additionally we set that:

$$f_t(s, a^1, a^2) = \begin{cases} \alpha_t(s_t, a_t^1, a_t^2) & \text{if}(s, a^1, a^2) \\ & = (s_t, a_t^1, a_t^2) \\ 0 & \text{otherwise} \end{cases}$$

For brevity, we denote $\alpha_t(s_t, a_t^1, a_t^2) = \alpha_t$. Further, if usual conditions of the stochastic approximation theory for the learning rate parameter α_t hold, we immediately see that condition 3 in Corollary 1 holds. We proceed by proving condition 1.

We start the proof by stipulating the operator P_t for the leader and for the follower as follows:

$$P_t Q^1 = r_{t+1}^1 + \gamma \max_{b \in A^1} Q^1(s_{t+1}, b, Tb)$$

$$P_t Q^2 = r_{t+1}^2 + \gamma \max_{b \in A^2} Q^2(s_{t+1}, g(s_{t+1}, a_{t+1}^c), b)$$

From condition 1 in Corollary 1, we get the following two processes:

$$\begin{aligned} Q_{t+1}^1(s_t, a_t^1, a_t^2) &= (1 - \alpha_t)Q_t^1(s_t, a_t^1, a_t^2) \\ &\quad + \alpha_t[P_t Q_S^1](s_t, a_t^1, a_t^2) \\ Q_{t+1}^2(s_t, a_t^1, a_t^2) &= (1 - \alpha_t)Q_t^2(s_t, a_t^1, a_t^2) \\ &\quad + \alpha_t[P_t Q_S^2](s_t, a_t^1, a_t^2). \end{aligned}$$

Now we can see that these two processes are in the one-dimensional form presented in Lemma 1. Thus, to prove condition 1, it is sufficient to show that:

$$Q_S^1 = E[P_t Q_S^1]$$
$$Q_S^2 = E[P_t Q_S^2],$$

where the expectation operator E is defined over the state space S. For the leader, we get as follows:

$$E[P_t Q_S^1] = r_{t+1}^1 + \gamma E[\max_{b \in A^1} Q_S^1(s_{t+1}, b, Tb)].$$

Moreover, it applies that

$$\begin{aligned} Q_S^1 &= r_{t+1}^1 + \gamma \sum_{s_{t+1} \in S} p(s_{t+1}|s_t, a_t^1, a_t^2) V_S^1 \\ &= r_{t+1}^1 \\ &+ \gamma \sum_{s_{t+1} \in S} p(s_{t+1}|s_t, a_t^1, a_t^2) \max_{b \in A^1} Q_S^1(s_{t+1}, b, Tb) \\ &= r_{t+1}^1 + \gamma E[\max_{b \in A^1} Q_S^1(s_{t+1}, b, Tb)]. \end{aligned}$$

1. In the state s_t, select actions a_t^1 and a_t^2.
2. Observe a new state s_{t+1} and rewards r_{t+1}^1, r_{t+1}^2.
3. Solve the matrix game associated with the state s_{t+1} by using Stackelberg equilibrium concept. Let us denote the values corresponding these solutions as $\mathrm{Val}^1(s_{t+1})$ and $\mathrm{Val}^2(s_{t+1})$ for the leader and the follower, respectively.
4. Update Q-values for both agents by using the equations:
$$Q_{t+1}^1(s_t, a_t^1, a_t^2) = (1-\alpha_t)Q_t^1(s_t, a_t^1, a_t^2) + \alpha_t[r_{t+1}^1 + \gamma \mathrm{Val}^1(s_{t+1})]$$
and
$$Q_{t+1}^2(s_t, a_t^1, a_t^2) = (1-\alpha_t)Q_t^2(s_t, a_t^1, a_t^2) + \alpha_t[r_{t+1}^2 + \gamma \mathrm{Val}^2(s_{t+1})].$$

Algorithm 1. Single learning step for the leader. Q-learning type off-policy method.

1. In the state s_t, select actions a_t^1 and a_t^2.
2. Observe a new state s_{t+1} and a reward r_{t+1}^2.
3. Obtain the action enforcement from the leader in the state s_{t+1}.
4. Maximize the utility with respect to the enforcement. Let us denote the value corresponding the solution $\mathrm{Val}^2(s_{t+1})$.
5. Update Q-value table by using the equation:
$$Q_{t+1}^2(s_t, a_t^1, a_t^2) = (1-\alpha_t)Q_t^2(s_t, a_t^1, a_t^2) + \alpha_t[r_{t+1}^2 + \gamma \mathrm{Val}^2(s_{t+1})].$$

Algorithm 2. Single learning step for the follower. Q-learning type off-policy method.

Hence, we conclude that $Q_S^1 = E[P_t Q_S^1]$. Similarly, for the follower we get:

$$E[P_t Q_S^2] = r_{t+1}^2 + \gamma E[\max_{b \in A^2} Q_S^2(s_{t+1}, g(s_{t+1}, a_{t+1}^c), b)]$$

and

$$Q_S^2 = r_{t+1}^2 + \gamma \sum_{s_{t+1} \in S} p(s_{t+1}|s_t, a_t^1, a_t^2) V_S^2$$
$$= r_{t+1}^2 + \gamma E[\max_{b \in A^2} Q_S^2(s_{t+1}, g(s_{t+1}, a_{t+1}^c), b)],$$

hence $Q_S^2 = E[P_t Q_S^2]$. The proof of condition 1 in Corollary 1 is complete.

To study condition 2 in Corollary 1, we study a stricter condition for both agents, i.e.

$$\|P_t Q - P_t \hat{Q}\| \leqslant a\|Q - \hat{Q}\|,$$

for some constant $0 < a < 1$ and for all $Q, \hat{Q} \in \boldsymbol{Q}$. This condition assures that the operator P_t is a real contraction operator. We begin by stipulating the norm $\|\cdot\|$:

$$\|\cdot\| = \max_{s \in S} \max_{a^1 \in A^1} \max_{a^2 \in A^2} |f(s, a^1, a^2)|,$$

where f is an arbitrary function of $s \in S$, $a^1 \in A^1$ and $a^2 \in A^2$. For the lhs of condition 2, we get the following:

$$\gamma |\max_{b \in A^1} Q(s_{t+1}, b, Tb) - \max_{b \in A^1} \hat{Q}(s_{t+1}, b, Tb)|.$$

The rhs of condition 2 is:

$$a \max_{s \in S} \max_{b \in A^1} \max_{c \in A^2} |Q(s, b, c) - \hat{Q}(s, b, c)|.$$

Now, by setting $a = \gamma$ we get the following inequality:

$$|\max_{b \in A^1} Q(s_{t+1}, b, Tb) - \max_{b \in A^1} \hat{Q}(s_{t+1}, b, Tb)|$$
$$\leqslant \max_{s \in S} \max_{b \in A^1} \max_{c \in A^2} |Q(s, b, c) - \hat{Q}(s, b, c)|.$$

If the inequality holds when we fix its rhs to the state s_{t+1}, it also holds in the maximizing state. Therefore it is sufficient to show that:

$$|\max_{b \in A^1} Q(s_{t+1}, b, Tb) - \max_{b \in A^1} \hat{Q}(s_{t+1}, b, Tb)|$$
$$\leqslant \max_{b \in A^1} \max_{c \in A^2} |Q(s_{t+1}, b, c) - \hat{Q}(s_{t+1}, b, c)|.$$

It is easy to show that this inequality holds for team Markov games in which the asymmetric solution is the global maximum of the game and the games with zero-sum payoff structure. However, because the follower's response $Ta, a \in A^1$ is depended on his Q-values and changes during learning, the inequality does not hold for general-sum Markov games. In each learning step, the leader effectively has only the fraction of Q-values available, i.e. Q-values indexed with a pair $(a, Ta), a \in A^1$. Therefore we write a weaker version of the inequality:

$$|\max_{b \in A^1} Q(s_{t+1}, b, Tb) - \max_{b \in A^1} \hat{Q}(s_{t+1}, b, Tb)|$$
$$\leqslant \max_{b \in A^1} |Q(s_{t+1}, b, Tb) - \hat{Q}(s_{t+1}, b, Tb)|,$$

which is straightforward to show to be true for any response Tb. The meaning of this inequality is that the

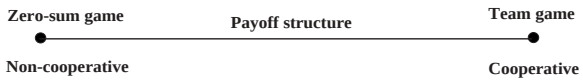

Fig. 5. Payoff structure vs. nature of games.

process moves the available Q-values closer to each other, in particular closer to the values associated with the asymmetric solution. If the follower's responses change between two updates in some state, it can cause a large difference of the leader's value in that state. However, if the follower is not changing its responses constantly, this would not cause convergence problems. In the case of the symmetric learning model, the response is not even unique (as there can exist multiple Nash equilibria) and thus there are many different distributions from where the responses are sampled. Similar deduction also applies for the follower.

The stronger convergence properties are straight consequence of the communication between agents. The value of the Stackelberg solution is unique for the leader and therefore there is an incentive for the leader always use the same Stackelberg strategy and this implies that the value of the equilibrium is always unique for both agents. For general-sum games, it is still not possible to give general convergence proof.

5.5. Applicability of the asymmetric learning model

The proposed asymmetric multiagent reinforcement learning model itself does not set any restrictions on the payoff structure of the game. However, the ordering among decision makers and the asymmetric nature of the proposed model lead to the immediate question: in what situations is the asymmetric learning model applicable? Another, more philosophical question is: do the decision makers agree on their roles in the decision process?

In Fig. 5, an illustration of how the nature of the mathematical game depends on its payoff structure is presented. In a team game, the same utility function is shared by both agents and hence the correlation between agents' payoff values is at maximum. On the other hand, in zero-sum games the payoffs are complements of each other and there is a maximal negative correlation between payoff values. In general-sum games, the objectives of the agents are partially conflicting and therefore these games lie between the extreme cases presented above in the line of Fig. 5.

In team games, there can be multiple different Nash equilibrium solutions and therefore some equilibrium selection method is needed. One possible solution is to fix the ordering of the decision makers and to apply asymmetric solution concept. In this case, it is also easy to agree on the roles of the decision makers (asymmetric solution is the global maximum).

In non-cooperative domains, the use of the asymmetric solution is only reasonable if the outcome of the game for the leader depends on his action choice. The reason for this is that, in a sequential decision process, the follower knows the leader's action choice and therefore has a vantage over the leader. At the end of this paper we present a pricing application where the leader (supplier) and the followers (brokers) have partially conflicting objectives and the leader can affect greatly to the outcome by his enforcements. Note, that in zero-sum games, the asymmetric solution is simply the alternating-turn version of the well-known MaxMin solution concept discussed in [19].

In learning problems, the payoff structure is not known to the agents a priori and it can vary in different states. Therefore, the selection of the equilibrium concept depends heavily on the problem instance. For example, in the two-level pricing application presented at the end of this paper, the asymmetric equilibrium concept is a natural choice between the supplier and the brokers (the supplier charges the brokers and the brokers in turn charge their customers). At the same time, the relationship between the two brokers is symmetric.

6. Space and computational requirements

The learning model of two agents was discussed in the previous section. The extension to the case of three or more agents is fairly straightforward and can be roughly divided into three categories:

1. **One leader and two or more followers**. In this category, the leader is enforcing his action to each follower. The followers are playing among themselves by using some equilibrium concept.
2. **Two or more leaders and two or more followers**. Leaders and followers are playing among themselves by using some equilibrium concept. The leaders are enforcing their action choices to the specific followers or all of them.
3. **Many levels of hierarchy.** The leader on the highest level enforces his action to the second level leader and the second level leader enforces, in turn, his action to the followers. It is possible that on all hierarchy levels there are more than one agents and they are playing among themselves by using some equilibrium concept.

Next we will inspect the space and computational requirements of asymmetric multiagent reinforcement learning. It is assumed that action selections of all agents in the system influence the payoff functions of all agents, i.e. the influence diagram of the system is a fully connected graph. It is noteworthy that these space requirements are also eligible for the comparison of computational requirements; one state-actions tuple is also a parameter in tabular implementations of these algorithms.

Let us denote the size of the state space as $|S|$ and assume that the action spaces are state independent. Furhermore, let $|A^i|$ be the size of action space for agent i. For brevity, we make an simplifying assumption that all action spaces are equal size, i.e. $|A^i| = |A|, \forall i$. Then we obtain the following space requirement:

$$S_{\text{Stackelberg}} = |S||A|^{N^{\text{Tot}}}, \tag{21}$$

where N^{Tot} is the total number of learning agents. This space requirement is the same as in the case of Nash equilibrium. Note that the space requirement corresponds the number of parameters required to store Q-values in each agent of the system. In the symmetric learning model, all agents must have a model of their opponents whereas in the case of asymmetric learning model only the leaders should have models of their followers.

7. Example problems

In this section, we solve two simple example problems by using both symmetric and asymmetric reinforcement learning models discussed earlier in this paper. These problems are variations from the commonly-used grid world problem, which is used for testing single-agent and multiagent reinforcement learning algorithms in many works, e.g. in [25], [21], [10] and [20]. Our test cases are the same as in [11], where the problems were solved using a tabular version of the symmetric multiagent reinforcement learning algorithm. Moreover, we test the proposed asymmetric learning model with a larger pricing application.

7.1. Grid world example

In both problems we have a grid world containing nine cells and two competing agents (Fig. 6). The agents start from the lower corners 1 and 2, respectively, and on each round they can move to the adjacent cells (4-neighborhood). In problem 1, there are two

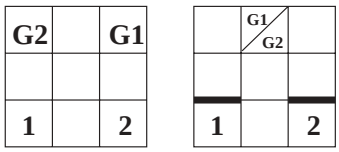

Fig. 6. The game boards used in the example problems. In both cases, agents are initially located in the cells marked with numbers **1** and **2**. Goal cells are marked with **G1**- and **G2**-symbols. In problem 2, there are barriers in the start cells (marked with thick lines).

distinct goal positions and in the second problem, the goal cell is the same for both agents. The agents get large positive payoffs when they reach the right goal positions. In the symmetric learning model both agents get the small negative payoffs when they try to move to the same position and agents are returned back to their original positions. In the asymmetric learning model, only the agent 1 (leader) gets the negative payoff and is thus trying to avoid the collision by its enforcements. Hence, the ultimate goal of the agents is to reach the goal cells using as few moves as possible.

In problem 2, there are two barriers in the cells 1 and 2. When the agent tries to move upward from the start position, it gets through with probability 0.5 and, respectively with probability 0.5 barrier blocks the movement and the agent remains in the start position. This extension can be modeled as a stochastic state transition in the corresponding Markov game.

We characterize the problem with the following Markov game:

- A state in this problem is a pair $s = (p_1, p_2)$, i.e. the positions of the agents. Hence, the state space of this example consists of $9 * 9 = 81$ states.
- Both agents get positive payoffs of 0.9 when they find the right goal cells.
- The action set for both agents is $A^i = \{\text{Left}, \text{Right}, \text{Up}, \text{Down}\}, i = 1, 2$. Both agents are restricted to stay on the game board.
- The discount factor γ is 0.99.
- Both agents select their actions using the softmax action selection method.
- In the asymmetric model the agent 1 is acting as the leader.
- In the symmetric model, if the agents collide both agents get small negative rewards of -0.1 and in the asymmetric model only the leader gets this negative reward.

We solve the problem by using Q-learning like off-policy methods presented earlier in this paper with symmetric and asymmetric learning models. The learning rate parameter α and the temperature parameter in

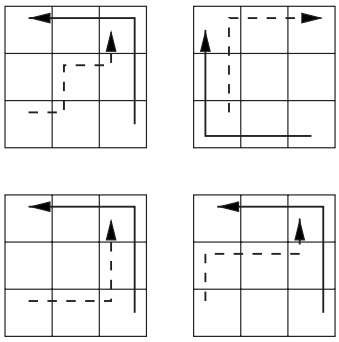

Fig. 7. Few optimal paths in problem 1.

Table 1
Averaged payoff values from 50 test runs in the form (agent 1, agent 2)

	problem 1	problem 2
symmetric	0.87,0.87	0.51,0.79
asymmetric	0.87,0.87	0.37,0.88

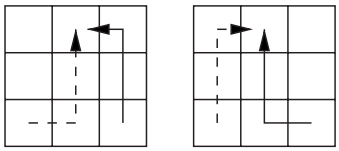

Fig. 8. Optimal paths in problem 2.

the softmax action selection are modified linearly with time. Note that the learning rate decaying scheme used in the simulation runs does not obey the conditions defined in stochastic approximation theory. However, in real problems, the learning rate decaying scheme that satisfies these theoretical conditions induces often very slowly convergence and thus this kind of scheme is seldom used in real applications and empirical research.

When one of the agents reaches the goal position, agents are moved back to their initial positions and the test run is restarted with a new episode. The maximum length of the episode is restricted to 10 moves. We repeated test runs 50 times in every test case. In Fig. 7, there are few equilibrium paths generated by the symmetric and the asymmetric learning models in problem 1. Respectively, in Fig. 8, there are equilibrium paths in problem 2.

The averaged payoff values calculated from test runs are shown in Table 1. In problem 1, both learning models performed equally and found optimal paths. In problem 2, the symmetric model found both optimal paths illustrated in Fig. 8 while the asymmetric model found only the rightmost path. The reason for this phenomenon is that only the leader gets negative feedback and this motivates him to move upward in the start state.

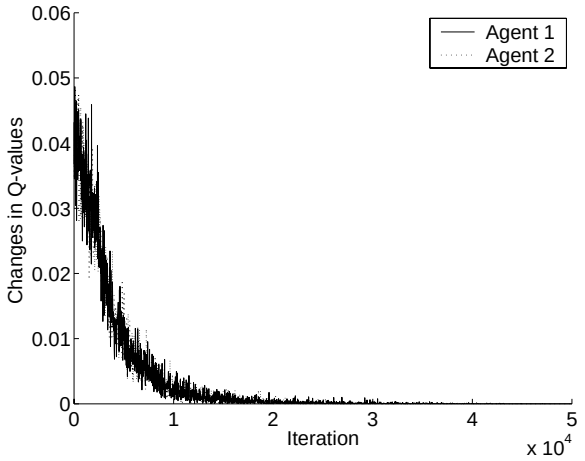

Fig. 9. The convergence of the symmetric multiagent reinforcement learning algorithm in problem 1.

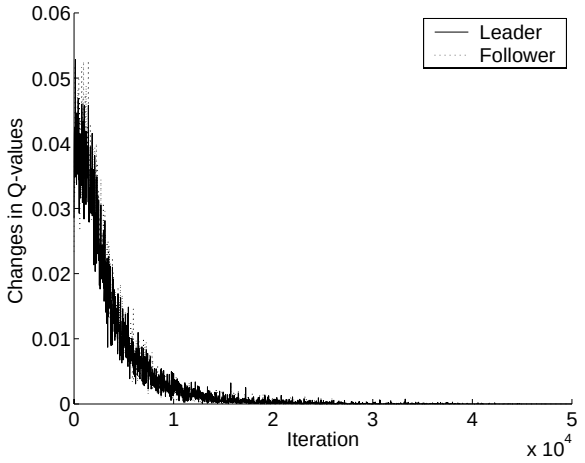

Fig. 10. The convergence of the asymmetric multiagent reinforcement learning algorithm in problem 1.

The convergence curves of the learning models are shown in Figs 9–12. In these curves one point is an Euclidean distance between two vectors containing Q-values from consecutive iterations of the algorithm. In all cases, both agents learned at the same pace. Overall, the learning in the asymmetric case is slightly faster than in the symmetric case. Note that in problem 2, the variance of the convergence speed is higher than in problem 1 due to the stochastic nature of the problem.

7.2. Two-layer pricing model

In this application, we extend the pricing model originally presented by Tesauro and Kephart in [28] to handle two-layer agent hierarchies. In this model, there

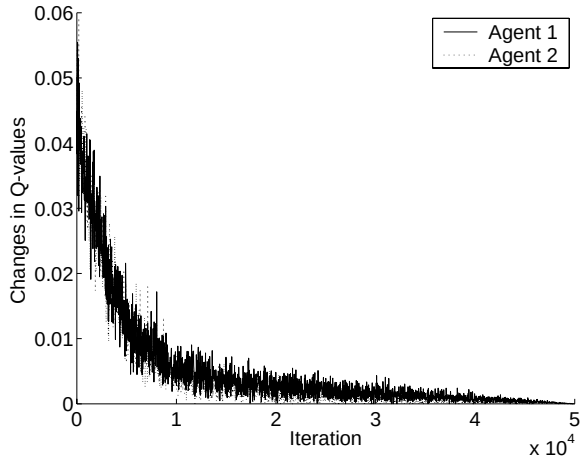

Fig. 11. The convergence of the symmetric multiagent reinforcement learning algorithm in problem 2.

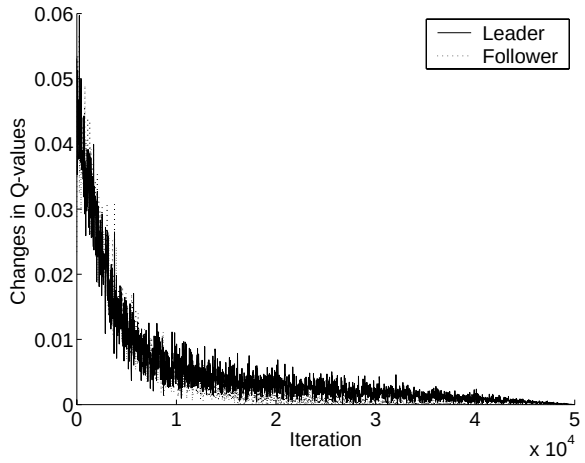

Fig. 12. The convergence of the asymmetric multiagent reinforcement learning algorithm in problem 2.

are two competing agents (brokers) that sell identical products and compete against each other on the basis of price. Additionally, there is a supplier that sell products to both brokers. At each time step, one of the brokers decides its new price based on the opponent's (other broker's) current price and the price set by the supplier. The supplier, in its turn, decides its action based on the asymmetric solution concept. After the prices has been set, the customer either buys a product from the seller or decides not to buy the product at all. After the customer's decision, the brokers get their profits according to the immediate reward functions presented in Eqs (26) and (27). Utility values for the supplier are shown in Eqs (28) and (29) in the case where the brokers 1 and 2 are charged, respectively.

$$u^1(p^1,p^2,s;l)=\begin{cases}p^1-s \text{ if } p^1\leqslant p^2 \text{ and}\\ \qquad s<lp^1\\ 0 \qquad \text{otherwise}\end{cases} \tag{22}$$

$$u^2(p^1,p^2,s;l)=\begin{cases}p^2-s \text{ if } p^1> p^2 \text{ and}\\ \qquad s<lp^2\\ 0 \qquad \text{otherwise}\end{cases} \tag{23}$$

$$u^{s1}(p^1,p^2,s;l)=\begin{cases}s-c \text{ if } p^1\leqslant p^2 \text{ and}\\ \qquad s<lp^1\\ 0 \qquad \text{otherwise}\end{cases} \tag{24}$$

$$u^{s2}(p^1,p^2,s;l)=\begin{cases}s-c \text{ if } p^1> p^2 \text{ and}\\ \qquad s<lp^2\\ 0 \qquad \text{otherwise}\end{cases} \tag{25}$$

In Eqs (26)–(29), 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. c is the fixed producing 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 the same value for each broker.

The meaning of the above proposed utility functions is that the customer purchases the product from the broker with the lowest price. If the supplier is charging too much from the broker (expected profit for the broker is too low), the broker does not buy the product from the supplier and the utility drops to zero for both the supplier and the broker.

In this study, 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. Particularly, the use of the reinforcement learning helps the agents to avoid "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 be very low. Furthermore, 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 a 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. (27). Further, the supplier also keeps its pricing strategy fixed with the broker 2. Figure 13 illustrates this 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. As the broker 2 uses the fixed strategy, there is no need for the game between

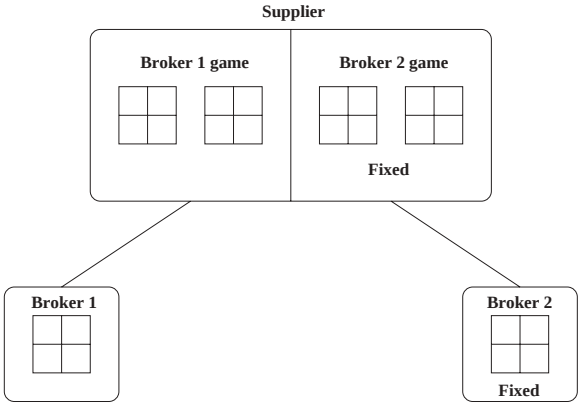

Fig. 13. Agent hierarchy in the pricing application.

the brokers. Hence, the update rules for the supplier and the broker 1 are as follows:

$$Q^{s1}_{t+1}(p^2_t, s_t, p^1_t) = (1-\alpha_t)Q^{s1}_t(p^2_t, s_t, p^1_t) + \alpha_t[u^{s1}(p^1_t, p^2_t, s_t; l) + \gamma \max_{b\in P} Q^{s1}_t(p^2_{t+1}, b, Tb)] \quad (26)$$

and

$$Q^{1}_{t+1}(p^2_t, s_t, p^1_t) = (1-\alpha_t)Q^{1}_t(p^2_t, s_t, p^1_t) + \alpha_t[u^{1}(p^1_t, p^2_t, s_t; l) + \gamma \max_{b\in P} Q^{1}_t(p^2_{t+1}, g(p^2_{t+1}, a^c_{t+1}), b)], \quad (27)$$

where P is the set of prices and T is the operator performing the response of the broker 1. 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 Eqs (26) and (28).

In our test runs, all prices lie in the unit interval and the number of different pricing options is 25 for all agents. The parameter l has a value of 0.8 and the producing cost for the supplier is $c = 0.2$ per product. Additionally, the maximum price for the supplier is 0.8. During training each state-action tuple was visited 1000 times. In the testing phase, the initial prices were selected randomly and one test run consisted of 100 pricing decisions per broker. In Fig. 14, 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 brokers learn a pricing strategy that leads to a notable 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 it has

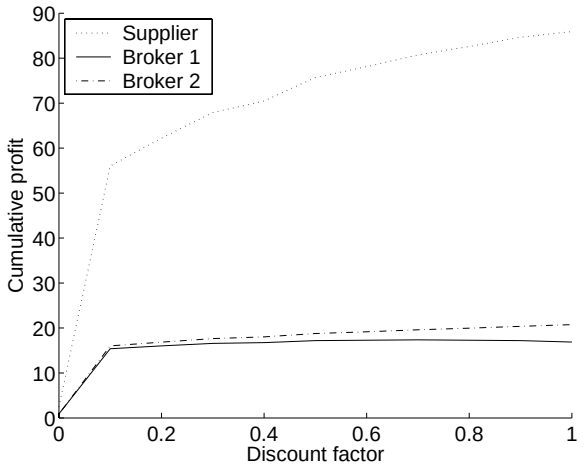

Fig. 14. Averaged profits in the two-layer pricing model. All data points are averages of 1000 test runs each containing 100 pricing decisions for both agents. The maximal possible profit for the brokers is 100 and for the supplier 200.

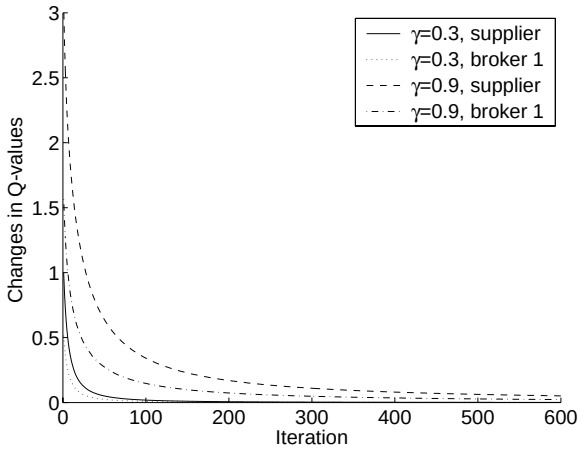

Fig. 15. Convergence of the Q-values in the two-layer pricing application.

slightly lower profits than the static broker has. However, the cumulative profits are much higher also for the broker 1 than in the myopic case.

The convergence of the agents' Q-value tables is illustrated in Fig. 15, where the Euclidean distance between Q-value vectors from consecutive training rounds is plotted against the round number. Two different cases with discount factors 0.3 and 0.9 are plotted for the broker 1 and the supplier. It can be seen that the algorithm converged very fast in every case. Convergence speed with high γ values, however, is much slower than with low values.

8. Conclusions and future research

A novel model for asymmetric multiagent reinforcement learning is presented in this paper. The paper extends the Stackelberg equilibrium concept to Markov games and, based on this concept, a learning model and a practical learning method are constructed. The proposed method has stronger convergence properties than the symmetric multiagent reinforcement learning methods have and the evaluation of the asymmetric solutions during the learning process demands less computation than in the symmetric case. In addition, the proposed method was tested with two example applications. In both cases, the method converged and showed good performance.

The growth of the inherent dimensionality of the problem is much bigger concern with multiagent reinforcement learning than with single-agent reinforcement learning. Thus, in future research, efficient numerical learning methods, both value function based and direct policy gradient methods, will be presented for asymmetric multiagent reinforcement learning.

References

[1] L. Baird and A. Moore, Gradient descent for general reinforcement learning, in: *Advances in Neural Information Processing Systems* (Vol. 11), Cambridge, MA, 1999, MIT Press.

[2] J.F. Bard, *Practical Bilevel Optimization – Algorithms and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[3] T. Basar and G.J. Olsder, *Dynamic Noncooperative Game Theory*, volume 160 of Mathematics in Science and Engineering. Academic Press Inc. (London) Ltd., London, UK, 1982.

[4] M. Bowling and M.M. Veloso, Multiagent learning using a variable learning rate, *Artificial Intelligence* **136**(2) (2002).

[5] M. Bowling and M.M. Veloso, *Scalable learning in stochastic games*, in: Proceedings of the AAAI-2002 Workshop on Game Theoretic and Decision Theoretic Agents, Edmonton, Canada, 2002.

[6] C. Claus and C. Boutilier, *The dynamics of reinforcement learning in cooperative multiagent systems*, in Proceedings of the Fifteenth National Conference of Artificial Intelligence (AAAI-98)/Proceedings of the Tenth Conference of Innovative Applications of the Artificial Intelligence (IAAI-98), Madison, WI, 1998. AAAI Press.

[7] V. Conitzer and T. Sandholm, *AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents*, in Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003. Morgan Kaufmann Publishers.

[8] V. Conitzer and T. Sandholm, *Complexity results about Nash equilibria*, in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-2003), Acapulco, Mexico, 2003. AAAI Press.

[9] J.A. Filar and K. Vrieze, *Competitive Markov Decision Processes*, Springer-Verlag, New York, NY, 1997.

[10] A. Greenwald and K. Hall, *Correlated-Q learning*, in Proceedings of the AAAI-2002 Spring Symposium Workshop on Collaborative Learning Agents, Stanford, CA, 2002. AAAI Press.

[11] J. Hu and M.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.

[12] L.P. Kaelbling, M.L. Littman and A.W. Moore, Reinforcement learning: A survey, *Journal of Artificial Intelligence Research* **4** (1996).

[13] S. Kapetanakis and Kudenko, *Improving on the reinforcement learning of coordination in cooperative multi-agent systems*, in Proceedings of the Second Symposium on Adaptive Agents and Multi-agent Systems (AAMAS-02), London, UK, 2002. Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

[14] S. Kapetanakis and D. Kudenko, *Reinforcement learning of coordination in cooperative multi-agent systems*, in Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02), Edmonton, Alberta, Canada, 2002. AAAI Press.

[15] S. Kapetanakis, D. Kudenko and M. Strens, *Learning to coordinate using commitment sequences in cooperative multiagent-systems*, in Proceedings of the Third Symposium on Adaptive Agents and Multi-agent Systems (AAMAS-03), Aberystwyth, UK, 2003. Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

[16] V.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.

[17] V.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.

[18] V.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.

[19] M.L. Littman, *Markov games as a framework for multi-agent reinforcement learning*, in Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, NJ, 1994. Morgan Kaufmann Publishers.

[20] M.L. Littman, *Friend-or-Foe Q-learning in general-sum games*, in Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williamstown, MA, 2001. Morgan Kaufmann Publishers.

[21] T.M. Mitchell, *Machine Learning*, McGraw-Hill, New York, NY, 1997.

[22] L. Peshkin, K.-E. Kim, N. Meuleau and L.P. Kaelbling, *Learning to cooperate via policy-search*, in Proceedings of the Sixteenth Conference on Uncertainty in Artifical Intelligence (UAI-2000), Stanford, CA, 2000. Morgan Kaufmann Publishers.

[23] G.A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*, Technical Report CUED/F-INFENG/TR166, Cambridge University, Engineering Department, 1994.

[24] S. Singh, M. Kearns and Y. Mansour, *Nash convergence of gradient dynamics in general-sum games*, in Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000), Stanford, CA, 2000. Morgan Kaufmann Publishers.

[25] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[26] R.S. Sutton, D. McAllester, S. Singh and Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: *Advances in Neural Information Processing Systems*, (vol. 12), Cambridge, MA, 2000. MIT Press.

[27] C. Szepesvári and M.L. Littman, A unified analysis of value-function-based reinforcement-learning algorithms, *Neural Computation* **11**(8) (1999).

[28] G. Tesauro and J.O. Kephart, *Pricing in agent economies using multi-agent Q-learning*, in Proceedings of Workshop on Game Theoretic and Decision Theoretic Agents, London, England, 1999.

[29] K. Verbeeck, A. Nowé, T. Lenaerts and J. Parent, *Learning to reach the Pareto optimal Nash equilibrium as a team*, in Proceedings of the Fifteenth Australian Joint Conference on Artificial Intelligence (AI 2002), Canberra, Australia, 2002. Springer-Verlag.

[30] X. Wang and T. Sandholm, *Reinforcement learning to play an optimal Nash equilibrium in team Markov games*, in Advances in Neural Information Processing Systems, volume 15, Cambridge, MA, 2002. MIT Press.

[31] X. Wang and T. Sandholm, Learning near-Pareto-optimal conventions in polynomial time, in: *Advances in Neural Information Processing Systems*, (Vol. 16), Cambridge, MA, 2003. MIT Press.

[32] C.J. Watkins, *Learning from Delayed Rewards*, PhD thesis, Cambridge University, Cambridge, England, 1989.

[33] D.H. Wolpert, S. Kirshner, C.J. Merz and K. Tumer, *Adaptivity in agent-based routing for data networks*, in Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona, Spain, 2000. ACM Press.

[34] D.H. Wolpert and K. Tumer, An introduction to collective intelligence, Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 2000.

[35] D.H. Wolpert, K. Tumer and J. Frank, Using collective intelligence to route Internet traffic, in: *Advances in Neural Information Processing Systems*, (vol. 11), Cambridge, MA, 1999. MIT Press.