Establishing Reputation Using Social Commitment in Repeated Games

Jacob W. Crandall and Michael A. Goodrich
Computer Science Department
Brigham Young University
Provo, UT 84602
crandall, mike@cs.byu.edu

Abstract

Best response algorithms have been the focus of much research in multiagent learning. However, this approach can yield undesireable results in some contexts, most notably in iterated social dilemmas. In this paper, we focus on establishing good reputations rather than finding best responses, which leads to a focus on pareto efficient solutions. We facilitate learning of good reputions by a) learning both social and payoff maximizing utility functions and b) combining them via a constrained maximization approach. We present a case study (using both humans and automated agents) that demonstrates the potential of this method as well as a deficiency of best response approaches.

1. Introduction

Most multiagent learning algorithms to date have tried to learn best-response strategies (e.g. [6, 2]). A best-response strategy is a strategy that maximizes an agent's payoff given the strategies of the other agents in the system. Generally, the approach taken is to try to learn to play a Nash equilibrium, which means that no agent has incentive to unilaterally change its strategy. This leads to good results at times, but at other times results in low payoffs for all agents involved. Iterated social dilemmas, such as the iterated prisoner's dilemma [1] and public goods games [3], are examples of games in which best-response strategies tend to converge to undesireable solutions since the (one-shot) Nash equilibrium is not pareto efficient.

The Nash equilibrium concept is important because a) it allows mutually adapting agents to establish equilibrium and b) it allows agents to protect themselves from receiving very low payoffs (in general). A downside to using such strategies, however, is that an agent, in the very process of playing them, eliminates other possibilities that could be more beneficial. This concept ties in deeply to the tradeoff between exploration and exploitation [7]. An agent can exploit its knowledge (i.e., play its part of a Nash equilibrium) at the cost of perhaps lower future payoffs, or it can explore (and teach) at the expense of perhaps lower current payoffs.

Psychology provides some important insights into multiagent learning. For example, Frank, while not throwing out traditional concepts of payoff maximization, points out that there is something more [5]. He shows that there are many times that people perform seemingly irrational actions (as far as payoffs are concerned) and end up thriving because of the reputations that such irrational actions establish. Frank argues that what is most important is not what an agent will actually do, but, rather, what other agents in the society believe that it will do (the agent's *reputation*).

We concur with Frank. In repeated play games, payoff maximizing techniques are not always sufficient to generate high payoffs when interacting with other learning agents. Agents must cultivate their reputation as well as deal with those of others. This shifts attention from the Nash equilibrium to reputational equilibrium. A *reputational equilibrium* exists when no agent believes that it has an incentive to unilaterally change its reputation. In this paper, we discuss reputational equilibrium and present a framework for establishing them.

2. Reputational Equilibrium

Consider the prisoners' dilemma payoff matrix shown in Table 1. At each iteration of the game, each agent can either cooperate (C) or defect (D). If both agents cooperate, then they both receive a payoff of 3. If they both defect, then they both receive a payoff of 2. If one agent cooperates and the other defects, then the cooperating agent receives a payoff of 1 and the defecting agent receives a payoff of 4. Defecting always yields a higher payoff than cooperating, and mutual defection is the Nash equilibrium strategy. However, if both agents cooperate, they both receive a higher payoff than if they both defect.

	C	D
C	(3, 3)	(1, 4)
D	(4, 1)	(2, 2)

Table 1. Payoff matrix for a prisoner's dilemma game.

We now discuss the repeated play of this game in terms of reputation. Suppose agent 1 defects on the first iteration of the game. From this action, agent 2 could label agent 1 as an aggressor, which will affect the way that he/she plays in the next iteration. On the other hand, if agent 1 cooperates, agent 2 could label agent 1 as someone that is willing to cooperate (i.e., share the wealth), or, perhaps, as someone who can be manipulated. Thus, with an action comes not only a material payoff, but also a social consequence leading to a reputation.

Now suppose that two agents have been playing the game with each other for some time, and both agents believe that the other agent will always defect. This is a reputational equilibrium, since both agents believe that changing their reputation (which would require at least one cooperative action) would lower their payoffs.

Consider, on the other hand, two agents playing (and believing that the other agent is playing) tit-for-tat (TFT) [1]. TFT begins an iterated prisoners' dilemma by cooperating, and thereafter plays the action that the other agent played on the previous iteration. This strategy builds the reputation: "I will cooperate if you will." Two agents, both playing TFT, will always cooperate. In this situation, reputations are also in equilibrium since if one of the agents unilaterally changes its reputation (which would require at least one defection) it would receive lower average payoffs in the future.

We have described two possible reputational equilibria for the same game. The first yields an average payoff per iteration of 2 for both agents and the second yields an average of 3. Thus, we would say that the second reputational equilibrium is better than the first (the reputational equilibrium corresponding to the Nash equilibrium) since it yields a higher average payoff for both agents involved.

We seek to identify ways in which learning agents can establish "good" reputational equilibria via reputation. In general, we classify "good" reputational equilibrium as combinations of reputations that yield ϵ-pareto efficient payoffs (that is, payoffs that are within ϵ of the pareto frontier).

We note that the outcome of a reputational equilibrium may possibly always be a Nash Equilibrium of the repeated game. Even if this is so, reputational equilibria differ from Nash equilibria in the way that they are formed. This changes the focus of learning and, hopefully, if agents are wise in their selections of reputation, increases the quality of the solutions that agents converge to. In short, reputational equilibria represent a process in which a pattern of actions build reputations whereas a Nash equilibria of the repeated game is only an outcome.

3. Reputation vs. Payoff Maximization

The conventional method of learning a best response in a repeated game has been to learn the expected payoffs an action brings over a window of time. This method was used by Sandholm and Crites [11] to test the effectiveness of various Q-learners in the iterated prisoner's dilemma. These Q-learners successfully learn to play optimally (cooperate) when associating with TFT. However, they learn suboptimal policies when they associate with other Q-learners. Sandholm and Crites stated that the reason the Q-learners are unable to learn mutual cooperation against other learners is due to the nonstationarity of the environment (caused by mutually adapting learners).

Littman and Stone verified these results ([8]) and showed the need for what they called "leader algorithms." Leader algorithms, such as TFT, teach best response agents via threats and punishments to play cooperatively. As a result, they perform better in the long run. They pointed out, however, that leader algorithms may not perform well when associating with each other. They concluded that a mix of follower-like (best response) and leader-like (reputation building) qualities should be present in an agent for it to perform well when faced with unknown opponents. Littman and Stone also used the principle of leader algorithms to show how a repeated-play Nash equilibrium could be computed for two-agent matrix games [9]. The algorithm, however, requires full knowledge of the structure of the game. Their approach is a reputational equilibrium-like approach since action selection is made not according to what will maximize payoffs, but rather what reputations are being established by the actions.

The above examples show the need for multiagent learning algorithms that can learn effectively against large classes of other learners. Best-response strategies alone cannot achieve this goal in some important games due to the nonstationarity of the world. Rather, agents must cultivate their reputations in addition to using payoff maximizing strategies. The goal is still payoff maximization, but the methodology is different.

In the next sections, we show how both leader-like learning algorithms and follower-like (best response) learning algorithms can be mixed to create a learner that performs well when associating with a large class of agents in both matrix and extensive-form games.

1. $S_s = \{i : U_s^{So}(i) \geq 0\} \bigcup \{\text{argmax}_i \, U_s^{So}(i)\}$
2. Select action i

$$i = \begin{cases} \text{argmax}_{i \in S_s} \, U_s^M(i) & \text{with probability } 1 - \eta \\ \text{rand}(|A_s|) & \text{otherwise} \end{cases}$$

where $|A_s|$ is the number of actions in state s.

Table 2. Action selection in the SPaM framework.

4. Reputation Through Social Committments

Establishing reputational equilibrium in repeated play games can be expensive, since both agents may try to change their reputation multiple times in multiple ways before reputations come into equilibrium. Moreover, they can be difficult to assess since an agent's reputation is "in the eye of the beholder," of which the agent does not have complete information. Because of these difficulties, we focus instead on learning social utility to establish good reputations rather than learning to establish an "optimal" reputation. Social utility should encode issues that establish reputation, such as trust, threats, revenge, etc.

In our approach, an agent learns both a payoff maximizing utility $U_s^M(i)$ and a social utility $U_s^{So}(i)$ for each action i from each state s. Using this framework (called SPaM (*S*ocial and *Pa*yoff *M*aximizing)), an agent selects its actions according to the constrained maximization method shown in Table 2. The basic notion of the method is to select the action that yields the highest payoff from the set of socially responsible actions. If no action is socially responsible, then the action that is considered to be the most socially responsible is chosen. The method assumes that actions that are socially responsible have social utility greater than or equal to zero, and actions that are not socially responsible have social utility less than zero. SPaM also calls for exploration a fraction of the time.

Since SPaM commits only to actions that are socially responsible, the reputations it establishes are based on the principles that social responsibility advocates. This means that if the agent's social utility funtion does, indeed, give high utility to those actions that are socially responsible, the agent will establish good reputations. This, in turn, leads to good reputational equilibria.

$U_s^M(i)$ can be learned via any best response learning algorithm including ficticious play (FP) [6], WoLF [2], etc. $U_s^{So}(i)$ is more difficult, however, since social values are case specific. Thus, it must be built on at least some knowledge of the agents with whom one interacts. We illustrate by example how social utility can be learned for a particular game in the next section.

5. Case Study

In this section, we design a learning agent for an extensive-form prisoner's dilemma using the SPaM framework. We then show how this agent learns when associating with both learning agents (self, ficticious play (FP), and humans) and static agents (TFT and various random agents). Before doing this we describe the social dilemma game that we use to evaluate the SPaM agent.

5.1. An Extensive-Form Prisoner's Dilemma

An extensive-form prisoners' dilemma (EFPD) is shown in Figure 1. In the game, two agents (shown in the figure as a circle and a square) begin on opposites sides (corners) of the world. The world is divided by a wall containing four different gates which are initially open to begin each round. The goal of each agent is to move across the world to the other agent's starting position as quickly as possible. This goal is represented by the payoff that each agent receives at the end of its turn which is a function of the number of moves it must take to reach its goal. The physics of the world are as follows:

1. Agents may move up, down, left, and right. (To speed learning, we reduced the size of the action space of the game by only allowing movements that could possibly move an agent closer to its goal or the gates, and omitted the use of other actions.)
2. Moves into walls or closed gates result in the agent remaining where it was before the action was taken.
3. If both agents arrive and attempt to move through gate 1 at the same time, gates 1 and 2 close (without allowing either of the agents passage).

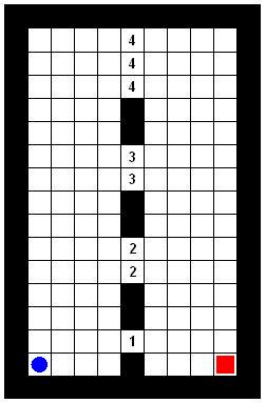

Figure 1. Prisoners' dilemma game in extensive form.

4. If one agent moves through gate 1 and the other agent does not, then gates 1, 2, and 3 close (after the defecting agent moves through the gate).
5. If one agent moves through any gate, then gate 1 closes.
6. Agents may move into the same square at the same time.
7. When an agent reaches its goal state, it receives a reward of $r = 40 - n$, where n is the number of steps taken to reach the goal.

When an agent attempts to move through gate 1, it is said to have defected (d) (the agent's stage action). Otherwise, it is said to have cooperated (c). Viewed in this way, the game turns into the game shown in matrix form in Table 3, where S, P, R, and T are all expected payoffs. If the stage actions (c or d) of the agents are played optimally, then $S = 8$, $P = 15$, $R = 24$, and $T = 30$. This is, by definition a prisoners dilemma since $\frac{S+T}{2} < R$ and $S < P < R < T$ [1].

The EFPD game is an abstraction of some real world situations [4], especially those in which agents (including humans and mixes of humans and automated agents) negotiate a disputed resource. It is useful since it encodes both communication and signaling [3, 10] (if each agent knows the other agent's state) into the decision making process via the various moves that agents can make towards the gates.

5.2. A SPaM Agent

The SPaM agent learns payoff maximizing and social utilities via the following algorithms.

Learning $U_s^M(i)$: We mentioned previously that many best response reinforcement learning algorithms can be used. We selected a ficticious play (FP) variant (in which more recent moves are given higher weight) because of its simplicity. Since a reward r is only received at the end of a stage (a stage consists of all the steps an agent takes to move from its start position to its goal position), all states and actions played during the stage are updated at the end of the stage using r.

Learning $U_s^{So}(i)$: The way in which social utility is learned is shown in detail in the algorithm in Table 4. The algorithm has a lot in common with FP, but actions are rewarded not according to the material payoffs received by the agent, but according to social consequences.

	c	d
c	(R, R)	(S, T)
d	(T, S)	(P, P)

Table 3. Generalized payoff matrix for the stage game of the EFPD.

- Let a be the SPaM agent and let $\hat{a}$ be the other agent.
- Let a_i^t and $\hat{a}_i^t$ be the stage actions of the two agents in stage t.
- Let $m_{a_i,\hat{a}_i}$ (where $a_i, \hat{a}_i \in \{d, c\}$ (d is defect and c is cooperate)) be the expected payoff for agent $\hat{a}$ (as observed in play so far) when the stage actions a_i and $\hat{a}_i$ are played.
- Let $s = (s_a, s_{\hat{a}}, g)$ be a state, where s_a and $s_{\hat{a}}$ are the positions of agents a and $\hat{a}$ respectively and g is the state of the gates.
- for each state s and action i (agent a's actions)
  $\kappa_s^i(a_i, \hat{a}_i) = 0$ for $a_i = d, c$ and $\hat{a}_i = d, c$
- Let $\mu_c = \mu_d = 0$
- t = 1

Repeat

1. While s (current state) not a goal state
   (a) Calculate $U_s^M(i)$ for all actions i (from state s) according to FP model
   (b) Calculate $U_s^{So}(i)$ for all actions i (from state s)
      i. $totalcount = \sum_{k=d,c} \sum_{j=d,c} \kappa_s^i(k, j)$
      ii. $U_s^{So}(i) = \frac{\kappa_s^i(d,d)}{totalcount}(\mu_d - m_{d,d})$
      - $\frac{\kappa_s^i(d,c)}{totalcount}(\mu_c - m_{d,c})$
      + $\frac{\kappa_s^i(c,d)}{totalcount}(\mu_d - m_{c,d})$
      - $\frac{\kappa_s^i(c,c)}{totalcount}(\mu_c - m_{c,c})$
   (c) Select action and move according to Table 2.
2. Receive payoff and update
   (a) FP model for all (s, i) played in stage t
   (b) The other agent's expected payoff $m_{a_i^t,\hat{a}_i^t}$
   (c) $\mu_c = \frac{m_{c,c}+m_{d,c}}{2}$ and $\mu_d = \frac{m_{c,d}+m_{d,d}}{2}$
   (d) $\kappa_s^i(a, \hat{a}) = \kappa_s^i(a, \hat{a}) + t$ for all (s, i) played in stage t
3. $t = t + 1$

Table 4. The SPaM learning algorithm.

The intuition behind Table 4 is that the social utility of an action $U_s^{So}(i)$ is decreased if it leads to decreasing the other agent's utility for playing c (see 2.b.iii part 2) or increasing the other agent's utility for playing d (see 2.b.iii part 3). Likewise, the utility of an action is increased if it leads to increasing the other agent's utility for playing c (see 2.b.iii part 4) or decreasing the other agent's utility for playing d (see 2.b.iii part 1). Thus, this social utility function is closely related to Littman and Stones leader algorithms [8].

5.3. Results

We conducted experiments in which various kinds of agents played the iterated EFPD game: humans, SPaM, two kinds of ficticious play agents (FP and FP2), TFT, and various random agents. Table 5 show the results of the various encounters between learning agents (all agents except TFT and the random agents). In the table, *# Iterations* refers to the average number of iterations it took for the agents to

Agents	# Iterations	Strategy
SPaM-SPaM	84	cc
SPaM-FP	731	cc
SPaM-FP2	965	cc
FP-FP	261	dd
FP-FP2	176	dd
FP2-FP2	170	dd
Human-SPaM	84	cc
Human-FP	206	cc, other
Human-FP2	134	cc, [cc-dc]
Human-Human	–	cc

Table 5. Results from interactions between various learning agents.

converge to their final strategy (meaning both agent's stage action was the same 17 out of 20 stages) and *Strategy* refers to the solution the agents converged to (cc, for example, means mutual cooperation). We describe the outcomes of the various encounters between the learners (as well as the FP and FP2 learners) below.

FP – FP is the same ficticious play agent used for the payoff maximizing part of the SPaM agent. FP, as do all the current best response algorithms such as Q-learning, WoLF, etc., learns mutual defection in self play and against other best response agents (such as FP2) in the EFPD. However, SPaM, because of its leader strategy-like approach is able to teach FP to cooperate, which gives higher payoffs to both agents. When FP associates with humans, results are mixed as humans sometimes are able to teach FP to cooperate. We discuss this in more detail below (see subsection Humans).

FP2 – FP2 is the same as FP except that the reward (reinforcement) given to an action is the average of the results of the current stage and the subsequent stage (as opposed to just the results of the current stage). Against best response agents, FP2 learns the same policies (mutual defection) as does FP. It also behaves about the same as FP against SPaM, learning mutual cooperation. However, it behaves quite differently with humans (see subsection Humans).

SPaM – SPaM teaches best response agents to cooperate in the EFPD game and learns to cooperate itself once the best response agents show they are willing to cooperate if SPaM does. Additionally, SPaM is able to learn mutual cooperation in self play. It also learns mutual cooperation when associating with humans.

Humans – The results are quite interesting when humans play the game. Our preliminary results (with informal test subjests) have all suggested that humans generally learn to cooperate when they play with each other in this game. It would seem natural that humans would be able to teach best response agent to cooperate since SPaM can. However, it does not appear to be so easy for humans. With FP, mutual cooperation was eventially learned (it usually took a while and caused the human frustration) with some humans. However, other humans have a very difficult time trying to teach FP, and get very frustrated in the process. In the case study, they typically gave up trying to teach FP to cooperate after two or three hundred stages, sometimes learning mutual defection and even other times getting exploited by FP. Thus, best response learning algorithms such as FP appear to be unacceptable when humans are involved.

The results are quite different when humans play FP2. Since FP2 bases its learning on the results of the current and the subsequent stage, humans are able to more easily teach it to cooperate. That is what happens with some humans. However, other humans are able to learn to exploit FP2 by basically alternating between playing c and d while FP2 cooperate. FP2 allows this since the average of S and R (see Table 3) is 16, whereas the payoff of mutual defection is 15. Thus, FP2 makes mutual cooperation with humans easy but is exposed to exploitation.

All humans that have particpated in our case study have learned mutual cooperation in play with SPaM in the EFPD. This is because humans realize that a) SPaM will not tolerate high levels of defection (it learns to defect if the human tries to defect) and b) SPaM shows the human it is trustworthy by not defecting when it has the opportunity to do so. The test subjects used both teaching and learning strategies with SPaM, both of which yield mutual cooperation. This differs from play with best response agents in which humans must take a teaching approach and not a learning approach in order for the agents to converge to mutual cooperation.

Learners Against Static Agents We also tested SPaM, FP, and FP2 against non-learning agents. These non-learning agents were TFT and a random agent that defected with various probabilities. All these agents chose the gate they would attempt to enter before the stage began, and then moved randomly toward that gate. The results are described below.

TFT – Like all history-free best response agents, FP learns to defect against TFT. Thus, it receives a payoff of about 15 each stage. FP2 learns to cooperate against TFT (as has been shown of this kind of best response agent previously [8, 11]) and receives a reward of about 22.5 (on average) per stage. This is lower than the 24 received for mutual cooperation because of occasional exploration by FP2. SPaM generally learns to cooperate with TFT although cycles of cd-dc and periods of mutual defection also emerged. This is because of exploration by SPaM and the fact that SPaM's social utility function was built on the assumption that an agent indicates its intent by which gate it moves toward (which TFT largely ignores) rather than the play from stage to stage. Nevertheless, SPaM still receives a coopera-

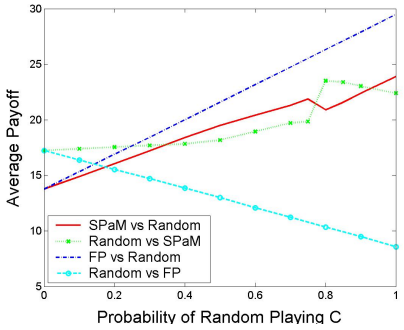

Figure 2. Payoffs in which a stationary agent (Random) is involved.

tive reward of about 21.2 per stage (on average) when playing TFT.

Random Agents – We also tested FP, FP2, and SPaM against agents that played randomly with various probabilities of defection. Static random agents violate the assumptions made by SPaM that reputation is important. The payoffs for each agent are shown in Figure 2 for various probabilities of cooperation by the random agent (x-axis). The results are slightly different than expected because of the exploration of the learning agents. The payoffs for FP2 are not shown but are essentially equivalent to those of FP.

While FP learns to always defect against all the random agents, SPaM generally only defects when it believes (based on the random agents moves toward the gates) that the random agent will defect. Thus, SPaM learns to always defect against an agent that always defects but learns to always cooperate against an agent that always cooperates. An interesting trend is shown in the figure at about the point of 80% cooperation by the random agent, at which point SPaM's average reward per stage drops and the random agent's payoff jumps sharply. It is at this point that SPaM's social utility function believes that it is better to not retaliate against defection. This point appears to be very difficult for learning agents (even humans) to find.

6. Conclusions and Discussion

While best response algorithms learn successful policies in many contexts, they frequently learn poor policies in others (such as iterated social dilemmas). Because of this, we focus attention, instead, on algorithms that place primary attention on cultivating good reputations, and secondary attention on material payoffs.

In this paper, we presented a framework that learns both social and payoff maximizing utility functions. We showed how the framework can be used to create a robust agent in an iterated extensive-form prisoner's dilemma game. The agent learns to cooperate in self-play, with humans, and with best response algorithms. We showed that best response algorithms learn mutual defection in self play. Also, humans have difficulty teaching history-free best response agents to cooperate in this game. Best response agents with history may expose themselves to being exploited.

The success of SPaM can be traced to its incorporation of both payoff maximizing and socially minded utilities. The combination of these via constrained maximization allows the agent to establish a positive reputation while still pursuing individually profitable actions. The result is good reputational equilibria.

While successful, the social utility function used in the case study presented is valid only for 2-player extensive-form games similar to this EFPD. For future work we plan to improve the social utility learning algorithm so that it works in a larger class of games, including games with more than two players. Also, the algorithm requires knowledge of the payoffs and state (position) of the other agent; we wish to remove or soften this requirement in future work.

References

[1] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[2] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 2001.

[3] C. F. Camerer. *Behavioral Game Theory*. Princeton University Press, 2003.

[4] J. W. Crandall and M. A. Goodrich. Multiagent learning during on-going human-machine interactions: The role of reputation. In *AAAI Spring Symposium on Interaction between Humans and Autonomous Systems over Extended Operation*, pages 35–40, 2004.

[5] R. H. Frank. *Passions Within Reason: The Strategic Role of the Emotions*. W. W. Norton and Company, 1988.

[6] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. The MIT Press, 1998.

[7] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[8] M. L. Littman and P. Stone. Leading best-response strategies in repeated games. In *IJCAI Workshop on Economic Agents, Models, and Mechanisms*, 2001.

[9] M. L. Littman and P. Stone. A polynomial-time nash equilibrium algorithm for repeated games. In *2003 ACM Conference on Electronic Commerce (EC '03)*, 2003.

[10] F. Sanabria, F. Baker, and H. Rachlin. Signaling cooperation and defection in a free operant prisoner's dilemma game. In *Society for the Quantitative Analyses of Behavior (SQAB) Conference*, 2002.

[11] T. W. Sandhom and R. H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Adaptation and Learning in Multi-Agent Systems*, pages 191–205, 1995.