Leader Strategies in Repeated Games

An investigation of advantages, weaknesses and improvements

Giliam de Carpentier (1056824)
Richard Noorlandt (1069764)
July 2006

Delft University of Technology

Faculty of Electrical Engineering,
Mathematics and Computer Science

Abstract

Against an unchanging opponent, best response strategies can provide a robust and natural means for maximizing an agent's payoff in simple repeated multi-agent games. Best response agents learn from previous experience in order to determine the optimal equilibrium against this opponent's fixed behavior. However, letting two best response agents play against each other may result in suboptimal performance. Littman and Stone [1] suggest an asymmetric solution to this problem. In their paper, best response strategies, called 'followers', play against 'leader' strategies. These leader strategies carefully plan the follower's experience, indirectly influencing the follower's best response reaction. By making use of this responsive behavior, leaders are able to maximize their own payoff. In this paper we extend and improve the strategies proposed by Littman and Stone, while trying to get more insight in the behavior of leading strategies.

1. Introduction

Multi-agents systems have found many uses in recent years [2],[3],[4]. However, guaranteeing strategy optimality for complex environments is hard or even impossible. Studying smaller and simpler multi-agent problems has proven to be more promising. Some non-trivial classes of multi-agent problems can be described using repeated bimatrix games. For these problems, general-sum bimatrix games are powerful enough to describe different game types that present various difficult challenges for simultaneous agent learning and planning. For a short discussion of bimatrix games, we would like to refer to [5].

A standard approach to learning in games is to apply a general-purpose adaptive strategy like Q-learning [6], which is a form of reinforcement learning [7]. Q-learner agents base their strategy on their previous experience to optimize a best-response model. These agents are guaranteed to be optimal against fixed strategies [1], but when paired with other best response learners, suboptimal strategies can occur.

Like Littman and Stone discuss in their paper "Leading Best-Response Strategies in Repeated Games" [1], we examine the approach of pairing best response agents with leaders. Leaders selectively plan their own actions, factoring in the best response behavior of the opponent. This makes it possible for leader agents to exploit the opponent's predictability, maximizing their own payoff on the long run. Littman and Stone introduced two leader strategies. The Godfather leader strategy is based on the reciprocity of choosing an action that is beneficial for the other, retaliating the best response opponent when the implicit offer is turned down. The Bully leader strategy determines the best-response reaction to each of its own choices, and chooses the action that is most beneficial to itself. Although these proposed leader strategies look promising and can be very successful, they have several weaknesses. These weaknesses will be presented and explored in this paper. Also, suggestions are made to improve the performance of these leader strategies. The exact strategies mentioned by Littman and Stone, together with our suggested extensions, were implemented and tested against each other to get a detailed view on the achieved performance.

In section 2 we present a brief overview of the strategies introduced by Littman and Stone, describe their shortcomings and introduce our suggested improvements. In section 3 we present and explain the results from our experiments.

2. Strategies

The behavior of a single agent in a multi-agent environment is determined by a strategy. This strategy decides on the agent's action to be taken at every stage, given a limited view on its environment. The expected payoff of a strategy is highly dependent of the environment and the rich interaction between strategies of agents. Hence, no single strategy is optimal, but rather, the

collection of strategies must be a Nash equilibrium [8]. Finding this Nash equilibrium for complex games where each agent uses only incomplete information has been proved to be very challenging.

One approach is 'best response'. A best response strategy tries to maximize its payoff by exploiting any observed patterns in its previous experience. The complexity of the patterns that a best response agent is able to recognize depends on the amount of information it observes at each stage and the size and complexity of its memory. Using only its own history of actions, simple but stable agents, called independent learners, can be trained. Using more information allows more complex behavior of an agent, but might result in slower convergence or even instability. At the other extreme, if an agent is able to fully observe the rules of the system and have complete information on the strategies of all other agents, complex solvers and/or heuristics might maximize its own expected payoff. However, this may be impractical or even impossible for larger systems and more complex opponents. Like Littman and Stone, we only examine two-player games in the form of bimatrix games. Generalization can probably be made for larger systems, but this is outside the scope of this paper.

Best response independent learners (ILs) adapt to their environment by modeling the expected payoff for choosing an action, given a limited history of their previous choices. Used often in research due to its simplicity, Littman and Stone implemented two best response IL variants as Q-learners. These ILs are an example of what Littman and Stone call 'follower' strategies because of their adaptive nature. In contrast, joint action learners (JALs) use more advanced reinforcement learning techniques to learn from their own choices together with the choices made by opposing agents. Although strictly not the same, the leader strategies proposed by Littman and Stone could be classified as JALs, since they coordinate their choices given the opponent's behavior. Pairing a follower with a leader creates an asymmetric relationship in which the leader can guide the follower into conditioned behavior, maximizing the expected payoff for the leader. This asymmetric relation causes typical symmetric IL-IL game instability to be resolved.

2.1 Q-Learning

A simple, well-understood algorithm for single agent learning is Q-learning [6]. Q-values, Q(x, i), provide an estimate of the value of performing action x when in state *i*. These Q-values are updated through explorative sampling to build an estimate over multiple steps. Different formulations of the update rule are found in literature, but we follow the notation used by Littman and Stone.

$$Q(x,i) = \alpha(r + \gamma \max_{i'} Q(y,i')) + (1-\alpha)Q(x,i)$$

These estimates can be used to exploitatively choose the action with the highest expected payoff. Deciding when to explore either exploitive or non-exploitive depends on the learning policy. Together with the learning rate and the state space size, this policy influences the convergence speed.

2.1.1 ε-greedy policy

The ε-greedy policy for Q-learning either chooses a random action with probability ε or chooses the action with the highest Q-value in the current state otherwise. In other words, the exploration-exploitation ratio is defined by ε / (1-ε). This ratio is kept fixed throughout the experiment. Choosing a larger ε will result in faster obtainment of more accurate estimations, and with it, faster convergence, but has the disadvantage of having a higher probability of choosing suboptimal actions after being converged. Choosing ε too small might cause the Q-value estimation to take more steps before it becomes accurate, having a negative effect on convergence. An advantage of this policy is that in dynamically changing multi-agent environments, agents implementing the ε-greedy policy will remain adaptive to their

surroundings. This policy was chosen by Littman and Stone because it is simple, easy to understand and slow to converge, making it ideal to study it in detail. We implemented their Q0 and Q1 agents. Q0 is a stateless Q-learner. Q1 has its own previous action as its state.

2.1.2 Boltzmann policy

A popular policy is the Boltzmann policy or Boltzmann exploration[9]. Simulating the process of annealing, action x is chosen with probability

$$P(x) = \frac{\exp(\frac{Q(x)}{T})}{\sum_{i \in X} \exp(\frac{Q(i)}{T})}$$

The temperature parameter T starts off high and decreases over time, so that the initial explorative nature of an agent using this policy will slowly change into exploitive behavior. This ensures fast convergence of the Q-values at its start-up time, after which the Q-values are used more and more exploitive. However, some care must be taken not to decrease T too rapid, since this might cause the explorative phase of the agent to become too short to acquire accurate estimations of the Q-values. In practice, this policy can be used robustly in many situation, while converging much faster than when using a conservative ε-greedy policy. One disadvantage of this policy in ever changing environments is that after the explorative phase, adapting to new opponent strategies becomes increasingly difficult. Its fast convergence makes it useful for many practical purposes, but also makes it more difficult to study in detail. For the purpose of comparison, this policy has been included in our experiments too. These Q-learning agents are called B0 and B1 and are otherwise identical to the Q0 and Q1.

2.1.3 Parameter settings

As mentioned before, the performance of a Q-learning agent is for one part determined by the exploration policy. However, this is not the only factor that influences the learning rate and accuracy. The formula that is used to update the Q-values has a few parameters that determine the speed in which the Q-values are updated, namely α and γ. Both of these parameters should have a value between 0 and 1. A higher value of α results in the table with Q-values being updated more rapid. The value of γ determines the role of future interactions on the Q-value: the higher the value of γ, the more weight is assigned to results of future choices while determining the Q-values.

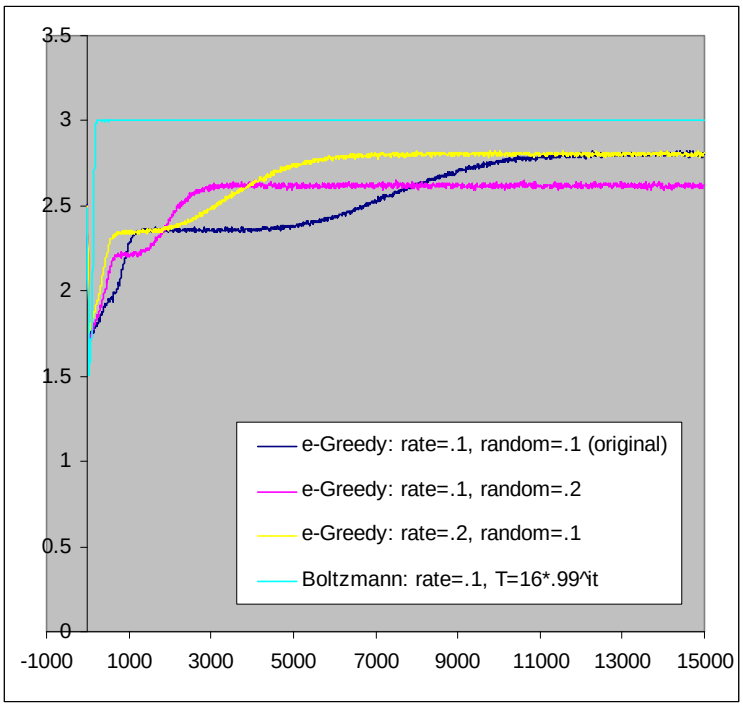

Figure 1: Q-learning, the effect of parameters and policy

That the used parameters and policy indeed have a large impact on the convergence and performance of a Q-learner can be seen in figure 1. This graph shows some performance characteristics of a Q1 (ε-greedy) and B1 (Boltzmann) learner when playing against a Godfather in the assurance game. On the y-axis we have plotted the average payoff against the number of iterations on the x-axis for a variety of α (rate) values. For the ε-greedy policy we have also included various ε (random) values. The value of γ is kept constant on 0.9. What the graph shows is that the learning rate and exploration settings have a big impact on the convergence speed. For example, an increased ε value will let the agent do more exploring, which leads to finding its optimum faster, but at the cost of the average payoff after convergence. It also becomes very clear that the Boltzmann policy converges even many times faster than a ε-greedy policy.

2.2 Bully

The Bully strategy is one of the two leader strategies introduced by Littman and Stone. This strategy assumes that the opponent will eventually play best response when the Bully consistently chooses the same action at every step. In order to maximize its own payoff, the Bully needs to know the payoff matrix of its opponent to determine what the best response choice for the opponent will be for each of its own actions. By pairing its own choice with the expected choice of its opponent, a Bully is able to figure out what its own expected payoff will be for each of its own actions. The payoff is maximized by consistently choosing for the action with the highest expected payoff.

The success of this strategy depends heavily on a game's payoff matrix structure and doesn't necessarily converge to an optimal combined strategy. Since the Bully limits itself to a subset of action-pairs, other action pairs that might be more promising are left unconsidered. This is exemplified in section 3.3 by the Prisoner's Dilemma.

2.3 Godfather

The other leader strategy proposed by Littman and Stone is the Godfather strategy. Like Bully, it chooses an action pair to play for and assumes that the opponent uses a best response strategy. However, instead of acting passively like the Bully strategy, Godfather uses active reciprocity to punish the opponent if it deviated from a certain appreciated action in the previous step. This action appreciated by the Godfather is part of an action pair, called the targetable pair. To determine this pair, the security levels of the Godfather and its opponent are determined first [1]. The action that forces the opponent to play for its security level payoff is the Godfather's way to punish its opponent. Then, the Godfather's half of the targetable pair must be the choice that allows the opponent to play for a higher payoff than its security level, or else punishment would be meaningless. Lastly, the opponent's half of this pair is determined by the Godfather to be the action that will result in the highest expected payoff for the Godfather, given the Godfather's half.

The successfulness of this strategy depends on the learning capabilities of the opponent. If an opponent is unable to comprehend the effects of being punished one step after choosing the 'wrong' action, the Godfather might be unable to condition its opponent in playing the targetable pair. In the case of a Q-learning opponent, this means that at stateless Q-learners will result in suboptimal behavior.

2.4 Extending Godfather

In this paper, we present two extensions to the Godfather strategy. The first extension is based on the observation that for some game matrices, passive Bully-like behavior might suffice to get a best response opponent to play its half of the targetable pair. The second extension attempts to use opponent prediction to punish the opponent in the same step it is most likely to make a wrong choice. By punishing the opponent in the step that a wrong choice is made instead of one step later, even stateless opponents are able to learn this lesson.

2.4.1 Best response exploit

Some types of games, like the Chicken game and Prisoner's dilemma need Godfather's punishment in order to make the opponent choose an action that wouldn't be chosen if the opponent would go for its own maximal payoff. However, it isn't always necessary to punish if the opponent chooses the wrong action. In the case of the Assurance and Deadlock game, the opponent will eventually choose for its half of the targetable pair if the Godfather plays its own half, even if punishment isn't executed when the best response opponent explores its other options. This will be true for any game for which the best response opponent will have a higher expected payoff when choosing to play its half of the targetable pair.

By not executing opponent punishment in games for which best response opponents tend to play their half of the targetable pair anyway (albeit without knowing it), the opponents have an easier lesson to learn. This means even simpler best response agents, that are unable to comprehend punishment, will be able to perform well in these games. Also, agents that are able to perform well against the standard Godfather, are likely to convergence in less steps.

2.4.2 Response prediction

For games in which there is no benefit from the best response exploit, another extension is presented here. A standard Godfather can be privately fitted with a 'predictor' of a simple best response strategy that would be unable to learn from punishment. This predictor is updated every step, exactly like an opponent with an identical best response strategy would be. When this best response strategy is implemented using a sufficiently simple Q-learning strategy where the Q-values would converge to the same numbers for each run, the response of the predictor and such an opponent are guaranteed to converge over time. This allows the Godfather to predict the action to be chosen by an opponent that implemented the identical strategy, after the predictor has been trained sufficiently.

If this is the case, a Godfather is able to punish the opponent in the current step when a wrong choice is predicted, instead of punishing it one step after it actually happened. This early punishment is an easier lesson to comprehend and can even be used against stateless Q-learners. The disadvantage of this technique is that not only the payoff matrix of the opponent needs to be known, but also the precise implementational details of the opponent that the Godfather is attempting to predict. However, if this limitation isn't problematic for a certain multi-agent environment, considerable performance gains can be achieved by this extension. To determine whether the Godfather's opponent is an opponent using the same strategy as the predictor, an assessment of the predictor's accuracy needs to be made. We implemented two different policies for comparison.

The first policy used a running average of predictor correctness R_i to get a measure between 0 and 1. This measure was then used to calculate the probability $P_i = \text{Max}(0, 2(R_i - .5))$ for acting reciprocally to the predicted action instead of the standard Godfather's behavior of acting to the opponent's previous choice. This relation has the characteristic of having zero probability for an average accuracy that is equal or less than what a random strategy would achieve.

The second policy bases its decision on whether or not the predictor made a correct prediction in the previous step. If it predicted the action of the opponent correctly in the previous step, the Godfather assumes the predictor will be correct for the current step and will act accordingly. However, if the predictor did not correctly predict the opponent's choice in the previous step, the Godfather will assume the strategy of the predictor is either not identical to the opponent's strategy, or the predictor is not trained enough. In either case, it chooses to go for the standard Godfather behavior for the current step.

2.4.3 Combining extensions

Response prediction might help improve the performance if the opponent is a best response agent. Moreover, response prediction can be used for every game type. On the other hand, the best response exploit can only be used for certain game types, but is advantageous for more types of opponent. To get the most out of our proposed extensions, the Godfather can check whether or not the game matrix is suitable for the best response exploit. If it is suitable, the best response exploit is used. Otherwise, the response predictor is used.

3 Results

In order to test the effectiveness of our extensions to the Godfather strategy we implemented a framework that runs a large number of simulations of the games being played by different agent combinations. Because the new strategies are based on the ones that are introduced by Littman and Stone, we decided to run the tests and compute the results in a similar way as described in their paper [1]. However, we did implement and test more agent types and ran more trials to get more accurate averages.

The results, as given in the following sections, were computed by running experiments that placed two agents against each other in a certain game. Each experiment consisted of 30,000 iterations. That is, each experiment is a repeated game where each agent takes 30,000 turns. For each experiment, the average payoff of the last 5,000 iterations was computed. We ran each experiment 100,000 times and measured the mean and standard deviation of this averaged payoff, as opposed to the 100 test runs used by Littman and Stone. It is worth noting that the reported results only represent the scores after the scores had time to stabilize, due to the fact that the average payoff is calculated only over the last 5,000 iterations of each experiment. We therefore only analyzed performance after the strategies of the agent pairs had stabilized, ignoring initial performance.

For the Q_0 and Q_1 learners we used identical parameters as Littman and Stone: $\gamma = 0.9$, $\alpha = 0.1$, and $\varepsilon = 0.1$ for the greedy policy. These same settings were used for the reinforcement learners that used the Boltzmann strategy, but with $T(i) = 16 * .99^i$ as its temperature at iteration i. For consistency, we will call action 0 "cooperate" and action 1 "defect" in the same way as in the Littman and Stone paper.

We ran tests for all possible agent combinations of implemented agents, including competing leader strategies. For comparison, we also added an agent type that acts randomly. Although the main goal is to see how well leaders perform against followers, it is interesting to see how leaders perform against each other as these numbers can give some insight on the overall performance and stability in the case where no assumptions can be made on the type of opponent.

General results in the form of the average payoff for each game are shown in tables 1-4. For brevity, the exact values of the standard deviations are omitted. However, high standard deviations, most likely due to instable solutions, are color-coded.

In the tables, the following abbreviations and color codes are used:
Q0: Q0 learner
Q1: Q1 learner
Bu: Bully
Gf: Godfather
Ra: Random
Gb: Godfather with best response exploit
Gp: Godfather with Q0 predictor, policy 2
Gn: Godfather with Q0 predictor, policy 1
Gu: Combination of Gb and Gp
B0: Q0 with Boltzmann policy
B1: Q1 with Boltzmann policy

Legend:
Mutual defection
Sucker
Tempted
Mutual cooperation
Unstable
Bad performance

3.1 Deadlock

The deadlock game is a simple game, consisting of the following matrices:

$$M_1=\begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix},\ M_2=\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$$

As mentioned by Littman and Stone, in this game both agents perform best if they both cooperate. A best response strategy will do this automatically, guaranteeing solid performance for most agents. As we can see in table [1], almost all combinations of agents come quite close to the maximum achievable score of 3.

		Player									
		Q0	Q1	Bu	Gf	Ra	Gb	Gp	Gn	Gu	B0	B1
Opponent	Q0	2,805	2,805	2,950	2,805	1,993	2,950	2,681	2,793	2,950	2,950	2,950
	Q1	2,805	2,805	2,950	2,805	1,993	2,950	2,681	2,793	2,950	2,950	2,950
	Bu	2,850	2,850	3,000	3,000	2,010	3,000	2,858	2,858	3,000	3,000	3,000
	Gf	2,805	2,805	3,000	3,000	1,898	3,000	1,048	2,756	3,000	3,000	3,000
	Ra	2,553	2,553	2,670	1,898	1,898	2,670	1,860	2,279	2,670	2,670	2,670
	Gb	2,850	2,850	3,000	3,000	2,010	3,000	2,857	2,858	3,000	3,000	3,000
	Gp	2,766	2,766	2,953	1,048	1,892	2,952	1,090	2,671	2,952	2,953	2,953
	Gn	2,801	2,801	2,953	2,756	1,953	2,953	2,590	2,781	2,952	2,952	2,952
	Gu	2,850	2,850	3,000	3,000	2,010	3,000	2,857	2,857	3,000	3,000	3,000
	B0	2,850	2,850	3,000	3,000	2,010	3,000	2,858	2,857	3,000	3,000	3,000
	B1	2,850	2,850	3,000	3,000	2,010	3,000	2,858	2,857	3,000	3,000	3,000

Table 1: Deadlock

There are, however, some interesting combinations of agents that result in one of the agents performing poorly. It is very clear that the random strategy overall performs a lot worse than other strategies. Also, when Gp plays against another Gp or Gf agent, both agents perform extremely poor.

Further analyzing the results, we can see that the Bully strategy is hard to beat on the long run. This is easily explained by the fact that the opponent's best response always results in cooperation. Thus, both the Bully and it's opponent have a tendency to mutual cooperation. The opponent's random exploration is the only factor that prevents both agents from reaching truly optimal results.

The Gb and Gu strategies perform as well as Bully. Because Gb will never need to execute punishment for this game, it will always play its half of the targetable pair. Random exploration by the opponent results in self punishment and explains the suboptimal performance against

agents with a stochastic element. Likewise, Gu will never need to execute punishment either and effectively becomes like Gb.

Other solid performers are the Boltzmann strategies. This is due to the fact that these learners converge very quickly, after which their choices become fixed on cooperating. Only random exploration on the opponent's side can have a small negative effect on the result. This is reflected by the somewhat lower scores when playing against stochastic strategies.

3.2 Assurance

Although its matrices are the transpose of the deadlock game, the assurance game has more complex characteristics. Optimal results can be achieved when both agents cooperate, but when one of the agents chooses to defect, it's important for the other agent to defect too. When both agents choose the same option they will always get the same payoff. If they both choose a different option, on the other hand, one of the agents will get a payoff of 0.

$$M_1 = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix}$$

The fact that this game is more difficult than deadlock can easily be seen from our test results, which show more variance than with deadlock. It's easy to see that Bu, Gb, Gu and the Boltzmann strategies are the best overall performers. Just like in Deadlock, this is the result from the fact that both agents are best off always cooperating, with deviations from an optimal score being the result of random explorations.

Opponent \ Player	Q0	Q1	Bu	Gf	Ra	Gb	Gp	Gn	Gu	B0	B1
Q0	1,561	1,683	2,850	1,343	2,239	2,850	2,711	2,109	2,850	1,940	1,927
Q1	1,683	1,952	2,850	2,805	2,017	2,850	2,765	2,401	2,850	1,850	1,916
Bu	2,950	2,950	3,000	3,000	2,670	3,000	2,952	2,953	3,000	3,000	3,000
Gf	1,343	2,805	3,000	3,000	1,898	3,000	1,048	2,732	3,000	3,000	3,000
Ra	1,947	1,915	2,010	1,898	1,898	2,010	1,888	1,925	2,010	2,000	2,000
Gb	2,950	2,950	3,000	3,000	2,670	3,000	2,952	2,952	3,000	3,000	3,000
Gp	2,635	2,676	2,857	1,048	1,828	2,857	1,117	2,538	2,857	2,858	2,857
Gn	2,107	2,374	2,858	2,732	2,084	2,857	2,617	2,683	2,858	2,857	2,790
Gu	2,950	2,950	3,000	3,000	2,670	3,000	2,952	2,953	3,000	3,000	3,000
B0	1,940	1,840	3,000	3,000	2,601	3,000	2,953	2,952	3,000	3,000	2,663
B1	1,924	1,915	2,993	3,000	2,448	2,993	2,952	2,876	2,993	2,553	2,292

Table 2: Assurance

What is different, however, is the occurrence of many unstable results when at least one of the agents uses a stochastic strategy. Especially when Q0 or Gn are involved, the results show a very large standard deviation:

	Q0	Q1	Gn
Q0	1.561 (0.821)	1.683 (0.860)	2.109 (0.865)
Q1	1.683 (0.861)	1.952 (0.875)	2.401 (0.686)
Gn	2.107 (0.857)	2.374 (0.669)	2.683 (0.238)

In the table, the standard deviation of the results is shown in parenthesis. It is obvious that the standard deviation for the shown player combinations is quite large. Where most measured results have a variance far below 0.01, these combinations are several orders of magnitude higher. This is due to the involvement of multiple random factors, whose interaction has a large influence on the outcome of the game. On one hand, mispredictions made by Gn's internal predictor can

destabilize the other agent's Q-values, while on the other hand, random exploration by the reinforcement learners affect performance in a negative way. Since bad decisions made by either agent are reflected in the Q-values of both agents (in the case of Gn inside the internal predictor), prediction errors can propagate indirectly between agents and influence future results. It is due to this interaction that eventual average payoffs may vary greatly.

An explanation for this behavior can be given by the fact that when one agent cooperates while the other defects, the cooperating agent gets a very low payoff of 0. Thus, asymmetric choices have a large impact on the results. When one of the agents uses a form of random exploration or punishment based on a stochastic predictor, these occasional asymmetric choices will immediately have a large impact on the Q-value of a Q-learning based agent. This is illustrated by figure [2], which is a plot of a Q1-Learner's Q-values during the first 8,000 iterations in a game played against Gn.

The Q1 agent's Q-values show an unstable pattern. This is due to ε-greedy exploration on the Q1 learner's side and false predictions on Gn's side. These two factors result in the agents making opposite choices, resulting in a low payoff for one of the agents. When this happens, the Q-value will drop immediately, as can be seen from the figure. Recovery from these low scores is fairly rapid, since Q1 realizes that cooperating remains the best option.

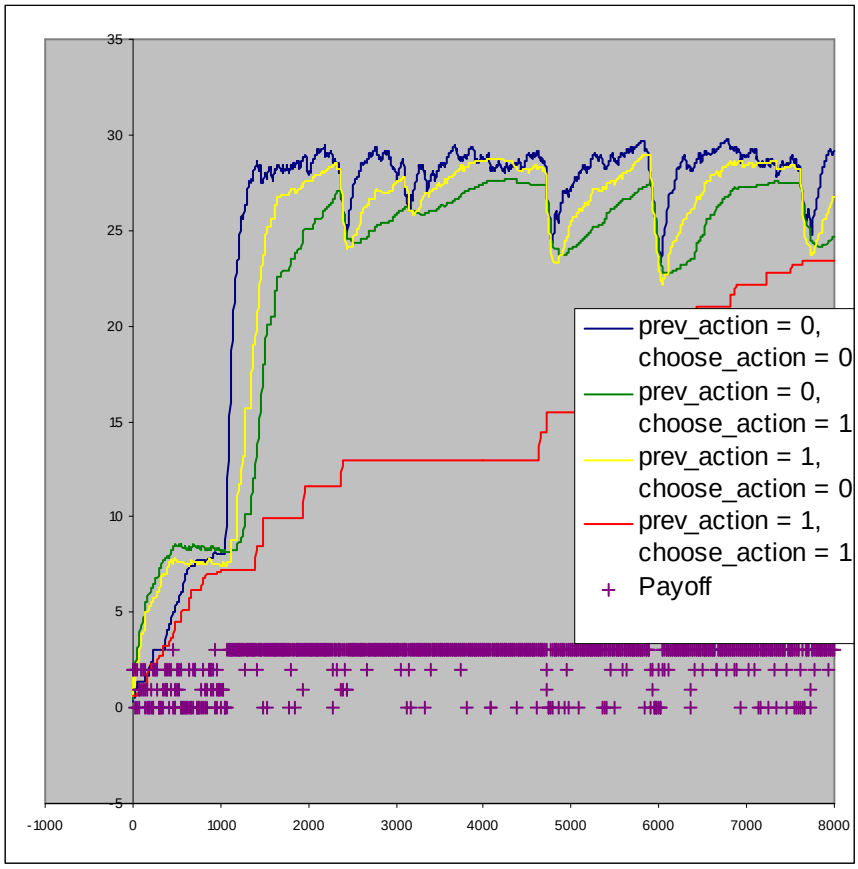

Figure 2: Q-values for Gn-Q1

Other combinations of agents that show striking results are Gp-Gf and Gp-Gp. These opponents get extremely poor results for reasons that are not immediately obvious. The explanation for this phenomenon lies in the fact that both strategies can get in a continuous cycle of mutual punishment. The problem lies in Gp's internal Q0-based predictor. Gp assumes that its prediction will be correct when the last prediction was. At some point in time, it will happen that the prediction will be inaccurate due to the predictor's stochastic element. In time it will occur that the predictor's Q-value reaches the point where it suggests that defecting will improve the payoff (due to the memoryless characteristics of Q0). If the previous prediction was correct, Gp will unjustly punish its opponent based on this false prediction. As a reaction, the Godfather opponent will punish Gn due to its betrayal. In the next turn, Gn will again predict that its opponent will defect, and since the last prediction was assumed to be correct, it will punish Gf again in return and keep the cycle alive until a new misprediction will be made and propagated.

3.3 Prisoner's Dilemma

The prisoner's dilemma is a classic problem in game theory. It is defined by the following matrices:

$$M_1 = \begin{bmatrix} 3 & 0 \\ 5 & 1 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 3 & 5 \\ 0 & 1 \end{bmatrix}$$

The main issue with the prisoner's dilemma is that although both agents are best off by always cooperating, defecting is a very attractive strategy. From the results below we can see a great difference in scores among the agent combinations. It is especially notable that the Bully strategy as proposed by Littman and Stone performs extremely poor against all agents (except Random). This is a result from Bully's assumption that its opponent will always play best response, which is defecting by default. Based on this assumption, the Bully will also choose to defect, resulting in a payoff of 1 for both agents.

Opponent \ Player	Q0	Q1	Bu	Gf	Ra	Gb	Gp	Gn	Gu	B0	B1
Q0	1,181	1,158	1,200	1,355	0,536	1,355	2,757	2,805	2,757	1,200	1,200
Q1	1,175	1,205	1,200	2,948	0,583	2,947	2,982	3,073	2,982	1,200	1,200
Bu	0,950	0,950	1,000	1,000	0,330	1,000	0,953	0,953	0,952	1,000	1,000
Gf	1,355	2,948	1,000	3,000	2,561	3,000	2,491	1,300	2,491	3,000	3,000
Ra	3,577	3,553	3,680	2,561	2,561	2,561	2,588	2,886	2,588	3,679	3,526
Gb	1,355	2,947	1,000	3,000	2,561	3,000	2,491	1,301	2,491	3,000	3,000
Gp	2,726	2,786	1,190	2,491	2,508	2,491	2,473	2,068	2,473	2,857	2,854
Gn	2,792	2,573	1,190	1,300	1,913	1,301	1,903	1,477	1,903	2,858	2,786
Gu	2,726	2,786	1,190	2,491	2,508	2,491	2,473	2,068	2,473	2,857	2,858
B0	0,950	0,950	1,000	3,000	0,331	3,000	3,095	3,095	3,095	3,000	2,413
B1	0,950	0,950	1,000	3,000	0,636	3,000	3,089	3,089	3,089	1,783	1,646

Table 3: Prisoner's Dilemma

From the results we can see that there is no obvious best strategy for the prisoner's dilemma, which is in line with prior research on the game. It is interesting to note that although the game has very complex dynamics, the simple reinforcement learners perform relatively well against all but Q-learners and Bully strategies.

Another interesting phenomenon which is related to learning speeds can be observed by comparing Gp and Gn. As Axelrod notes in [10], "The speed of response depends upon the time required to detect a given choice by the other player. The shorter this time is, the more stable cooperation can be." When we look at the predicting extensions to Godfather, it can easily be seen that Gp will be more predictable and transparent to an opponent than Gn. Since Gp bases its decision on whether or not to use its prediction only on the accuracy of last turn's prediction. This behavior can be observed relatively quick by an opponent, giving it a chance to react in a timely manner. Gn, on the other hand, bases its decisions on a much slower and more gradually changing parameter: the running average of the predictor's performance. This behavior is very obscure and difficult to detect for an opposing agent, limiting the opponent's reaction time in return. According to Axelrod, Gp should thus give it's opponents a better chance to achieve cooperation than Gn. Our results indeed seem to confirm this, as Gn's performance is much more unstable than Gp.

3.4 Chicken

Chicken represents the game where two cars are approaching each other on a highway. If none of the agents changes it's course, both will crash on each other giving a low score for both. On the other hand, if only one of the agents changes course, the one that is changing lanes gets a low payoff, and the other gets a high payoff. Essentially, both agents must try to choose the opposite action of their opponent to maximize their own performance. The matrices are given below:

$$M_1 = \begin{bmatrix} 3.0 & 1.5 \\ 3.5 & 1.0 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 3.0 & 3.5 \\ 1.5 & 1.0 \end{bmatrix}$$

When analyzing our results, we can see that the Bully as proposed by Littman and Stone performs extremely well against reinforcement learners, but can't handle other leading strategies. This again is a result from the deterministic nature of the Bully which essentially always chooses to defect, thereby limiting the other agent's payoff to a maximum of 1.5 while enforcing its own payoff to be 3.5. While playing against other leader strategies the score is disastrous due to constant mutual defection (be it because of another Bully whose standard choice is to defect, or due to punishment executed by a Godfather-like strategy).

		Player									
		Q0	Q1	Bu	Gf	Ra	Gb	Gp	Gn	Gu	B0	B1
Opponent	Q0	*2,432*	*2,479*	3,375	2,850	1,871	2,850	2,909	2,907	2,909	2,217	2,187
	Q1	*2,440*	2,867	3,375	2,948	2,004	2,948	2,966	2,978	2,966	2,777	2,246
	Bu	1,475	1,475	1,000	1,000	1,335	1,000	1,024	1,024	1,024	1,500	1,500
	Gf	2,850	2,948	1,000	3,000	2,561	3,000	2,492	2,821	2,492	3,000	3,000
	Ra	2,625	2,613	2,675	2,561	2,561	2,561	2,562	2,573	2,562	2,628	2,560
	Gb	2,850	2,948	1,000	3,000	2,561	3,000	2,492	2,821	2,492	3,000	3,000
	Gp	2,845	2,881	1,119	2,492	2,551	2,492	2,498	2,737	2,498	2,929	2,929
	Gn	2,900	2,805	1,119	2,821	2,428	2,821	2,780	2,840	2,780	2,929	2,929
	Gu	2,845	2,881	1,119	2,492	2,551	2,492	2,498	2,737	2,498	2,929	2,929
	B0	2,762	2,695	3,500	3,000	1,842	3,000	3,024	3,024	3,024	2,701	2,511
	B1	2,791	2,697	3,488	3,000	2,581	3,000	3,021	3,024	3,024	2,892	2,806

Table 4: Chicken

The Godfather and our extensions to it perform quite well against most agents except the Bully. Their good performance is due to their continuous choice for the targetable pair. The reinforcement learners quickly learn their lessons, while other Godfathers will always cooperate by also choosing their part of the targetable pair.

4 Conclusions

We have extended the Godfather strategy with many different features to improve its performance. In the paper by Littman and Stone, the only combination of agents that always settles on mutual cooperation is Gf-Q1. In other combinations, good performance could not be guaranteed. As our results have shown, the Godfather strategy can be made more robust to perform more consistently to other types of agents. The best response exploit helps Godfather to perform better or equal against Q0 agents in all games but the prisoner's dilemma. Because of the lack of memory, a Q0 learner is not capable to learn in the prisoner's dilemma and thereby continuously chooses to defect. Due to Q0's tendency to defect without the effect of self-punishment, Gb will very often have to execute punishment. Almost continuous mutual defection is the result.

Another extension that tries to make Godfather perform better against Q0 agents is the use of an internal Q0 predictor. This approach improves performance slightly in the prisoner's dilemma and chicken games. Performance is improved significantly in assurance, but here the extension cannot prevent the results from having a high variance and thus being unstable. In the deadlock game, performance against Q0 and Q1 is slightly decreased due to mispredictions in Godfather's internal learner. This phenomenon is also visible while playing against other leading strategies where the use of a Q0 predictor is greatly misplaced. The trend seems to be that stochastic strategies decrease performance while playing against a deterministic agent.

When combining the best response exploit with a Q0 predictor we get the best of both worlds. Comparing the results of Gu with Gb and Gp we can see that the score that Gu obtains is actually max(Gb, Gp), thereby indicating that combining strategies can work well in most cases. It should be noted though that every Godfather based strategy is unable to handle consistently non-cooperating agents in games where defecting can increase an agent's performance. This indeed is in line with previous research showing that an always-defect strategy is evolutionary stable in the prisoner's dilemma, making it impossible for another strategy to invade on the long term. See [11].

One big advantage of the Godfather based strategies is that they tend to perform relatively well against other cooperating leaders. This is due to the fact that in such a situation both agents keep aiming for the targetable pair, resulting in frequent mutual cooperation. The more deterministic the involved strategies are, the better the overall performance in this case. Since the basic strategy on which these agents rely is deterministic in nature, using stochastic elements only tends to disrupt the trend of mutual cooperation for a short time.

Furthermore, we have seen that all Godfather variants tend to lead reinforcement learners well. As long as an opponent is willing to adapt itself to the leading strategy the overall performance will generally be good. In these cases the stability of the follower has the most impact on the average payoff of both agents. This indeed explains why the Boltzmann strategy works better than the ε-greedy strategy: Boltzmann quickly converges to mutual cooperation, where ε-greedy occasionally keeps exploring with negative effect.

We have shown that extending the Godfather strategy can improve performance in many situations. It however remains a fact that a strategy which aims at mutual cooperation does not work well in every game/agent combination. To further improve the agents it will be necessary to implement heuristics to discover if an opponent is playing with an aggressive non-cooperative strategy, and adapt its playing style to handle this kind of situation.

5. References

[1] M. L. Littman and P. Stone. "Leading Best-Response Strategies in Repeated Games". AT&T Labs Research. 2001

[2] B. Burmeister, A. Haddadi, G.Matylis. Application of Multi-Agent Systems in Traffic and Transportation. IEE Proceedings, Vol 144, Issue 1. p. 51-60. 1997

[3] J. Ferber. Multi-Agent System: An Introduction to Distributed Artificial Intelligence. Harlow: Addison Wesley Longman. ISBN 0-201-36048-9. 1999

[4] N. R. Jennings and M. J. Wooldridge. Agent Technology: Foundations, Applications, and Markets. Springer-Verlag New York, Inc. ISBN: 3-540-63591-2. 1998

[5] K. Fukuda. Bimatrix Games.

[6] C. J. C. H. Watkins and P. Dayan. "Technical Note on Q-learning". Machine Learning 8, p 279-292. 1992

[7] C. Claus and Craig Boutilier. "The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems". Department of Computer Science, University of British Colombia. 1998

[8] J. Nash. Equilibrium Points in N-Person Games. Proceedings of NAS. 1950

[9] M. Mundhe and S. Sen. "Evaluating concurrent reinforcement learners". Department of Mathematical & Computer Sciences, University of Tulsa.

[10] R. Axelrod. *The Evolution of Cooperation.* ISBN 0465021212. 1984

[11] R. Axelrod. “The Evolution of Cooperation”. Science, 211, p. 1390-1396. 1981