Multi-Agent Learning in Conflicting Multi-level Games with Incomplete Information

Maarten Peeters[*] and **Katja Verbeeck** and **Ann Nowé**
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels
Belgium
mjpeeter@vub.ac.be, kaverbee@vub.ac.be, asnowe@info.vub.ac.be
WWW home page: http://como.vub.ac.be

Abstract

Coordination to some equilibrium point is an interesting problem in multi-agent reinforcement learning. In common interest single stage settings this problem has been studied profoundly and efficient solution techniques have been found. Also for particular multi-stage games some experiments show good results. However, for a large scale of problems the agents do not share a common pay-off function. Again, for single stage problems, a solution technique exists that finds a fair solution for all agents. In this paper we report on a technique that is based on learning automata theory and periodical policies. Letting pseudo-independent agents play periodical policies enables them to behave socially in pure conflicting multi-stage games as defined by E. Billard (Billard & Lakshmivarahan 1999; Zhou, Billard, & Lakshmivarahan 1999). We experimented with this technique on games where simple learning automata have the tendency not to cooperate or to show oscillating behavior resulting in a suboptimal pay-off. Simulation results illustrate that our technique overcomes these problems and our agents find a fair solution for both agents.

Introduction

Analysis of the collective behavior of agents in a distributed multi-agent setting has received quite a lot of attention lately. To be more precise, coordination is studied profoundly because coordination enables agents to converge to the desired equilibrium points. These equilibrium points might be points where all the agents receive their highest possible pay-off or they might be points where only one agent receives a high pay-off and the rest of the agents perform suboptimal. Alternating between these equilibria might be interesting to improve the average pay-off of the worst performing agent. Because we assume the agents are distributed we are looking for a distributed approach.

For single-stage games different techniques exist that guarantee the agents to reach a global optimum in common interest games without central control. We distinguish two different categories of solutions: the joint-action learners (Boutilier 1996; Hu & Wellman 1998; Mukherjee & Sen 2001) and techniques for independent agents (Kapetanakis, Kudenko, & Strens 2003; Lauer & Riedmiller 2000; Verbeeck, Nowé, & Tuyls 2003). The difference between those two categories is created by the amount and type of information the agents receive. Joint-action learners get informed not only about their own reward but also about the actions taken by other agents in the environment or their pay-offs. With this extra information the agents are able to construct a model of the opponents and their strategies. The independent agents only get informed about their personal pay-off. Also for multi-stage games, algorithms have been designed that reach an equilibrium (Littman 1994; Wang & Sandholm 2002; Brafman & Tennenholtz 2003). The technique we report on in this paper is for independent agents and is an extension of the periodical policies for single-stage games (Nowé, Parent, & Verbeeck 2001). The basic idea of this technique is that the best performing agent excludes his action and gives his opponent a chance to improve his pay-off and as such equalizes the long term average pay-offs of both agents. This kind of solution concept is interesting in applications such as job-scheduling or network routing, where we want to achieve the highest possible overall throughput.

Although the latter technique produces good results, many real-world applications cannot be formalized in a single-stage game. A larger class of games, multi-stage games, can model more general decision problems. By applying the periodical policy technique to a hierarchy of learning agents, we try to create a solution technique that solves both conflicting interest single-stage as well as multi-stage games. For now, our algorithm is designed for stochastic tree-structured multi-stage games as defined by E. Billard (Billard & Lakshmivarahan 1999; Zhou, Billard, & Lakshmivarahan 1999). He views multi-stage (or multi-level) games as games where decision makers at the top-level decide which game to play at the lower levels. Specifically, he assumes there are two agents each consisting of two learning automata. And thus the automata at the high level decide which game the automata at the lower level play. He also proves that for certain types of problems the agents decide not to cooperate in a pure conflicting interest game. Since we will be talking about conflicting games, we need to clarify which solution concept we would like to find. Since are solution technique is based on a social homo egualis concept, we will be looking for a social solution.

We will show that with our technique the agents altern-

[*]Research funded by a Ph.D grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen).

ate between strategies, trying to minimize the difference between pay-offs, resulting in a fair pay-off, even if some of these strategies require cooperation.

In the next section we start with a description and an example of a multi-stage game, more specifically the setup proposed and studied by E. Billard is introduced. We also focus on some of the problems that can occur in pure conflicting (multi-stage) games. Thereafter follows a brief overview of the theory of learning automata and we describe the hierarchical setup of E. Billard and of K.S. Narendra and Parthasarathy. This latter setup is extended with our periodical technique that we also explain in that section. The next section reports the results we obtained with our exclusion technique applied to the games of Billard. The last section concludes and proposes future work.

Multi-Stage games

In (Boutilier 1999) the author extends the traditional Markov decision proces (MDP) to the multi-agent case, called multi-agent Markov decision processes (MMDP). MMDP's can be seen as a MDP with the difference that the actions are distributed over the different agents. Formally a MMDP is a quintuple $\langle \mathbb{S}, \mathbb{A}, \langle A_i \rangle_{\forall i \in \mathbb{A}}, T, R \rangle$ where $\mathbb{S}$ is the set of states, $\mathbb{A}$ the set of agents, A_i a finite set of actions available to Agent i, $T : \mathbb{S} \times A_1 \times \ldots \times A_n \times \mathbb{S} \rightarrow [0, 1]$ the transition function and $R : S \rightarrow \mathbb{R}$ is the reward function for all agents. Thus we are playing a game in a cooperative setting where each agent receives the same reward. MMDP's can also be viewed as a type of stochastic game as formulated by Shapley (Shapley 1953). An example of an MMDP can be seen in Figure 1. In this

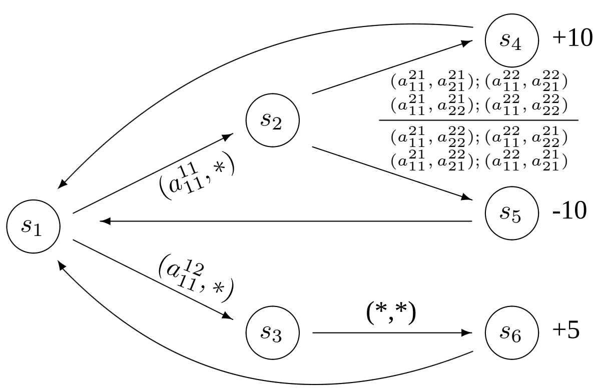

Figure 1: The Opt-In Opt-Out game: A two-stage MMDP with a coordination problem from (Boutilier 1996). Both agents have two actions. A * can be either action.

figure we see 2 agents both with 2 actions. The notation we adopt to describe a certain action is: $a^{level,action}_{agent,automaton}$ (the $automaton$ in the notation will become clear in the next section). The game is divided into 2 stages. At the startstate s_1 both the agents have to take an action. If Agent 1 takes action a^{11}_{11}, we end up in state s_2 where the agents have to coordinate on joint-action $(a^{21}_{11}, a^{21}_{21})$,$(a^{22}_{11}, a^{22}_{21})$,$(a^{21}_{11}, a^{21}_{22})$ or $(a^{22}_{11}, a^{22}_{22})$ to reach the optimal pay-off of $+10$ and to avoid the heavy penalty of -10. On the other hand if at state s_1, Agent 1 takes actions a^{12}_{11} then we end up in stage s_3 where we can only continue to the safe state s_6. This results in a pay-off of $+5$. We can say that at stage 1 the agents play a normal form game M^1 (no reward given in this example) and in stage 2 the agents play a normal form game M^2. The previous example can thus be translated into the matrices M^1 and M^2:

$$M^1 = \left(\begin{array}{c|cc} & a^{11}_{21} & a^{12}_{21} \\ \hline a^{11}_{11} & 0 & 0 \\ a^{12}_{11} & 0 & 0 \end{array} \right)$$

$$M^2 = \left(\begin{array}{c|cccc} & a^{11}_{21}a^{21}_{21} & a^{11}_{21}a^{22}_{21} & a^{12}_{21}a^{21}_{22} & a^{12}_{21}a^{22}_{22} \\ \hline a^{11}_{11}a^{21}_{11} & +10 & -10 & +10 & -10 \\ a^{11}_{11}a^{22}_{11} & -10 & +10 & -10 & +10 \\ a^{12}_{11}a^{21}_{12} & 5 & 5 & 5 & 5 \\ a^{12}_{11}a^{22}_{12} & 5 & 5 & 5 & 5 \end{array} \right)$$

In (Verbeeck, Nowé, & Peeters 2004) we experimented with games that can be categorized as an MMDP. However the MMDP formalization leaves out the games where the agents have different or competing interests. E. Billard introduced a multi-stage game where the agents have pure conflicting interests (Billard & Lakshmivarahan 1999; Zhou, Billard, & Lakshmivarahan 1999). Since the pay-offs of the agents aren't shared in conflicting games, for each joint-action, we have an element in the game matrix of the form (p^h, p^k), where p^h is the probability for Agent h to receive a positive reward of +1 and p^k is the probability for Agent k. However, since we are playing a constant sum game, we can simplify the representation since $p^h = 1 - p^k$ [1]. Thus the actions positive for one agent are negative for the opposing agent and vice versa. In systems where the performance of the global system is just as good as the pay-off of the worst performing agent, the ideal solution for the agents would be to play a joint-action resulting in the pay-off $(0.5, 0.5)$ (if the constant sum equals 1). Unfortunately this action will not always be available or the agents might be biased to other actions resulting in a higher personal pay-off but a worse global performance. Playing periodical policies is proposed in these situations, where the best performing agent temporarily excludes one of his actions. Thus giving the worst performing agent a chance to increase his pay-off. A global result of these exclusions is that the difference between the personal pay-offs is minimized and the average pay-offs go to $(0.5, 0.5)$. We call this the fair solution for all agents.

Hierarchical Learning Automata

In this section we describe briefly the theory of learning automata. We also rewiew how both, we as well as how Billard, use an hierarchical learning automata setup.

[1] Since the reward matrices contain probabilties of having a reward the constant sum is 1. Thus the matrices only need to store one value for each joint-action

Learning Automata

In the context of automata theory, a decision maker that operates in an environment and updates its strategy for choosing actions based upon responses from the environment is called a learning automaton. Formally an automaton can be expressed as a quadruple $\langle A, \mathbf{p}, T, R \rangle$ where $A = (a_1, \ldots, a_r)$ is the set of actions available to the automaton, $\mathbf{p}$ is the probability distribution over these actions, R is the set of possible rewards (e.g. $R \in [0, 1]$) and U is the update function of the learning automaton. The environment can be modeled as $\langle A, \mathbf{c}, R \rangle$ with A the input for the environment, $\mathbf{c}$, the probability vector over the penalties and R the output of the environment. Thus the output of the automaton is the input of the environment and vice versa. This is visualized in Figure 2.

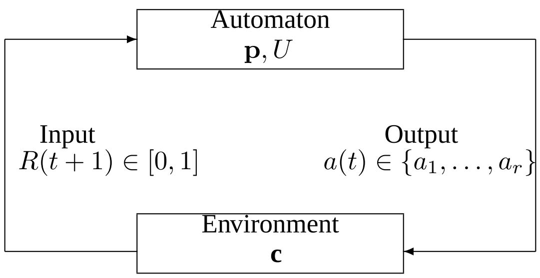

Figure 2: A learning automaton

The update function T greatly determines the behavior of the automaton. In this paper we used the linear reward-inaction update scheme (L_{R-I}). This update scheme behaves well in a wide range of settings and is absolutely expedient and ϵ-optimal (Narendra & Thathachar 1989). This means that the expected average pay-off is strictly monotonically increasing and that the expected average penalty can be brought arbitrarily close to its minimum value.

The idea behind this scheme is to increase the probability of an action if this action was chosen and resulted in a success and to ignore it when the action led to a failure. All the probabilities of the actions that weren't chosen are decreased. The update scheme is given by (where $r = 0$ is a success and $r = 1$ is a failure):

$$\begin{aligned} p_i(t+1) &= p_i(t) + \alpha(1 - r(t))(1 - p_i(t)) \\ &\quad \text{if action } a_i \text{ is chosen at time } t \\ p_j(t+1) &= p_j(t) - \alpha(1 - r(t))p_j(t) \\ &\quad \text{if } a_j \neq a_i \end{aligned}$$

The constant α is called the reward or step size parameter and belongs to the interval $[0, 1]$. In stationary environments $p(t)_{t>0}$ is a discrete-time homogeneous Markov process and convergence results for L_{R-I} have been proved. Despite the fact that multiple automata operating in the same environment makes this environment non-stationary, the L_{R-I} update scheme still produces good results in learning automata games.

Learning automata games

We define a learning automata game as a game where several learning automata are acting simultaneously in the same environment, see Figure 3.

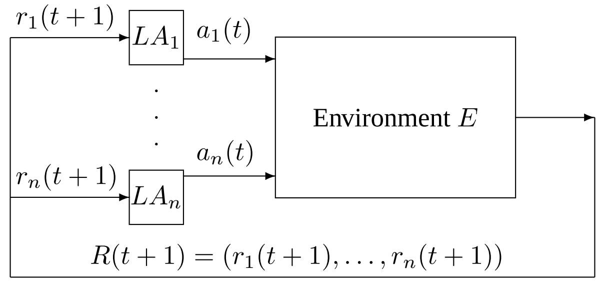

Figure 3: A learning automata game

Let $\mathbb{L} = \{LA_1, \ldots, LA_n\}$ be the set of learning automata acting in the environment E. At timestep t all the learning automata take an action, $a_i(t)$. The joint-action of the learning automata can be represented by $\vec{a}(t) = (a_1(t), a_2(t), \ldots, a_n(t))$. Based on this joint-action $\vec{a}(t)$ the environment provides a response $R(t+1) = (r_1(t+1), r_2(t+1), \ldots, r_n(t+1))$. The responses $r_i(t+1)$ are fed back into the respective learning automata and they update their action probabilities based on this response. All the learning automata are unaware of the other automata, i.e. they get no information about the pay-off structure, the actions, strategies, or even the reward of the other agents.

Narendra and Thathachar proved in (Narendra & Thathachar 1989) that for zero-sum games the L_{R-I} update scheme converges to the equilibrium point if it exists in pure strategies (i.e. if there is a pure Nash equilibrium). In identical pay-off games as well as some non zero-sum games it is shown that when the learning automata use the L_{R-I} update scheme the overall performance improves monotonically. Moreover if the identical payoff game is such that a unique pure equilibrium point exists, convergence is guaranteed. In cases where the game has more than one pure Nash equilibrium the scheme will converge to one of these pure Nash equilibria, which one depends on the initial settings.

The hierarchical learning automata, which we extend with our exclusion ability, are based on these learning automata games and are introduced in the next section.

Hierarchical settings

The hierarchical setup Billard used in (Billard & Lakshmivarahan 1999; Zhou, Billard, & Lakshmivarahan 1999) consists of 2 levels and is depicted in Figure 4.

In this setting of Billard there are 2 agents both with 2 separate learning automata (Agent 1 consists of LA_1^1 at the high level and LA_1^2 at the low level). All of the learning automata have 2 actions. His multi-stage decision process determines which game to play and what action to play within the game. This interaction between the agents goes as follows: first the automata from the high level decide which game to play. If both LA_1^2 and LA_2^1 select game A, this game is played at the lower level (by LA_1^2 and LA_2^2) . If either LA_1^1 or

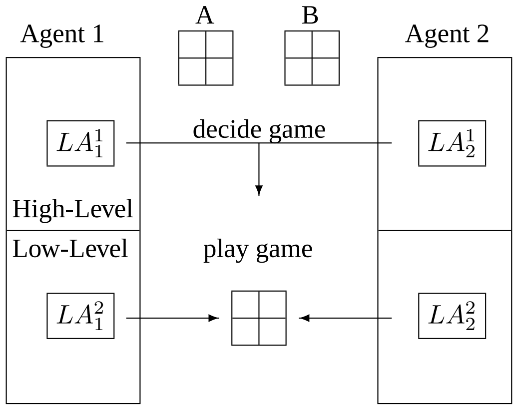

Figure 4: The hierarchical decision structure as used by E. Billard

LA_1^2 selects game B (independently of what the other agent chooses), game B is played at the lower level.

Let $u_1(t)$ and $u_2(t)$ denote the probabilities for LA_1^1 to choose game A or B at timestep t and $u_1(t) + u_2(t) = 1$. Likewise $v_1(t)$ and $v_2(t)$ denote the probabilities for LA_2^1 to choose game A or B. Since game A will be played only when both agents choose game A, the probability for game A to be played is $\delta(t) = u_1(t)v_1(t)$. The average pay-off matrix on the kth play for LA_1^2 and LA_2^2 is a convex combination $D(t) = \delta(t)A + (1 - \delta(t))B$. Billard proved (Billard & Lakshmivarahan 1999; Zhou, Billard, & Lakshmivarahan 1999) for this setup that in games where A and B have saddle-points or pure Nash equilibria (which are equivalent in constant sum games) one of the two agents deviate from choosing game A, resulting in non-cooperation, independent of the games A and B.

The hierarchical set-up we adopted for our exclusion technique is based on learning automata theory and is slightly different from the one defined above and is depicted in Figure 5. We see again 2 agents interacting with each other. Yet for each action of the learning automata at level n, there exists a learning automaton at level $n + 1$. For instance we see that learning automaton LA_{11}^1 at level 1 (which is the top-level) has 2 actions (a_{11}^{11} and a_{11}^{12}) thus at level 2 we have 2 learning automata (LA_{11}^2 and LA_{12}^2). Again we can describe how the interaction between the different agents goes. At the top level (or the first stage of the game) Agent 1 and 2 meet each other in the game with pay-off matrix M^1. Here learning automata LA_{11}^1 and $LA^1 21$ have to take an action. At this point a first reward r^1 is generated . Performing action $a_{o,p}^{l,i}$ by $LA_{o,p}^l$ is equivalent to choosing automaton $LA_{o,p}^{l+1}$ to take an action at the next level. At the second level automata two automata of the next level get to play the game M^2 resulting in a reward r^2. At the end, all the automata that were active in the last play update their action probabilities based on the weighted responses r^1 and r^2, namely $r(t) = \lambda r^1(t) + (1 - \lambda)r^2(t)$ for Agent 1 and $1 - r(t)$ for Agent 2. $\lambda \in [0, 1]$ is a weight factor that determines the weight of the pay-offs, equivalent to the discount factor in reinforcement learning. In the experiments, a reward is only

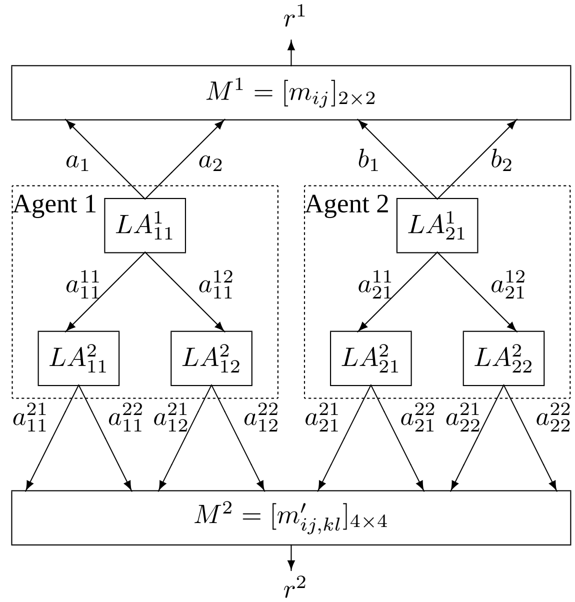

Figure 5: An interaction of two hierarchies of learning automata at 2 stages (Narendra & Thathachar 1989)

given by M^2 therefore we can set our learning rate $\lambda = 0$.

When all the learning automata use a linear reward inaction update scheme and the step sizes of the automata at the lower levels vary over time, the overall performance of the hierarchical automata system will improve at every stage. At any stage t the step sizes of the lower level automata taking part in the game are changed to:

$$\alpha_{reward}^{l+1}(t) \quad = \quad \frac{\alpha_{reward}^{l}(t)}{p_i^l(t+1)}$$

where $\alpha_{reward}^{l}(t)$ is the step size of the parent automaton and $p_i^l(t+1)$ is the probability of the parent for taking action i, where i is the action taken during the last play.

When translating the games by Billard into the setup of learning automata, matrices M^1 and M^2 become respectively:

$$M^1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad M^2 = \begin{pmatrix} A & B \\ B & B \end{pmatrix}$$

where A and B are 2×2 matrices. Thus at the first stage no reward is given and at the second stage game A is only played if both agents select it, otherwise game B is played.

Learning Fair periodical policies

As stated in the introduction we will extend the hierarchical agents of Narendra with the ability to exclude actions temporarily to alternate between policies resulting in an overall fair average reward for all the learning automata. The core of the technique is not new, in (Nowé, Parent, & Verbeeck 2001) an algorithm is introduced to let agents learn fair periodical policies in one-stage conflicting general-sum games. However in this paper we extend this technique to a multi-stage setting.

Before we explain the actual algorithm we introduce the initialization of the necessary parameters for the agents. The pseudo-code for this is given in Algorithm 1. The algorithm itself consists of two different phases, the independent learning phase and the communication phase. During the independent learning phase the agents start to explore their action space without any form of communication and because the agents act as selfish reinforcement learners (using the L_{R-I} update scheme) the group will converge to one of the Nash equilibria. After a given period of time (during which the agents converged) the communication phase takes place (see Algorithm 2). For the current experiments the learning period is fixed and set by the author. However it is possible to let the agents determine for themselves when they converged to a pure Nash equilibrium.

Algorithm 1 Pseudo-code for the initialization of the agents

```
Initialize actionprobabilities: p_i ← 1/(number−of−actions).
my-average ← 0
my-last-average ← 0
excluding ← true
```

Algorithm 2 Pseudo-code communication phase for periodical policies

```
Communicate my-average and my-last-average to the opponent
if excluding = true then
    if my-averge > opponent-average ∨ my-last-average > opponent-last-average then
        Exclude current action
    else
        Include all actions
        excluding ← false
    end if
else
    if my-average > opponent-average ∧ my-last-average > opponent-last-average then
        Exclude action 2 at top level
        excluding ← true
    else
        excluding ← false
    end if
end if
my-last-average ← 0
```

The communicate phases works as follows: the agents communicate their global average pay-off and the average-pay-off they received during the last learning phase to their opponent. The agent that is excluding actions checks whether he still is the best performing agent (i.e. a higher global average or a better average during the last phase). If he still is performing better he excludes another action. The agent starts by excluding an action at the top level and during a next phase he will exclude one action at the next level, the maximum number of actions he excludes is therefore two. If the agent is not excluding (meaning that he is the worst performing agent) then he checks whether he switched from the worst performing agent to the better performing agent. If this is the case, it is now his turn to exclude one or more actions.

In the introduction we explained about independent agents and joint-action learners. Actually the hierarchical agents equipped with the extended exclusion technique fall in neither category. Since the agents don't act completely independent (they communicate only after a long period of learning), we call them pseudo-independent agents.

Experiments

In this section we report on the results of the experiments we did with hierarchical learning automata with exclusions. We played two player pure conflicting interest games as described in a previous section. We start with simple games where both matrices have pure Nash equilibria and we end with difficult games without saddle points or pure Nash equilibria. Because a reward is given only at the end of the game, matrix M^1 is the zero matrix and $\lambda = 0$ (equal to the setting of Billard). We experimented with different values for the reward parameter (α). The value of α has effect on the speed of convergence rather than on the convergence itself.

Games with saddle-points and Nash equilibria

First we present the results for games where both matrices have a saddle-point and thus have a pure Nash equilibrium. We played the game with the following matrices (the saddle-points are underlined). In these matrices the saddle-points are coincident (from the viewpoint of Agent 1).

$$A = \begin{pmatrix} \underline{0.6} & 0.8 \\ 0.35 & 0.9 \end{pmatrix} \quad B = \begin{pmatrix} \underline{0.7} & 0.9 \\ 0.6 & 0.8 \end{pmatrix}$$

According to the proof of Billard, Agent 1 will deviate from playing game A since the saddle-point in game B leads to a higher pay-off for Agent 1, expressed as $(u(k), v(k)) \rightarrow ((0,1),(1,0))$. Figure 6 shows the results when we play the game with the hierarchical setup of Narendra without the exclusion technique. On the X-axis we see the number of epochs and on the Y-axis the probability of taking action 1 (i.e. choosing game A) at the top level. The result is the average over 50 runs with reward parameter $\alpha = 0.0009$. We see in the figure that Agent 1 is not very keen on playing game A and forces on its own to play game B (remember that game A is only played if both agents choose game A). This behavior was theoretically predicted by Billard since the saddle-point of game B is better for Agent 1 (Zhou, Billard, & Lakshmivarahan 1999). Since the agents play game B in the long run, and since the agents will converge to a Nash equilibrium in this game, the pay-off for Agent 1 will be 0.7 and Agent 2 his pay-off will therefore be 0.3, despite the fact that both agents can receive a better pay-off, although not simultaneously. It is clear that if the performance of the system is determined by the pay-off of the worst performing agent then this is not a good result. By equipping the hierarchical agents with our technique to play periodical

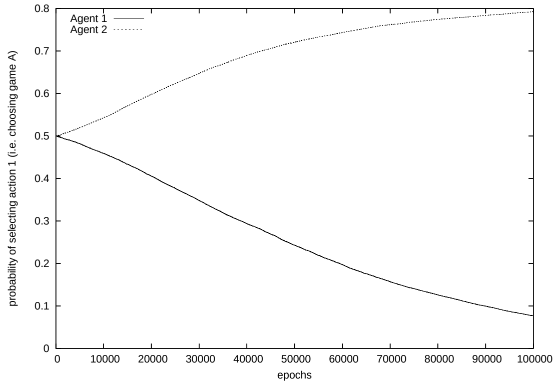

Figure 6: Hierarchical agents playing a game with coincident saddle-points.

policies we obtain a fairer result as shown in Figure 7 (average over 20 runs, $\alpha = 0.02$). The Y-axis plots the average pay-off.

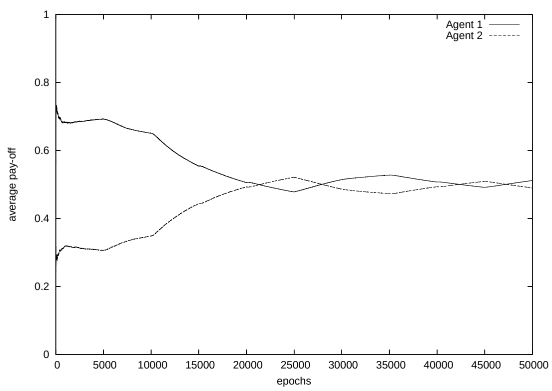

Figure 7: Hierarchical agents playing periodical policies in a game with coincident saddle-points.

Theoretical results can also be proved when the saddle-point are non-coincident (Billard & Lakshmivarahan 1999; **?**), for example the game matrices:

$$A = \begin{pmatrix} \underline{0.6} & 0.8 \\ 0.35 & 0.9 \end{pmatrix} \quad B = \begin{pmatrix} 0.6 & 0.8 \\ \underline{0.7} & 0.9 \end{pmatrix}$$

where for Agent 1 the saddle-points are non-coincident. Again we plot the probability of choosing game matrix A, averaged over 50 runs with reward parameter $\alpha = 0.0003$ (see Figure 8). The theory of Billard predicts that in this game the probabilities of LA_1^1 and LA_2^1 converge to $(u(k), v(k)) \to ((0, 1), (1, 0))$. Thus again the agents will not play the cooperation game and instead converge to the Nash equilibrium in game B, resulting in a pay-off of 0.7 for Agent 1 and 0.3 for Agent 2. The social agents that play periodical policies are again able to alternate between personal better actions as we see in Figure 9, resulting in an average of 0.5 for both agents in the long run. The plot is an average over 25 runs with $\alpha = 0.02$.

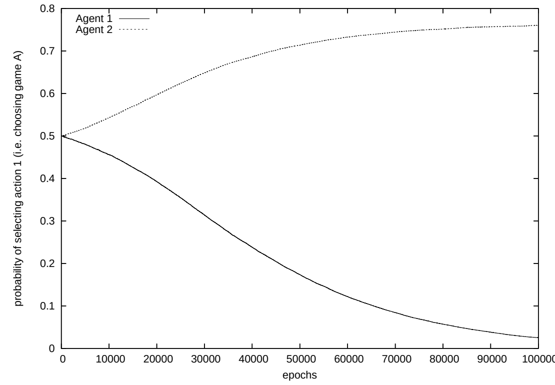

Figure 8: Hierarchical agents playing a game with non-coincident saddle-points.

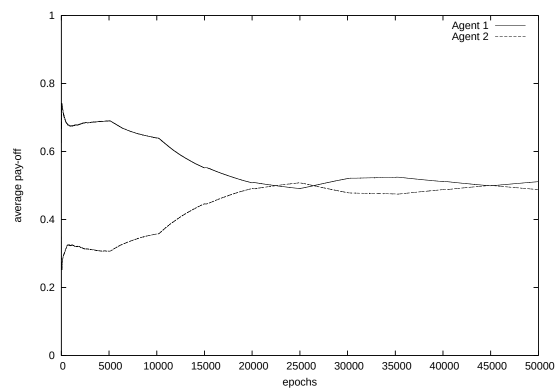

Figure 9: Hierarchical agents playing periodical policies in a game with non-coincident saddle-points.

Games without saddle-points and Nash equilibria

In the above games, both reward matrices A and B contained saddle-points. The results generated by the setup of Billard are proved in (Billard & Lakshmivarahan 1999; Zhou, Billard, & Lakshmivarahan 1999). In these publications the authors mention no theoretical results for matrices where only one the matrices A and B has a saddle-point or where neither has a saddle-point. However, they report on experimental results where agents play games of this particular type. Take for instance the matrices:

$$A = \begin{pmatrix} \underline{0.6} & 0.8 \\ 0.35 & 0.9 \end{pmatrix} \quad B = \begin{pmatrix} 0.6 & 0.2 \\ 0.35 & 0.9 \end{pmatrix}$$

Note that the matrices are identical except for one value and only matrix A has a saddle-point. When playing this game with the hierarchy proposed by Billard, Agent 2 is more attracted to game B since he gets a better pay-off for the differing joint-action. Thus Agent 2 his probability of choosing game A converges to 0. And since matrix B has no saddle-point the agents oscillate between different joint-actions giving no clear answer about what the average pay-off will be. However when we play this game with the hierarchical setup by Narendra this does not happen, instead the agents oscillate between the matrices A and B. Agent 1 has a slightly higher preference for game A and Agent 2 a higher preference for game B as we see in Figure 10. The plot is the average over 20 runs, with a learning rate $\alpha = 0.0001$ and one run consists of 2 million epochs.

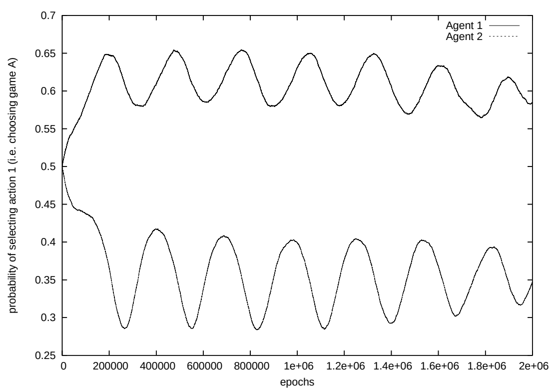

Figure 10: Hierarchical agents playing a game where only game A has a saddle-point.

The social agents are again able to find a fair solution for both agents, resulting in an average pay-off of 0.5 for both the agents (see Figure 11). In this figure, we do not plot an average pay-off instead we plot one long run to show how close our solution technique reaches to the optimal solution of 0.5. In the final experiment we played a game where none of the matrices have a saddle-point (thus also no pure Nash equilibrium). Billard showed empirically that his learning automata converged to very different solution in multiple runs. Figure 12 shows how the periodical polices find a fair solution.

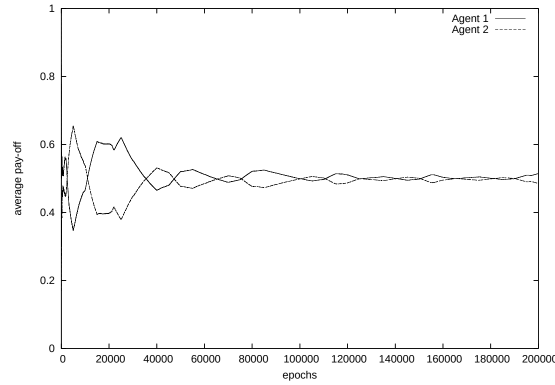

Figure 11: A typical run of social agents playing a game where only game A has a saddle-point.

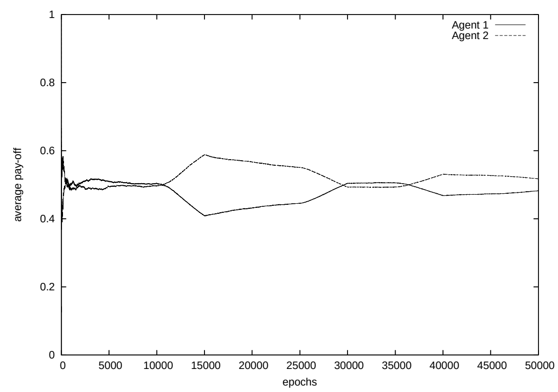

Figure 12: A typical run of social agents playing a game without saddle-points.

Discussion

In this paper we made a first step in the direction of enhancing independent agents with social behavior. By extending hierarchical learning automata with the ability to play different periodical policies we were able to create a distributed framework that can find a fair solution in pure conflicting games, in particular for the constant sum games. We defined a fair as a solution concept where the relative difference between the agents is minimized. The theory of Billard showed that hierarchical agents without our extension are not able to find good solutions. Instead they show oscillating behavior or converge to a pure Nash equilibrium at most.

In systems where the global performance is only as good as the performance of the worst performing agent this solution concept can be used. Examples of applications that fall under this category are load balancing, job scheduling or network routing. The idea of alternating between policies by temporarily excluding actions originated in single-stage games where it proved its value in a single-stage job scheduling application (Nowé, Parent, & Verbeeck 2001).

For now we only showed, by means of experiments, how the solution technique works in two agent, pure conflicting games consisting of two levels that are built according to structure explained above. We are currently investigating how we can provide a mathematical proof why our technique is able to find the solution we propose. Other research could focus on scaling the algorithm to a more common general-sum, multi-agent, multi-level, mult-actions case.

References

Billard, E., and Lakshmivarahan, S. 1999. Learning in multilevel games with incomplete information-part i. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* 29(3):329–339.

Boutilier, C. 1996. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, 195 – 210.

Boutilier, C. 1999. Sequential optimality and coordination in multiagent systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 478 – 485.

Brafman, R. I., and Tennenholtz, M. 2003. Learning to coordinate efficiently: A model-based approach. *Journal of Artificial Intelligence Research* 19:11–23.

Hu, J., and Wellman, M. 1998. Multi agent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 242–250.

Kapetanakis, S.; Kudenko, D.; and Strens, M. 2003. Learning to coordinate using commitment sequences in cooperative multi-agent systems. In *Proceedings of the Third Symposium on Adaptive Agents and Multi-agent Systems, (AISB03) Society for the study of Artificial Intelligence and Simulation of Behaviour.*

Lauer, M., and Riedmiller, M. 2000. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the seventeenth International Conference on Machine Learning*.

Littman, M. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, 157–163.

Mukherjee, R., and Sen, S. 2001. Towards a pareto-optimal solution in general-sum games. In *Working Notes of the Fifth Conference on Autonomous Agents*.

Narendra, K., and Thathachar, M. 1989. *Learning Automata: An Introduction*. Prentice-Hall International, Inc.

Nowé, A.; Parent, J.; and Verbeeck, K. 2001. Social agents playing a periodical policy. In *Proceedings of the twelfth European Conference on Machine Learning*, 382–393. Freiburg, Germany: Springer-Verlag LNAI2168.

Shapley, L. 1953. Stochastic games. In *Proceedings of the National Academy of Sciences*, volume 39, 327–332.

Verbeeck, K.; Nowé, A.; and Peeters, M. 2004. Multi-agent coordination in tree structured multi-stage games. In *Proceedings of the Fourth Symposium on Adaptive Agents and Multi-Agent Systems, (AISB04) Society for the study of Artificial Intelligence and Simulation of Behaviour*.

Verbeeck, K.; Nowé, A.; and Tuyls, K. 2003. Coordinated exploration in stochastic common interest games. In *Proceedings of the Third Symposium on Adaptive Agents and Multi-agent Systems, (AISB03) Society for the study of Artificial Intelligence and Simulation of Behaviour.*

Wang, X., and Sandholm, T. 2002. Reinforcement learning to play an optimal nash equilibrium in team markov games. In *Proceedings of the 16th Conference on Neural Information Processing Systems (NIPS'02)*.

Zhou, J.; Billard, E.; and Lakshmivarahan, S. 1999. Learning in multilevel games with incomplete information-part ii. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* 29(3):340–349.