Multi-agent Learning Experiments on Repeated Matrix Games

Bruno Bouzy BRUNO.BOUZY@PARISDESCARTES.FR
Université Paris Descartes, 45 rue des saints-pères 75006 Paris, FRANCE

Marc Métivier MARC.METIVIER@PARISDESCARTES.FR
Université Paris Descartes, 45 rue des saints-pères 75006 Paris, FRANCE

Abstract

This paper experimentally evaluates multi-agent learning algorithms playing repeated matrix games to maximize their cumulative return. Previous works assessed that Q-learning surpassed Nash-based multi-agent learning algorithms. Based on all-against-all repeated matrix game tournaments, this paper updates the state of the art of multi-agent learning experiments. In a first stage, it shows that M-Qubed, S and bandit-based algorithms such as UCB are the best algorithms on general-sum games, Exp3 being the best on cooperative games and zero-sum games. In a second stage, our experiments show that two features - forgetting the far past, and using recent history with states - improve the learning algorithms. Finally, the best algorithms are two new algorithms, Q-learning and UCB enhanced with the two features, and M-Qubed.

1. Introduction

Multi-agent learning (MAL) is a recent field which asks how to learn when there are several learners in the environment. It inherits from reinforcement learning, game theory, and from bandit algorithms. Works in MAL can be divided into 5 agendas (Shoham et al., 2007): computational (compute equilibria), descriptive (describe human learners), normative (understand equilibria arising between learners with game theory tools), prescriptive cooperative (learn with communication, distributed problem solving), and prescriptive non cooperative (learn without communication). The background of this paper is the prescriptive non cooperative agenda in which the goal of an agent is to maximize its cumulative return over time.

Many MAL algorithms already exist (Littman, 1994; 2001; Stimpson & Goodrich, 2003; Conitzer & Sandholm, 2003; Crandall & Goodrich, 2005; Powers & Shoham, 2004; 2005). Their performances have been assessed on well-chosen Repeated Matrix Games (RMG) highlighting them, sometimes by using GAMUT (Nudelman et al., 2004), a tool offering a way to generate instances of specific MG. Few works exist to compare the various algorithms systematically on a general and common test-bed RMG. First, (Zawadzki, 2005) compares several algorithms including Nash-equilibrium-oriented algorithms (Conitzer & Sandholm, 2003; Littman, 1994; Singh et al., 2000; Banerjee & Peng, 2004), Q-learning (QL) (Watkins & Dayan, 1992), and portfolio algorithms (Powers & Shoham, 2004; 2005) on instances of MG generated with GAMUT, concluding on the superiority of QL. This result confirms that, against various and unknown players, using the concept of Nash Equilibrium is worse than simply being a best response (BR) to other players. Second, (Airiau et al., 2007) compares several algorithms, including some learning ones, and concludes on the superiority of Fictitious Play (FP) (Brown, 1951).

The objective of the paper is many-fold: first evaluate MAL algorithms including the best algorithms of (Zawadzki, 2005) and (Airiau et al., 2007) (i.e. QL and FP), promising algorithms not yet evaluated like (Stimpson & Goodrich, 2003; Crandall & Goodrich, 2005) and bandit-based algorithms (Auer et al., 2002a), and other reference algorithms. Second, draw conclusions, identify the relevant features, and assess the enhanced versions of the previous algorithms by including the features. Finally, the best resulting algorithms, enhanced or not, are tested in several classical games.

Section 2 presents the MAL algorithms implemented,

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

and evaluated in this work. Section 3 describes the framework used to select good algorithms, and shows the first results obtained in order to identify efficient MAL features. Section 4 presents the results obtained by the previous algorithms enhanced, or not, by those features. Before conclusion, section 5 shows the results obtained by our best players on some well-known MG.

2. The players

This section lists the players considered in our evaluation framework. On the one hand, since they were considered inferior to QL by (Zawadzki, 2005), and difficult to implement, we did not include MetaStrategy (Powers & Shoham, 2004), Awesome (Conitzer & Sandholm, 2003), Minimax-Q (Littman, 1994), RV (Banerjee & Peng, 2004) and GIGA (Bowling, 2003). PCM (Powers et al., 2007), Manipulator (Powers & Shoham, 2005) and FFQ (Littman, 2001) being extensions of Metastrategy and Minimax-Q, requiring having classes of opponents, we did not insert them in our league. On the other hand, we included learning algorithms, simple to implement, issued from game theory, multi-agent learning or bandit-based approach. Besides, we included reference non learning algorithms: Minimax, Optimistic, Bully and Random. Each algorithm has specific parameters whose values are given below, and tuned experimentally (see section 3.5).

JR. (Robinson, 1951) JR relies on the external regret minimization: for every actions, taken or not, JR saves the cumulative return. JR selects the action with the best cumulative return.

Minimax. Minimax guarantees reaching at least the minimax value on every game, but it cannot find Pareto states when playing with friendly agents on cooperative games. Minimax is a synonym of Determined in (Zawadzki, 2005), or Nash in (Airiau et al., 2007).

Fictitious Play (FP). FP (Brown, 1951) computes empirical estimations of opponents' actions, and plays each step the BR to that empirical distribution. It was ranked first by (Airiau et al., 2007).

Q-learning (QL). Q-learning (Watkins & Dayan, 1992) is a BR player. It is the best algorithm in (Zawadzki, 2005). QL has two parameters: the discount factor γ and the learning rate α. We set γ to 0.95 and define α as $1/t$ with the t timestep of the current game. Moreover, we used ϵ-greedy as exploration scheme with ϵ defined as $1/\sqrt{t/n_a}$ with t the timestep and n_a the number of actions avalaible to the player. In section 4, QL is tested when using a fixed-width windows, i.e. with a fixed learning rate. α is then fixed to 0.01.

S. The S algorithm (Stimpson & Goodrich, 2003) converges to Pareto optimal states in social dilemma games. At each step, S updates an aspiration value. Whenever its return is superior to the aspiration value, S keeps the same action, otherwise it changes randomly to another action. S has two parameters: the initial aspiration level α and the learning rate λ. We used the following values: $\alpha = 12$, $\lambda = 0.99$.

M-Qubed (M3). M-Qubed (Crandall & Goodrich, 2005) is an offspring of S and QL. M-Qubed is designed to either play the minimax policy, or a max policy. M-Qubed plays greedily over Q-values when superior to a minimax value, and minimax otherwise. M-Qubed has three parameters: the discount factor γ, the learning rate α, and a parameter λ used to update the policy selection probabilities. We used the following values: $\gamma = 0.95$, $\alpha = 0.1$, and $\lambda = 0.01$.

UCB. UCB (Auer et al., 2002a) is a regret minimization algorithm for the multi-armed bandit (MAB) problem. It was never evaluated in RMG before. UCB selects the action j with the highest value $U(j) = \bar{x_j} + \sqrt{C\ln(t)/n_j}$, where $\bar{x_j}$ is the average return obtained when action j is played, t is the timestep and n_j is the number of time action j has been played. C is a parameter set to 100.

Exp3. Exp3 (Auer et al., 2002b) is designed to minimize the regret on the MAB problem against an evil player minimizing the returns of Exp3. The urgency of an action is the sum of two terms. The first term is an exponential term in the success of the action for exploitation, and the second term is a constant term for exploration. Exp3 has one parameter, γ, set to 0.001 in our experiments.

HMC. (Hart & Mas-Colell, 2000) describes a simple method, called HMC, converging to correlated equilibria. HMC is based on internal regret minimisation. HMC has one parameter, μ, set to 3.

Bully. Since a good MAL algorithm is necessarily a BR to a stationary environment, Bully (Stone & Littman, 2001), which exploits the BR learners, is an interesting non learning player. Bully chooses the action that maximizes its own return assuming its opponent is a BR learner.

Optimistic. The optimistic player basically chooses the action with the best return assuming the other player is friendly. In self-play on cooperative games with exactly one optima, it plays optimally.

Random. This is the player that plays randomly according to the uniform probability.

Moreover, each algorithm has its own information requirements: the whole matrix (Minimax, Optimistic and Bully), the virtual returns (i.e. the returns that would have been obtained with all the actions) (JR and HMC), the opponent actions and the actual return (FP), the actual return (QL, S, UCB, Exp3), or nothing (Random). M3 needs the actual return but also the minimax value, consequently the matrix. All the experiments let all the players have its requirements satisfied.

3. Evaluation framework and first assessments

This section presents the tournament framework (subsection 3.1), a first set of results with a fixed number of RMG (subsection 3.2), the elimination mechanism (subsection 3.3), a second set of results on general-sum games with the elimination mechanism (subsection 3.4), the importance of tuning parameters (subsection 3.5), a third set of results on cooperative games and competitive games (subsection 3.6), and a fourth set of results depending on the number of actions (subsection 3.7).

3.1. The tournament framework

Our evaluation framework is based on experiments. An experiment is a set of N RMG tournaments. A RMG is a game consisting in playing the same MG repetitively, and cumulating the outcomes obtained at each repetition. One experiment consists in drawing N matrix games at random with return values in the interval $[-9, +9]$ and execute N tournaments, one for each MG generated, in which each player meets all players in a RMG match. We call this an all-against-all tournament. When two players play a RMG match, they have to maximize their cumulative return. At the end of a tournament, a player has a cumulative return over all his matches. In an experiment, the output of an algorithm is its cumulative return over repetitions and matrix games played. The output of an experiment is a ranking. To some extents, our framework is similar to (Zawadzki, 2005; Airiau et al., 2007). The differences are that (Zawadzki, 2005) draws the games in specific categories with GAMUT, and (Airiau et al., 2007) selected its games with (Brams, 1994). We think our game selection method is not biased with human interpretation.

3.2. First results

As first results, we performed experiments in 3x3 general-sum matrix games. The first experiment involves all the players presented in section 2 in 100 matrix games drawn randomly. This experiment thus permits to see the evolution of the players ranking according to the number of steps allowed in the games.

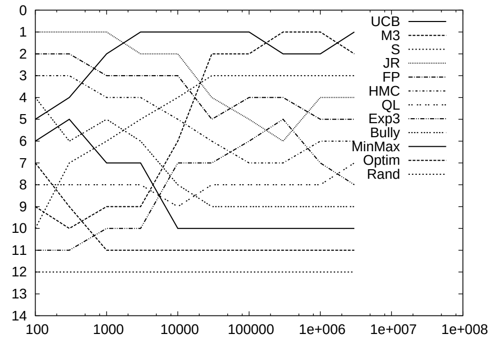

Figure 1. Exp. 1: ranking evolution according to the number of steps played in games (logscale). The key is ordered according to the final ranking.

Figure 1 shows the ranking evolution when the number of steps increases from 100 repetitions up to 3,000,000 repetitions. Figure 1 highlights the number of steps each algorithm needs to stabilize its best strategy. FP, JR and HMC, which take the first three ranks in 100-step RMG, converge faster than the other algorithms but their ranks decrease as the step number increases. In 1,000,000-step RMG, these players are surpassed by UCB, M3 and S. UCB is ranked first in 10,000-step RMG, but it is surpassed by M3 in 300,000-step RMG, and it is ranked first in 1,000,000-step RMG. The M3's rank is improved as the number of steps increases. Having states corresponding to joint actions may account for this result. S algorithm is very simple, and it improves its ranks as the number of steps increases until 30,000 steps and then stays in third rank until the end. Exp3 improves its rank until being 5th when 300,000 steps are played but goes down to 8th rank at the end of the experiment. QL is ranked 8th in 100-step RMG, behind Bully and Optimistic. When the steps number increases, its rank remains almost stable because it surpasses non-adaptive players while it is surpassed by M3 and S. Non-adaptive players such as Minimax, Bully, Optim and Random obtain their best rank in 100-step RMG, which is normal. Random is the worst player, behind Optim, Minimax and Bully which is the best non adaptive player. It is worth noting that (Airiau et al., 2007) used few hundred repetitions only. Figure 1 clearly shows that many changes occur after 1,000 repetitions. (Zawadzki, 2005) used 10,000 repetitions, which is still unsufficient according to our experiment. Besides, this first experiment underlines that QL and FP, ranked first by (Zawadzki, 2005) and (Airiau et al., 2007), are far from the top-ranking programs, UCB, M3 and S. This is caused by our set of players, very different from Zawadzki's one.

3.3. The elimination mechanism

The performances of an algorithm may be related with some opponents' performances. The ranking of a given algorithm depends on the presence, or absence, of other given players. For instance, if algorithm A succeeds well against algorithm C, and if algorithm B plays poorly against algorithm C. Then, an all-against-all tournament with A, B and C, favours A against B while a tournament with A, B and not C, favours B against A.

To illustrate that point, figure 2 presents new rankings of the first results based on successive eliminations of the last players. In that figure, a ranking is obtained by performing several elimination stages. The players are first ranked according their cumulative returns. The last player is then eliminated and all returns obtained against him are removed from the cumulative returns of the other players. These new cumulative returns enable to obtain a new ranking, and to eliminate a new last player, and so on. Finally, each ranking observed in figure 2 is computed as the opposite of the players elimination order.

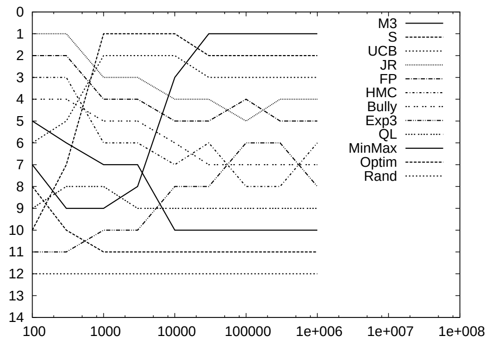

Figure 2. Exp. 1: rankings based on eliminations. The key is ordered according to the final ranking.

Figure 2 shows more stable rankings than figure 1. Passed 30,000 steps most ranks are well defined. M3 has the first rank, followed by S and UCB. In the next ranks, we may identify two groups of players. The first one is composed of JR and FP, sharing the 4th and 5th ranks. The second one concerns HMC, Bully and Exp3, sharing the 6th, 7th and 8th ranks. Then we find QL in 9th rank, followed by Minimax, Optim and Random.

Taking into account these results, we introduced an *elimination mechanism* in our framework. It consists in launching an experiment with a large number of players and removing the worst player when his cumulative return is significantly behind the before-last player or if the global ranking does not change during a long time. When a player is removed, all the returns obtained against him from the beginning of the experiment are suppressed from the cumulative returns of other agents. This removing process is applied at the end of each RMG and is continued until only one player remains.

In practice, at the end of a RMG tournament, if the difference between the cumulative returns of the two worst players is up than $600/\sqrt{n_T}$ with n_T the number of tournaments performed, then the worst player is removed. The worst one is also removed if the global ranking has not changed during $100 \times {n_p}^2/(n_p-1)^2$ tournaments, with n_p the current number of players, since the last elimination.

Beyond the relative assessment of each algorithm, the first experiment informs us that at least 30,000 steps are necessary to have most ranks well defined when the ranking is based on elimination. We thus fixed the number of steps at 100,000 in the most part of the experiments, except for experiments assessing algorithms using states as joint actions in which we used 1,000,000 steps.

3.4. Experiments in general-sum games

The second experiment has been performed with 3x3 action general-sum matrix games too. It applies the elimination mechanism presented in section 3.3.

Figure 3 shows the mean return evolution when the number of RMG increases up to 231 which is the number of RMG necessary to eliminate all players except the first one. Table 1 shows the final ranking. One may observe that, roughly, the better the rank, the greater the mean return. However, the improvement of the mean returns is not always increasing. For instance, UCB has a greater mean return than S, and S is ranked before UCB. Actually, before its elimination, UCB had a smaller mean return than S, but after its elimination, the confrontations against UCB are not taken into account, and the S mean return decreases.

This elimination experiment being performed with

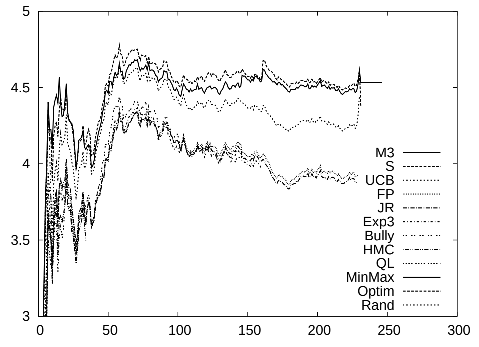

Figure 3. Exp. 2: Evolution of mean returns over time. The key is ordered according to the final ranking.

Rank	Player	Mean return	#Games
1	M3	4.532	-
2	S	4.381	231
3	UCB	4.450	230
4	FP	3.926	229
5	JR	3.878	228
6	Exp3	3.969	160
7	Bully	4.032	145
8	HMC	3.497	34
9	QL	3.328	9
10	Minimax	0.9181	3
11	Random	-1.796	2
12	Optimistic	-3.304	1

Table 1. Exp. 2: final ranking with mean return and number of games played before being eliminated.

100,000-step RMG, it is worth comparing the ranking of table 1 with the ranking given by figure 2 at 100,000 steps. We observe that the ranking is the same except for Random and Optimistic which have switched their last ranks. Using the elimination mechanism brings some results conform with results observed in the first experiment.

An experiment with the elimination mechanism provides the final ranking with better correctness guarantees on the top of the ranking than an experiment without it. Indeed, an elimination only occurs when the last player is significantly behind the before-last player. In the remaining of the paper, all experiments use the elimination mechanism.

3.5. About players' parameters

Several algorithms have specific parameters. To set them we developed a method based on our tournaments framework consisting in using several players of a same algorithm but with different parameter settings. This enables to determine which value intervals give the best performances for each algorithm in such RMG tournament contexts.

For instance, Q-learning algorithm uses a γ parameter named *discount factor* whose value in $[0, 1]$ is usually set near to 1. We created six Q-learner players with $\gamma = 0.99,\ 0.95,\ 0.90,\ 0.80,\ 0.60,\ 0.20$ and launched an experiment with these 6 copies of Q-learning and the other players. The results then showed that Q-learning had better cumulative returns with $\gamma = 0.95$. As a result, we set $\gamma = 0.95$ in all the experiments presented in this paper.

3.6. Experiments in cooperative and competitive games

Matrix games can be divided in three classes: cooperative (the two players receive same payoffs), competitive (the two players receive opposed payoffs) or general (the remaining cases). In this section, we present the results of an experiment involving cooperative games only, and another one involving competitive games only.

Cooperative games Tables 2 shows the ranking of our players on cooperative RMG. It underlines several differences with the ranking obtained on general-sum RMG. The main information is the first place occupied by Exp3 (Exp3 was ranked 6th on general-sum RMG). This fact shows that an algorithm may work well on a specific class of games, and less well on other games. Bully and Optimistic perform well on cooperative games. Optimistic is non-adaptive and designed to be optimal in self-play. However, Optimistic is ranked 5th only, which shows that against non optimal players, being optimal and non-adaptive is not good. Bully performs better on cooperative games than on general-sum games. This can be explained by the fact that, for our learners, being a best response on cooperative games is easier than on general-sum games. Consequently, Bully has more space to show its strength in cooperative games than he has on general-sum games. Minimax, not designed for cooperative games, is weak at these games. Among the set of learners, M3, JR, HMC and QL remain stable. UCB, S and FP do not succeed on cooperative games.

Competitive games Table 3 shows the ranking on competitive RMG. It underlines several differences with the ranking obtained on general-sum RMG as well. The main information is once more the first place occupied by Exp3. Exp3 outperforms the other learn-

Rank	Player	Mean return	#Games
1	Exp3	7.340	-
2	M3	7.318	5848
3	Bully	7.388	5847
4	JR	7.312	2868
5	Optimistic	7.352	2867
6	S	7.343	2190
7	HMC	7.309	875
8	FP	7.311	341
9	UCB	7.272	24
10	QL	7.111	23
11	Minimax	4.342	2
12	Random	0.319	1

Table 2. Cooperative games: final ranking with mean returns and number of games played before elimination.

Rank	Player	Mean return	#Games
1	Exp3	0.01143	-
2	M3	-0.01143	1723
3	Minimax	-0.06883	1695
4	JR	-0.2171	1694
5	FP	-0.1781	1693
6	S	-0.2921	1017
7	UCB	-0.1808	1016
8	QL	-0.218	29
9	HMC	-1.035	4
10	Bully	-1.442	3
11	Random	-1.482	2
12	Optimistic	-2.705	1

Table 3. Competitive games: final ranking with mean return and number of games played before elimination.

ers in cooperative and competitive games while he is ranked 6th only in general-sum games. In competitive games, Minimax is good because it is designed to this aim. Among the learners, M3, FP, JR, HMC, QL are stable. UCB and S loose several ranks on competitive games. Bully and Optim are not designed to play well in competitive games.

Finally, it is worth noting that UCB and S show the reverse property of the feature of Exp3: they are bad players in both cooperative games and competitive games, but they rank very well in general-sum games. At the light of these experiments, we may say that Exp3 is a good specific game player, and that S and UCB are good general MG players.

3.7. Experiments with different number of actions

In this section, we test the performances of the players in general-sum games involving number of actions set to 2, 3, 5, 10 or 30. The maximum number of steps is set to 1,000,000 in order to give more learning steps in games involving many actions.

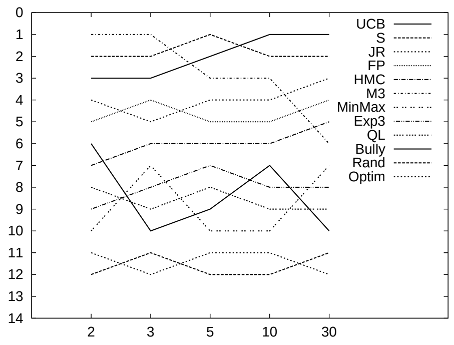

Figure 4. Evolution of mean returns according to the number of actions. The key is ordered according to the final ranking.

Figure 4 shows the effect of increasing the number of elementary actions in the RMG. M3, using states, needs more steps to learn well with many actions, this is the explanation of its low ranking with 5 actions and more. UCB, and S stay at the top of the ranking when the number of actions increases. FP and JR also remain stable when the number of actions increases. In 30 actions experiments, we can observe that FP, JR and HMC all take advantage of the weakness of M3 and get one place.

4. Experiments with enhanced players

This section presents results obtained by players enhanced with two features: the fixed-width window feature, and the state feature.

4.1. Fixed-width window

Given that the opponent's behavior may change at every moments, it may be important to be able to forget the past further than a threshold. Among the players, S and M3 use such a window mechanism.

This mechanism consists in updating the mean values with a constant rate $\alpha = 0.01$. This way, the old returns are forgotten. We selected three learners without the enhancement: QL, UCB and JR, and we added the window mechanism giving new players: QLwin, UCBwin, and JRwin respectively. We removed Exp3, FP and HMC from the league, and we launched the experiment. Table 4 shows the final ranking. Each learner without the window mechanism is surpassed by the corresponding learner with it, which shows the significance of the window mechanism.

Rank	Player	Mean return	#Games
1	M3	5.115	-
2	UCBwin	4.862	54
3	S	4.854	53
4	UCB	4.739	52
5	QLwin	4.481	51
6	JRwin	4.334	50
7	JR	4.219	29
8	QL	4.755	16
9	Bully	1.684	5
10	Minimax	1.669	3
11	Random	-1.766	2
12	Optimistic	-3.298	1

Table 4. Fixed-width window experiment: final ranking with mean returns and number of games played before elimination.

Rank	Player	Mean return	#Games
1	QLw+s	4.194	-
2	M3	4.053	18
3	UCBw+s	3.868	17
4	JRw+s	3.923	16
5	Minimax	2.922	12
6	UCBwin	2.615	11
7	S	3.337	10
8	JRwin	3.637	9
9	QLwin	3.101	6
10	Bully	2.091	3
11	Random	-1.465	2
12	Optimistic	-3.309	1

Table 5. History state experiment: final ranking with mean return and number of games played before elimination.

4.2. States and history

M3 uses states, each state corresponding to the last joint action performed (Sandholm & Crites, 1996), and compute state-action values. To take into account the past history, we define states corresponding to the last H joint actions performed. To highlight the improvement brought by states, we adapted QLwin, UCBwin and JRwin to use states yielding QLw+s, UCBw+s and JRw+s, and set $H = 1$. Table 5 shows the final ranking. All the learners without the state feature (QLwin, JRwin, UCBwin) are surpassed by all the learners with it (QLw+s, JRw+s, UCBw+s), which clearly shows the significance of this feature. In this experiment, we used 1,000,000 steps instead of 100,000 steps.

5. Experiments in specific matrix games

In this section, we test the algorithms having the best results in the previous experiments: M3, S, QLw+s, UCBw+s and JRw+s on very classical MG selected from the literature: the penalty game ($k = -100$), the climbing game, a coordination game (Claus & Boutilier, 1998), and the Prisoner Dilemma. Table 6 shows the results obtained in self-play. It is worth noting that QLw+s does not solve these games well, as IL and JAL players (Claus & Boutilier, 1998) which were QL based players. JRw+s does not solve these games well neither. UCBw+s does not solve the penalty game correctly but solves the other games well. M3 and S correctly solve the four games. We see that almost all our learners, previously tuned on a large set of MG drawn randomly, play more or less well on very specific MG extracted from the literature.

Game	+	-	M3	S	UCB	QL	JR
Pen.	10	2	10.0	10.0	5.9	7.9	9.3
Clim.	11	5	11.0	11.0	11.0	10.1	9.0
Coor.	4	2	4.0	4.0	4.0	3.4	3.0
PDil.	3	2	3.0	3.0	3.0	2.5	2.0

Table 6. Self-play results of MAL algorithms on specific games. The cells show the average return obtained by an algorithm on a RMG with 100,000 repetitions, and averaged over 10 runs. The + (resp. -) column indicates the maximal (resp minimal) average return that can be obtained.

6. Conclusion

The main contribution of this paper is two-fold. First, our experiments give rankings of existing MAL algorithms playing RMG on a wide set of MG. M3, S, UCB are assessed as the best algorithms in this context. This result updates the previous ones significantly (Zawadzki, 2005; Airiau et al., 2007). Second, our experiments show the crucial importance of two features: the fixed-width window feature, and the state feature. Adding such a feature into a MAL algorithm greatly improves its rank. We never observed a MAL algorithm becoming worst with the feature than the algorithm without.

Two other contributions are worth mentioning. First, our results highlight the good performances of three algorithms, QLw+s, M3 and UCBw+s, among which UCBw+s is a new promising bandit-based algorithm. Second, a remarkable result is the great effectiveness of Exp3 in competitive and cooperative settings.

In the future, this work may be developed through several directions. A promising concept to be studied for the MAL prescriptive agenda is the one of hedging or expert algorithms (Chang & Kaelbling, 2005). Another interesting direction is exploring why Exp3 is the best MAL player on cooperative games, and on competitive games as well, but not on general-sum games, and to exploit this fact to design a new MAL algorithm.

References

Airiau, S., Saha, S., and Sen, S. Evolutionary tournament-based comparison of learning and non-learning algorithm for iterated games. *Journal of Artificial Societies and Social Simulation*, 10(37), 2007.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002a.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R.E. The non stochastic multi-armed bandit problem. *SIAM Journal*, 32(1):48–77, 2002b.

Banerjee, B. and Peng, J. Performance bounded reinforcement learning in strategic interactions. In *AAAI*, pp. 2–7, 2004.

Bowling, M. *Multiagent Learning in the Presence of Agents with Limitations*. PhD thesis, CMU, 2003.

Brams, S. J. *Theory of Moves*. Cambridge University Press, 1994.

Brown, G.W. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, pp. 374–376, 1951.

Chang, Y.H. and Kaelbling, L.P. Hedged learning: regret-minimization with learning experts. In *ICML*, pp. 121–128, 2005.

Claus, C. and Boutilier, C. The dynamics of reinforcement learning in cooperative multi-agent systems. In *AAAI*, pp. 746–752, 1998.

Conitzer, V. and Sandholm, T. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *ICML*, pp. 83–90, 2003.

Crandall, J.W. and Goodrich, M.A. Learning to compete, compromise, and cooperate in repeated general-sum games. In *ICML*, pp. 161–168, 2005.

Hart, S. and Mas-Colell, A. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, September 2000.

Littman, M. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, pp. 157–163, 1994.

Littman, M. Friend-or-foe Q-learning in general-sum games. In *ICML*, 2001.

Nudelman, E., Wortman, J., Shoham, Y., and Leyton-Brown, K. Run the gamut: a comprehensive approach to evaluating game-theoretic algorithms. In *AMAAS*, 2004.

Powers, R. and Shoham, Y. New criteria and a new algorithm for learning in multiagent systems. In *AMAAS*, 2004.

Powers, R. and Shoham, Y. Learning against opponents with bounded memory. In *IJCAI*, 2005.

Powers, R., Shoham, Y., and Vu, T. A general criterion and an algorithmic framework for learning in multi-agent systems. *Machine Learning*, 67(1-2):45–76, 2007.

Robinson, J. An iterative method of solving a game. *Annals of Mathematics*, 44:296–301, 1951.

Sandholm, T. and Crites, R. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37(1-2):147–166, 1996.

Shoham, Y., Powers, R., and Grenager, T. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171:365–377, 2007.

Singh, S., Kearns, M., and Mansour, Y. Nash convergence of gradient dynamics in general-sum games. In *UAI*, pp. 541–548, 2000.

Stimpson, J.L. and Goodrich, M.A. Learning to cooperate in a social dilemma: a satisficing approach to bargaining. In *ICML*, 2003.

Stone, P. and Littman, M. Implicit negotiation in repeated games. In Meyer, John-Jules and Tambe, Milind (eds.), *ATAL*, pp. 96–105, 2001.

Watkins, C. and Dayan, P. Q-learning. *Machine Learning*, 8:279–292, 1992.

Zawadzki, E. Multiagent learning and empirical methods. Master's thesis, University of British Columbia, 2005.