Competition and Coordination in Stochastic Games

Andriy Burkov, Abdeslam Boularias, and Brahim Chaib-draa

DAMAS Laboratory
Université Laval
G1K 7P4, Quebec, Canada
{burkov,boularia,chaib}@damas.ift.ulaval.ca

Abstract. Agent competition and coordination are two classical and most important tasks in multiagent systems. In recent years, there was a number of learning algorithms proposed to resolve such type of problems. Among them, there is an important class of algorithms, called adaptive learning algorithms, that were shown to be able to converge in self-play to a solution in a wide variety of the repeated matrix games. Although certain algorithms of this class, such as Infinitesimal Gradient Ascent (IGA), Policy Hill-Climbing (PHC) and Adaptive Play Q-learning (APQ), have been catholically studied in the recent literature, a question of how these algorithms perform versus each other in general form stochastic games is remaining little-studied. In this work we are trying to answer this question. To do that, we analyse these algorithms in detail and give a comparative analysis of their behavior on a set of competition and coordination stochastic games. Also, we introduce a new multiagent learning algorithm, called ModIGA. This is an extension of the IGA algorithm, which is able to estimate the strategy of its opponents in the cases when they do not explicitly play mixed strategies (e.g., APQ) and which can be applied to the games with more than two actions.

1 Introduction

Competition and coordination between autonomous agents are two classical and most important tasks in multiagent systems. Coordination is especially important in multi-robotic systems where a number of non-adversarial robots (but not necessarily explicitly cooperative) are aimed to accomplish a task while being limited in communication and in knowledge about principles of rationality underlying their counterparts. On the other hand, competition is a natural condition of most real life situations. The agents that share limited resources, negotiate about prices and, in general, have proper interests first or last find themselves in a competitive situation.

Typically, multiagent environments are modeled as *stochastic games* [1]. Stochastic game is a model to represent multi-state multiagent environments having Markovian property and a stochastic inter-state transition rule, and can be used to model inter-agent interactions in such environments. Formally, a

satochastic game is a tuple $(n, \mathbf{S}, \mathcal{A}^{1\dots n}, T, R^{1\dots n})$, where n is the number of agents, $\mathbf{S}$ is the set of states $\mathbf{s} \in \mathbf{S}$ now represented as vectors, $\mathcal{A}^j$ is the set of actions $a^j \in \mathcal{A}^j$ available to agent j, $\mathbf{A}$ is the joint action space $\mathcal{A}^1 \times \ldots \times \mathcal{A}^n$, T is the transition function: $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \mapsto [0, 1]$, R^j is the reward function for agent j: $\mathbf{S} \times \mathbf{A} \mapsto \mathbb{R}$ and $\mathbf{s}_0 \in \mathbf{S}$ is the initial state.

Further, in the article we will refer to a game theoretic terminology, therefore let's introduce some useful notions of the Game Theory here. A matrix game is a tuple $(n, \mathcal{A}^{1\dots n}, R^{1\dots n})$, where n is the number of players, $\mathcal{A}^j$ is the strategy space of player $j, j = 1 \dots n$, and the value function $R^j : \mathcal{A}^1 \times \mathcal{A}^n \mapsto \mathbb{R}$ defines the utility for player j of a joint action $\mathbf{a} \in \mathbf{A} = \mathcal{A}^1 \times \ldots \times \mathcal{A}^n$. A *mixed* strategy for player j is a distribution π^j, where $\pi^j_{a^j}$ is the probability for player j to select some action a^j. A strategy is *pure* if $\pi^j_{a^j} = 1$ for some a^j. A *strategy profile* is a collection $\mathbf{\Pi} = \{\pi^j | j = 1 \dots n\}$ of all players' strategies. A *reduced profile for player* j, $\mathbf{\Pi}^{-j} = \mathbf{\Pi} \backslash \{\pi^j\}$, is a strategy profile containing strategies of all players except j, and $\mathbf{\Pi}^{-j}_{\mathbf{a}^{-j}}$ is the probability for players $k \neq j$ to play a joint action $\mathbf{a}^{-j} \in \mathbf{A}^{-j} = \mathcal{A}^1 \times \ldots \times \mathcal{A}^{j-1} \times \mathcal{A}^{j+1} \times \ldots \times \mathcal{A}^n$ where $\mathbf{a}^{-j}$ is $\langle a^k | k = 1 \dots n, k \neq j \rangle$.

Recently, there was a number of learning algorithms proposed to resolve decision problems in stochastic games [1–11]. Typically, these algorithms are constructed to iteratively *play* a game with an opponent, and, by playing this game, to *converge* to a solution. Solution in game theory is called *equilibrium*. We say that the playing strategies of all agents forms an equilibrium in a stochastic game if a unilateral deviation of an agent from its current strategy contradicts its principles of rationality (usually, maximization of the utility).

Among the learning algorithms proposed for the stochastic games, there is an important class, which we call *adaptive learning algorithms*, that are proven to be able to converge in self-play (i.e., when learning "against" agents that are using the same learning algorithm) to an equilibrium solution in a wide variety of repeated matrix games. The advantage of the adaptive learning algorithms with respect to other class of multiagent learning algorithms, such as *equilibrium learning algorithms* [1, 7, 8], is that the latter are calculating an equilibrium solution regardless the other agents' actual behavior (i.e., equilibrium learners assume that their opponents are rational, though they may not be) and their convergence is limited to a number of cases where these equilibria are identifiable. The adaptive learning agents, on the contrary, make no assumptions about their opponents' rationality and learning capabilities, and about the solution type they are searching. Adaptive agents are adapting to their opponents and a solution is found as an *emerging result* of this adaptation. Among adaptive algorithms, the most outstanding and theoretically sound ones are Infinitesimal Gradient Ascent (IGA) [4], Policy Hill-Climbing (PHC) [2] and Adaptive Play Q-learning (APQ) [3]. These algorithms were empirically tested by their respective authors on the different test benches. However, although these algorithms were tested on a number of repeated matrix games and on some examples of stochastic games, a number of questions is remaining. First, whether these algorithms are well extensible to the general form stochastic games. Second, how

these algorithms are comparable between themselves (in terms of convergence and relative effectiveness against each other).

In this paper we are trying to answer these questions. To do that, we analyse these algorithms in detail and give a comparative analysis of their behavior on a set of competition and coordination stochastic games, which includes two-robot-on-the-grid coordination game and two-robot-predator-prey competition game. Further, we introduce a new multiagent learning algorithm, called ModIGA, a modification of the IGA algorithm.

2 Adaptive Learning Algorithms

As we noted above, to learn a "good" policy in stochastic games a number of adaptive algorithms have been proposed. They can be conventionally divided onto three groups: (1) Opponent Modelling algorithms [3, 5], (2) Policy Gradient based algorithms [2, 4] and Adaptivity Modelling algorithms [9–11]. Although the algorithms of the third group are very interesting and empirically shown to have several attractive properties, such as exploiting their opponents in adversarial games [10, 11] and converging to a solution maximizing welfare of both players in non-adversarial two-player matrix games [11], there are still no theoretical proofs of their correctness, while in the first two groups there are algorithms that were formally proven to have such properties as rationality and convergence. In our analysis, we opted for the following three adaptive learning algorithms: Infinitesimal Gradient Ascent (IGA) [4], Policy Hill-Climbing (PHC) [2] and Adaptive Play Q-learning (APQ) [3] because, as we have just noted, (1) they are theoretically proven to converge to a stable solution (at least in self-play), (2) they represent two major classes of learning algorithms, those able to play pure strategies only (APQ) and those able to play mixed strategies (IGA, PHC).

In this section we analyze in detail these algorithms. Also, we introduce a new multiagent learning algorithm, called ModIGA. This is an extension of the IGA algorithm, which is able to estimate the strategy of its opponents in the cases when they do not explicitly play mixed strategies (e.g., APQ) and which, unlike IGA, can be applied to the games with more than two actions.

2.1 Adaptive Play Q-learning

Formally, each player j playing Adaptive Play [12] saves in memory a history $H_t^j = \{\mathbf{a}_{t-p}^{-j}, \ldots, \mathbf{a}_t^{-j}\}$ of the last p joint actions played by the other players. To select a strategy to play at time $t+1$ each player randomly and irrevocably samples from H_t^j a set of examples of length l, $\hat{H}_t^j = \{\mathbf{a}_{k_1}^{-j}, \ldots, \mathbf{a}_{k_l}^{-j}\}$, and calculates the empiric distribution $\hat{\mathbf{\Pi}}^{-j}$ as an approximation of the real reduced profile of strategies played by the other players, using the following:

$$\hat{\mathbf{\Pi}}_{\mathbf{a}^{-j}}^{-j} = \frac{C(\mathbf{a}^{-j}, \hat{H}_t^j)}{l} \tag{1}$$

where $C(\mathbf{a}^{-j}, \hat{H}_t^j))$ is the number of times that the joint action $\mathbf{a}^{-j}$ was played by the other players according to the set $\hat{H}_t^j$. Given the probability distribution over the other players' actions, $\hat{\mathbf{\Pi}}^{-j}$, the player j plays its best reply, $BR^j(\hat{\mathbf{\Pi}}^{-j})$, to this distribution with some exploration. If there are several equivalent best replies, the player j randomly chooses one of them. Young [12] proved the convergence of Adaptive Play to an equilibrium when played in self-play for a big class of games such as the coordination and common interest games.

Adaptive Play Q-learning (APQ) is an extension of Young's algorithm to the multi-state stochastic game context. To do that, the usual single-agent Q-learning update rule [13] was modified to consider multiple agents as follows:

$$Q^j(s, \mathbf{a}) \leftarrow (1-\alpha)Q^j(s, \mathbf{a}) + \alpha[R^j(s, \mathbf{a}) + \gamma \max_{a^j \in \pi^j(s')} U^j(\hat{\mathbf{\Pi}}(s') \cup \{\pi^j(s')\})]$$

where j is an agent, $\mathbf{a}$ is a joint action played by the agents in state $s \in \mathcal{S}$, $Q^j(s, \mathbf{a})$ is the current value for player j of playing the joint action $\mathbf{a}$ in state s, $R^j(s, \mathbf{a})$ is the immediate reward the player j receives if the joint action $\mathbf{a}$ is played in the state s and $\pi^j(s')$ are all possible *pure* strategies that are available for player j in state s'.

2.2 Infinitesimal Gradient Ascent

To examine the dynamics of using policy gradient in repeated games, Singh, Kearns and Mansour modeled this process for two-player, two-action matrix games. They called their approach Infinitesimal Gradient Ascent (IGA) [4]. Unlike APQ, which can learn and play pure strategies only, IGA players were designed to be capable to learn and play mixed strategies.

The problem of the gradient ascent in matrix games was modelled by Singh and colleagues as having two payoff matrices for the row and column players, r and c, as follows:

$$R^r = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}, \quad R^c = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

If row player r selects an action i and the column player c selects an action j, then the payoffs they obtain are R^r_{ij} and R^c_{ij} respectively.

Because the game being modelled has only two available actions for each agent, a mixed strategy can be represented as a single value. If we let $\alpha \in [0, 1]$ be a probability the player r selects the action 1, then $1-\alpha$ will be the probability to play the action 2. Similarly, we can define as $\beta \in [0, 1]$ and $1-\beta$ the probabilities to play actions 1 and 2 by the player c. The expected utility of playing a strategy profile $\{\alpha, \beta\}$ for player r can then be calculated as follows:

$$U^r(\{\alpha, \beta\}) = r_{11}\alpha\beta + r_{22}(1-\alpha)(1-\beta) + \ r_{12}\alpha(1-\beta) + r_{21}(1-\alpha)\beta$$

At each game iteration, to estimate the effect of changing its current strategy, player r calculates a partial derivative of the expected utility with respect its current mixed strategy:

$$\frac{\partial U^r(\{\alpha, \beta\})}{\partial \alpha} = \beta u - (r_{22} - r_{12})$$

where $u = (r_{11} + r_{22}) - (r_{21} + r_{12})$.

Having calculated the gradient, IGA agent adjusts its current strategy in the direction of this gradient as to maximize its utility:

$$\alpha_{t+1} = \alpha_t + \eta \frac{\partial U^r(\{\alpha_t, \beta_t\})}{\partial \alpha}$$

where η is a step size, usually $0 < \eta \ll 1$. Similar equations can be written for the column player c as well. Obviously, the opponent's mixed strategy is supposed to be known by the players.

Singh and colleagues proved the convergence of IGA to an equilibrium (or, at least, to the equivalent average reward of an equilibrium), when played in self-play, in the case of the infinitesimal step size ($\lim_{\eta \to 0}$).

2.3 Policy Hill-Climbing

The first practical algorithm capable to play mixed strategies that realized the convergence properties of IGA was Policy Hill-Climbing (PHC) learning algorithm [2]. The PHC algorithm requires neither knowledge of the opponent's current stochastic policy nor its recently executed actions (the latter is required for the APQ algorithm, for example). The algorithm, in essence, performs hill-climbing in the space of mixed strategies and is, in fact, a simple modification of the single-agent Q-learning technique. It is composed of two parts. The first part is the reinforcement learning component, which is based on the Q-learning technique to maintain the values of the particular actions in the states:

$$\hat{Q}^j(\mathbf{s}_t, a_t^j) \leftarrow (1-\alpha)\hat{Q}^j(\mathbf{s}_t, a_t^j) + \alpha \left[R_t^j(\mathbf{s}_t, a_t^j) + \gamma \max_{a_{t+1}^j} \hat{Q}(\mathbf{s}_{t+1}, a_{t+1}^j) \right]$$

The second part is the game theoretic component, which maintains the current mixed strategy in each system's state. The policy is improved by increasing the probability that the agent selects the highest valued action, by using the small step δ which is called learning rate:

$$\pi_{a^j}^j(\mathbf{s}) \leftarrow \pi_{a^j}^j(\mathbf{s}) + \Delta_{\mathbf{s}a^j} \tag{2}$$

where

$$\Delta_{\mathbf{s}a^j} = \begin{cases} -\delta_{\mathbf{s}a^j} & \text{if } a^j \neq \operatorname{argmax}_{a'^j} \hat{Q}(\mathbf{s}, a'^j) \\ \sum_{a'^j \neq a^j} \delta_{\mathbf{s}a'^j} & \text{otherwise} \end{cases} \tag{3}$$

$$\delta_{\mathbf{s}a^j} = \min\left(\pi^j(\mathbf{s}, a^j), \frac{\delta}{|\mathcal{A}^j| - 1}\right) \tag{4}$$

while constrained to a legal probability distribution. If $\delta = 1$ the algorithm is equivalent to the single-agent Q-learning as soon as the learning agent will deterministically execute the best action (greedy policy). As well as the single-agent Q-learning, this technique is rational and converges to the optimal solution if the other players follow a fixed (stationary) policy. However, if the other players are learning, the PHC algorithm may not converge to a stationary policy though its average reward will converge to the reward of a Nash equilibrium [2].

2.4 ModIGA

While IGA demonstrated good convergence results, its applicability in reality is limited to the two-action case where the opponent is playing an *identifiable mixed strategy*. This assumption does not reflect real nature problems. In reality, we are usually expecting agent to observe the opponent's actions rather than its mixed strategy. Furthermore, the real life learning agents, as well as their counterparts, are intended to have more than two available actions.

We introduce an improved version of the IGA algorithm, which is able to learn a mixed strategy for more than two simple actions and to estimate the strategy of its opponents even if they do not explicitly play a mixed strategy. (This is the case, for example, when playing against APQ algorithm.)

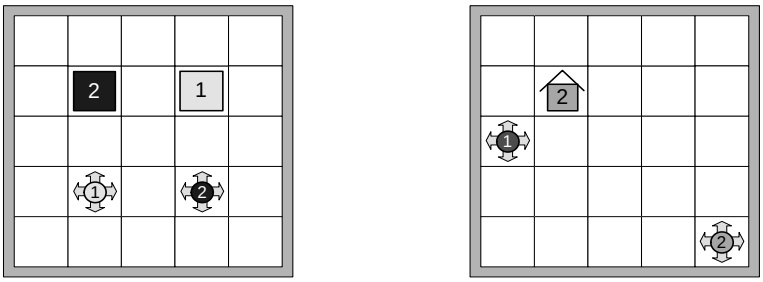
To make IGA agent able to estimate the strategy of its opponent we used the Adaptive Play's probability estimation technique described in Subsection 2.1. Having calculated the estimation of the opponent's strategy, $\mathbf{\Pi}_{t+1}^{-j}$, the IGA agent is able to calculate the gradient of its own current strategy by using the equations of Subsection 2.2. It is important to note that even if the opponent is not playing explicitly mixed strategies (e.g., APQ), the IGA agent using this technique is still able to calculate the gradient of its strategy, though this gradient will be calculated to the averaged opponent's strategy rather than to its real strategy.

When there are more than two actions at the agents' disposal, the techniques of gradient calculation and strategy update of two-action case do not work well and, as we observed, cannot be readily extended to the case of multiple actions. First, this is because in this case there can not be one variable to represent the agent's strategy and another one depending on it. Second, in the two-action case, an increase of the probability to make one action tacitly and at the same degree decreased the probability of the other action to be executed, which always kept the total probability equal to 1. In the multiple action case, this is no longer so.

To deal with this problem, we adapted the technique used in PHC algorithm. It consists in updating the strategy in the direction of the action with the higher Q-value (see equations 2,3,4). But unlike PHC, in our ModIGA algorithm, δ is proportional to the Q-value. This keeps the gradient ascent property, i.e., the step in the direction of the gradient is proportional to the gradient itself.

3 Environments

To make our experiments, we programmed two stochastic games, which model two the most important types of multiagent interactions: coordination and competition. The first game is called two-robot-on-the-grid coordination problem, first introduced by Hu and Wellman [7]. The game consists of the *grid* containing a number of *cells*. There are *two robots* on the grid, which have four available actions, *up*, *down*, *left* and *right*. By making actions, robots are able to transit between cells with a certain probability of the transition success. If transition is successful, robot changes the cell in the intended direction. Otherwise, robot



(a) (b)

Fig. 1. (a) The two-robot-on-the-grid coordination problem and (b) The two-robot-predator-prey competition game.

keeps its current position. For each action made in each cell, except the goal cell, robot receives a *negative reward*. A *collision* is possible if robots are trying to transit into the same cell or to trade cells. In the case of collision, robots receive a negative *collision reward*. The goal of each robot hence is to reach its respective goal cell by collecting the minimal value of negative reward. In our experiments we set the following values of the parameters of the model. The action reward in all non-goal cells is -0.04 and is 0 in the goal cell, the collision reward is 0.1, the probability of action success is 0.9 and the discount factor is 0.95. The configuration of the grid and the start and goal cells of robots are depicted in Figure 1(a).

The second stochastic game we programmed is called two-robot-predator-prey competition game. In this game, there are two robots on the same grid as in the coordination game, but the robots play different roles. The first robot (player 1) is called "predator" and its goal in the game is to catch (to achieve a collision with) the second robot, called "prey". The goal of the "prey" (player 2) is to reach a refuge where it cannot be catched. I.e., the *goal situations* for both robots are opposite. If the predator has achieved its goal (i.e., catched the prey) its reward for any action in this state is 0 and the prey, in turn, receives a negative reward of -1 for any action. On the other hand, if the prey has reached the refuge, its reward for any action in this state is 0 and the reward of the predator is -1 regardless its position and action. In all other states robots receive a negative reward of -0.04 for any action. So, we see, that this game is strictly competitive. We set the following values of the other parameters of the model. The probability of action success of predator was set to 0.9, the same parameter of the prey was set to 0.65. These values equalize the chances of winning of both predator and prey, as it was determined in self-play (when both predator and prey used the same learning algorithms). The discount factor was set to 0.95. The configuration of the grid, the start cells and the refuge cell for the prey are depicted in Figure 1(b).

4 Experiments

In our experiments we compared the convergence processes and the final solution quality of all algorithm pairs (i.e., IGA versus IGA, IGA versus PHC, and so on) in the both environments presented in Figure 1. The curves in Figures 2-5 show the results of these experiments.

Figures 2-3 show the results of the experiments in the two-robot-on-the-grid coordination problem. The curves represent the average number of inter-cell transitions of player 1 in one trial as a function of the number of trials. To build each curve, we averaged data over 10 similar experiments. (The results are shown for the first agent only, because the curves for the second agent are the same.) To reflect the convergence speed of each algorithm pair, Figure 2 represents the first $50,000$ trials of the learning process. We can easily see that the IGA×IGA pair converges slower than the other pairs, and, on the contrary, the pair PHC×PHC demonstrates the fastest convergence speed. This can be explained by the fact that the learning space of the APQ and IGA algorithms is $|\mathbf{S}||\mathcal{A}|^2$, since they learn in the space of joint actions, instead of $|\mathbf{S}||\mathcal{A}|$ of the PHC algorithm, which considers its own actions only.

Figure 3 reflects the final $100,000$ trials of the same learning processes. These curves reflect the solution quality of each algorithm pair. We can see here that whereas PHC demonstrated the faster convergence speed in the first learning trials, all algorithm pairs with a participation of PHC demonstrated a worse final solution quality, i.e., in these curves, the final value of average trial length is higher than this for the algorithm pairs without PHC. On the other hand, the cases APQ×APQ and IGA×IGA demonstrated the best average solutions. This can be explained by the fact that both APQ and IGA can observe the actions of their opponents, and, by so doing, to adapt better to the strategy of the opponent. Moreover, since in the two-robot-on-the-grid problem the solution is deterministic (a pair of trajectories) and APQ learns pure strategies directly, it is obvious that in that case the solution found by APQ×APQ cannot be worse than the others.

In our opinion, the results obtained, in particular the empirical convergence of the algorithms of different types against each other, are very interesting, because there have been no theoretical guarantees that these algorithms converge when playing not against themselves. This could be explained by the the similarity of the convergence curves of these algorithms in self-play. Hence, the policies generated at the end of each trial differ not much. Thus, the agents had almost the same behavior when we combined these different algorithms in one play. Additionally, the convergence properties can be held in this situation, because the agents were not able to distinguish whether the other agent was using the same algorithm or not.

Figures 4-5 show the results of the experiments in the two-robot-predator-prey problem. As in the coordination problem's case, we tested all the possible two-by-two combinations of the chosen algorithms. The curves represent average trial length of the predator agent. For the same reasons as sated above, we did not present the curves for the prey agent.

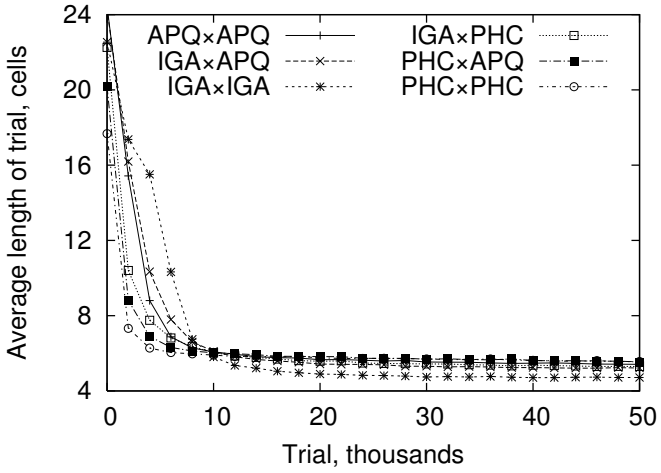


Fig. 2. The optimal trajectory learning in a 5×5 two-robot-on-the-grid game. The curves reflect the length of a trial as a function of the trials number, where the agents use the algorithms PHC, IGA/ModIGA and APQ: the first $50,000$ trials.

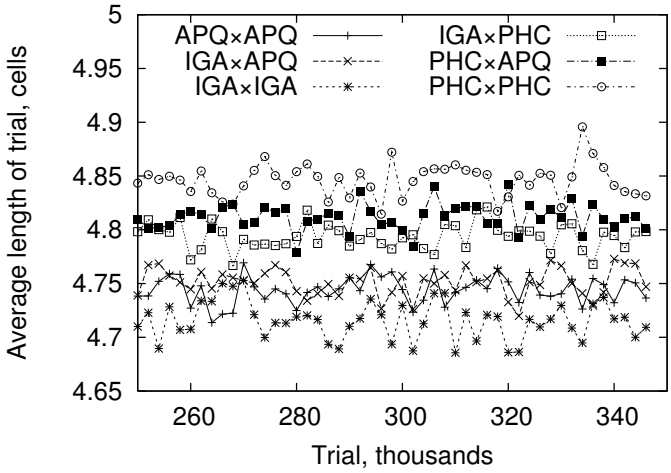


Fig. 3. The optimal trajectory learning in a 5×5 two-robot-on-the-grid game. The curves reflect the length of a trial as a function of the trials number, where the agents use the algorithms PHC, IGA/ModIGA and APQ: the final $100,000$ trials.

Similarly to the results obtained in the coordination game, in this adversarial game we observed the convergence to a stable value for each algorithm pair. Because of the same factors, the convergence speed during the first $50,000$ trials was the slowest for the IGA×IGA algorithm pair and the fastest for the PHC×PHC case (Figure 4). However, in terms of the solution quality (final value

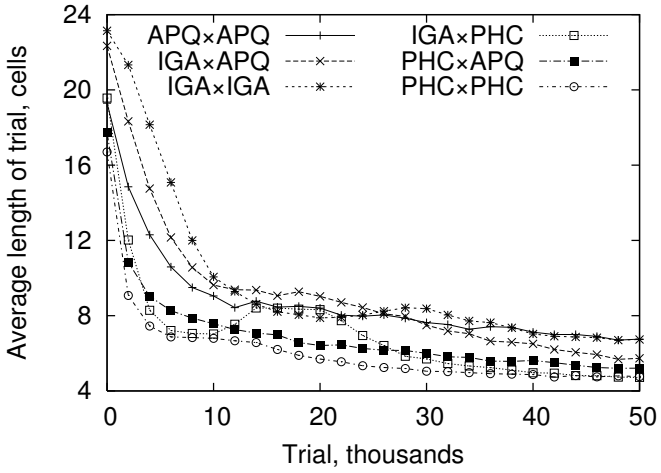


Fig. 4. The dynamics of learning in the two-robot-predator-prey game, with a 5×5 grid. The curves show the length of a trial as a function of the trials number, where the agents use the algorithms PHC, IGA/ModIGA and APQ.

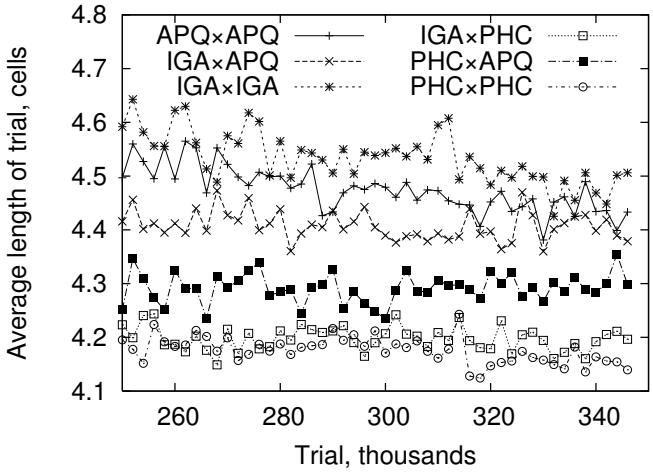


Fig. 5. The dynamics of learning of the last 100,000 trials in the two-robot-predator-prey game, with a 5×5 grid. The curves show the length of a trial as a function of the trials number, where the agents use the algorithms PHC, IGA/ModIGA and APQ.

of average trial length), the results are inverse. All the algorithm pairs with a participation of PHC (PHC×PHC, PHC×APQ and IGA×PHC) behaved better than those without PHC during the last $100,000$ learning trials (Figure 5). We explain this by the ability of PHC to learn mixed strategies, which can bring better solutions in adversarial games than pure strategies can do. For the same

reasons, APQ cannot perform better than PHC in this case. But, surprisingly for us, IGA×IGA case demonstrated the longest average trial length at the end of the learning, which is somewhat unexpected, since its convergence properties are the same as for the PHC algorithm. This fact is remaining for further investigation.

Finally, we measured the average running time of all experiments (Table 1). As expected, the PHC algorithm was the fastest in terms of calculation time (it is, in fact, the simplest in terms of the amount of calculations required at each iteration). APQ was, as expected, the slowest among all algorithms in both competition and coordination games, since at each iteration it performs a computationally hard operation of the opponent strategy estimation.

Table 1. Effective running time in different games, in seconds.

Game	PHC×PHC	PHC×IGA	PHC×APQ	IGA×IGA	IGA×APQ	APQ×APQ
Coordination	69	98	113	128	117	153
Adversarial	86	121	147	171	280	218

5 Conclusion and Future Work

In this work we compared different multiagent learning algorithms in play in two different stochastic games, a coordination game and an adversarial game. To do that, we extended Infinitesimal Gradient Ascent algorithm [4] to the case where the environment has multiple states and the agents can execute more then two different actions. The other two algorithms, namely Policy Hill Climbing [2] and Adaptive Play Q-learning [3] have already been adapted to the stochastic game setting by their respective authors. These algorithms was proven to converge to an equilibrium in self-play in the repeated matrix games, but, to our knowledge, they were never compared with each other in the case of stochastic games. This encouraged us to do this research. The goals we aimed were to investigate these algorithms in detail and to make a preliminary conclusion about their performance in stochastic games when playing against each other.

The first important observation, which we noted as a result of our experiments, is that these algorithms converge in play against each other, which was not observed and theoretically proved before. The second observation is the different quality of the solutions found by the different algorithm pairs.

In terms of execution time, we observed that the PHC algorithm required less time to get a decision in each state and, thus, it converged more quickly in the examples we used in this work. On the other hand, in the cooperation game the algorithms, which were able to observe the actions of their opponents (i.e., AQP and IGA) learned better solutions in terms of the average trajectory length, than PHC which had not such ability.

In our future work we would like to focus our attention to the finding of the formal convergence properties of these algorithms when used one against other. Also, we would extend our experiments to the more complex and unpredictable environments and to the algorithms using the learning principles other then adaptivity to the opponent's current policy, such as Hyper-Q [10] and some non-stationary algorithms such as [14, 15].

References

1. Littman, M.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the Eleventh International Conference on Machine Learning (ICML'94), New Brunswick, NJ, Morgan Kaufmann (1994) 157–163
2. Bowling, M., Veloso, M.: Multiagent learning using a variable learning rate. Artificial Intelligence **136**(2) (2002) 215–250
3. Gies, O., Chaib-draa, B.: Apprentissage de la coordination multiagent : une méthode basée sur le Q-learning par jeu adaptatif. Revue d'Intelligence Artificielle **20(2-3)** (2006) 385–412
4. Singh, S., Kearns, M., Mansour, Y.: Nash convergence of gradient dynamics in general-sum games. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI'94), San Francisco, CA, Morgan Kaufman (1994) 541–548
5. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of AAAI'98, Menlo Park, CA, AAAI Press (1998)
6. Hu, J., Wellman, P.: Multiagent reinforcement learning: Theoretical framework and an algorithm. In: Proceedings of ICML'98, San Francisco, CA, Morgan Kaufmann (1998) 242–250
7. Hu, J., Wellman, M.: Nash Q-learning for general-sum stochastic games. Journal of Machine Learning Research **4** (2003) 1039–1069
8. Littman, M.: Friend-or-foe Q-learning in general-sum games. In: Proceedings of ICML'01, San Francisco, CA (2001) Morgan Kaufman
9. Chang, Y., Kaelbling, L.: Playing is believing: The role of beliefs in multi-agent learning. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS'01), Canada (2001)
10. Tesauro, G.: Extending Q-learning to general adaptive multi-agent systems. In Thrun, S., Saul, L., Scholkopf, B., eds.: Advances in Neural Information Processing Systems. Volume 16., Cambridge, MA, MIT Press (2004)
11. Burkov, A., Chaib-draa, B.: Effective learning in adaptive dynamic systems. In: Proceedings of the AAAI 2007 Spring Symposium on Decision Theoretic and Game Theoretic Agents (GTDT'07), Stanford, California (2007) To appear.
12. Young, H.: The evolution of conventions. Econometrica **61(1)** (1993) 57–84
13. Watkins, C., Dayan, P.: Q-learning. Machine Learning **8(3)** (1992) 279–292
14. Powers, R., Shoham, Y.: New criteria and a new algorithm for learning in multi-agent systems. In Saul, L.K., Weiss, Y., Bottou, L., eds.: Advances in Neural Information Processing Systems. Volume 17., MIT Press (2005)
15. Powers, R., Shoham, Y.: Learning against opponents with bounded memory. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05). (2005)