Multiagent Learning During On-Going Human-Machine Interactions: The Role of Reputation

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

Abstract

Multiagent learning is an important tool for long-lasting human-machine systems (HMS). Most multiagent learning algorithms to date have focused on learning a best response to the strategies of other agents in the system. While such an approach is acceptable in some domains, it is not successful in others, such as when humans and machines interact in social dilemma-like situations, such as those arising when human attention is a scarce resource shared by multiple agents. In this paper, we discuss and show (through a user study) how multiagent learning algorithms must be aware of reputational equilibrium in order to establish neglect tolerant interactions.

Introduction

When humans interact with automated systems for extended periods of time, multiagent learning can be important to the success of the system. This is because learning allows agents to adapt, which is necessary for complex and ongoing HMS since a) automated systems (and humans) encounter unexpected situations that the system is unequipped to handle without adaptation, b) automated systems must be tuned to the user's needs, and c) preferences and roles of the agents (human and automated) will change. However, learning in the presence of other learning agents is difficult.

When agents learn in the presence of other learning agents, strategies should come into equilibrium. This is a minimum requirement, since high performance in unstable environments is difficult. Traditional multiagent learning algorithms have sought to learn equilibrium strategies based directly on expected payoffs (such as the Nash equilibrium. Such learning algorithms, however, often converge to solutions that yield undesireable payoffs. In this paper, we discuss finding an equilibrium based on the concept of reputation. Informally, an agent's *reputation* is its expected pattern of behavior.

Integral to social communities are *reputational equilibria*. A reputational equilibrium is when no agent has an incentive to unilaterally change its reputation. The success of a society and the individuals within the society is dependent on the reputational equilibrium maintained in the society. This relates closely to the concept of a repeated play Nash equilibrium (Littman & Stone 2003) since repeated interactions mean that current actions can have an effect on payoffs that might be received far into the future.

"Good" reputational equilibria lower the cost of interactions within the group and, thus, increases the benefits of interactions. Thus, reputations affect *relationship maintenance*. Relationship maintenance is the time that is required to maintain and establish relationships. Simply put, if a human and a machine know what to expect from each other, interactions will be more productive.

Relationship maintenance is an important part of the neglect tolerance (Crandall & Goodrich 2003) of an automated agent. To better illustrate the relationship between reputational equilibrium, relationship maintenance, and neglect tolerance, consider Figure 1. In the figure, three different interaction schedules are represented. In the top schedule, a "good" reputational equilibrium has been established between an automated agent and a human. First, the human and automated agent interact for a short time (task 1), after which the human turns his/her attention to another task while the automated agent carries on the task. After a time, the human returns his/her attention back to task 1, and the process continues. Consider, however, the case in which reptutations are not in equilibrium (middle schedule). Since the human does not understand what the automated agent is going to do, he/she spends more time interacting with the automated agent to establish a "good" reputation and relationship. Thus, the human has less time to perform other tasks (i.e., neglect tolerance is decreased). Consider, also, the case in which reputations are in equilibrium, but the reputations between the agents are not "good" reputations. In such cases, it is likely that a human must interact with the automated agent more often to maintain high performance levels (bottom schedule). Thus, neglect tolerance is also decreased in this case.

In this paper, we discuss and develop the concepts of reputational equilibrium and relationship maintenance in multiagent systems. First, we review best-response learning algorithms followed by a discussion on perspectives from psychology. Second, we develop the concepts of reputational equilibrium and relationship maintenance by examples. Third, we discuss several multiagent algorithms that take into consideration reputations when selecting actions. Finally, we use one of these algorithms to show, through a user study, that considering reputations when selecting ac-

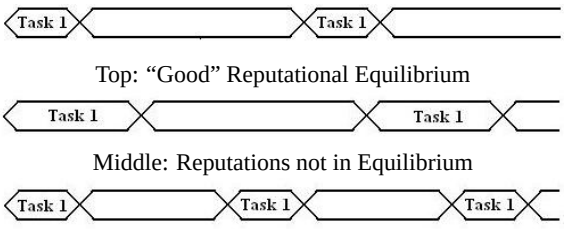

Figure 1: The neglect tolerance of an automated agent is dependent on the reputations established between human and machine. Different interaction schedules are required (because of relationship maintenance), depending on the reputations established by the agents in the system.

tions reduces relationship maintenance (and, thus, neglect tolerance).

Best Response Learning Algorithms and Perspectives from Psychology

Most of the multiagent learning algorithms to date have tried to learn best-response strategies (e.g. (Fudenberg & Levine 1998; Bowling & Veloso 2001)). 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 Nash equilibrium, which means that no agent has incentive to unilaterally change its strategy. This approach leads to good results at times, but at other times results in low payoffs for all agents involved. Social dilemmas, such as the iterated prisoner's dilemma (Axelrod 1984) and public goods games (Camerer 2003), are examples of games in which best-response strategies tend to converge to undesireable solutions since the Nash equilibrium is not pareto optimal.

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 processes of playing them, eliminates other possibilities that could be more beneficial. This concept ties in deeply to the trade-off between exploration verses exploitation (Kaelbling, Littman, & Moore 1996). 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 at the expense of perhaps lower current payoffs.

Psychology provides some important insights into multiagent learning, especially when dealing with interactions between humans and machines. Psychologist have repeatedly pointed out that traditional game theory (and, thus, many of the important concepts that can be obtained from it), do not model human behavior (e.g. (Colman 2003)[1]). This suggests that artificial agents that use such techniques may not interact with humans in efficient ways.

[1]Colman suggests the use of spychological game theory, which relies on biases related very much to reputational equilibrium.

Frank, while not throwing out traditional concepts of rationality completely, points out that there is something more (Frank 1988). Frank shows that there are many times that people perform seemingly irrational actions (as far as payoffs are concerned) 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 think that it will do. Hence, an agent's reputation is of utmost importance.

Reputational equilibrium and Relationship Maintenance

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 reward 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 the prisoners' dilemma.

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 in the form of 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) (Axelrod 1984).[2] 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.

[2]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."

Notice that we have described two reputational equilibria for the same game. The first yielded an average payoff per iteration of 2 for both agents and the second yielded 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 payoff. Also, we would say that the second reputational equilibrium has lower relationship maintenance since it requires only two iterations to receive a cumulative payoff of 6, whereas it takes three iterations to receive the same cumulative payoff.

Playing until reputations get sorted out can be expensive, since both agents may try to change their reputation multiple times in multiple ways before reputations come into equilibrium. This may mean low payoffs on some iterations so relationship maintenance could be high (i.e., low average payoffs because of poor expectations) until reputations come into equilibrium.

In this paper, we seek to identify ways in which learning agents can establish "good" reputational equilibrium. In general, we classify "good" reputational equilibrium as combinations of reputations that a) yield ϵ-PO payoffs (that is, payoffs that are with ϵ of the pareto frontier) and b) avoid begin exploited by agents that are not willing to cooperate.

Examples of Neglect Tolerant Learning

Neglect tolerant learning refers to learning that reduces relationship maintenance. In this section, we discuss several different algorithms that establish reputations which frequently reduce relationship maintenance.

Leader Algorithms

Littman and Stone (Littman & Stone 2001) presented an algorithm designed to lead best-response strategies in repeated-play games. The algorithm (called Godfather), which is based on the tit-for-tat strategy, is designed for two-agent games and assumes knowledge of the payoff matrix. Godfather calculates the security level of each agent, which is the expected payoff an agent can guarantee itself if it played the minimax strategy. It then searches for a *targetable pair* which is a solution to the game for which each player receives more than its security level. If such a solution exists, Godfather plays its half of the strategy, hoping the other agent will do so as well. If the other agent does, then Godfather continues to play its part of the targetable pair on the next iteration. If the other agent does not, however, then Godfather, in the spirit of tit-for-tat, plays a strategy that forces the other agent to receive no more than its security level. This behavior leads a class of best-response learning agents (those that calculate their payoffs over several iterations) to learn to play cooperatively. However, Godfather is not guaranteed to perform well in self-play. Littman and Stone extended this work (Littman & Stone 2003) to compute a repeated-play Nash equilibrium strategy. While the algorithms they propose are not learning algorithms, they demonstrate the need to consider reputational equilibrium (which they do through threats) and not just best-response strategies in multiagent algorithms.

1. Let $\lambda \in (0, 1]$ be a learning rate, let α_0 be high, and initialize sat to false.
2. Repeat,
   (a) Select an action a_t according to the following criteria:

$$a_t \leftarrow \begin{cases} a_{t-1} & \text{if (sat)} \\ \text{rand}(|\mathcal{A}|) & \text{otherwise} \end{cases}$$

   where $|\mathcal{A}|$ is the number of actions available to the agent.
   (b) Receive reward r_t and update:
      i. Update: $sat \leftarrow \begin{cases} true & \text{if } (r_t \geq \alpha_t) \\ false & \text{otherwise} \end{cases}$
      ii. Update: $\alpha_{t+1} \leftarrow \lambda\alpha_t + (1 - \lambda)r_t$

Table 2: The S-Algorithm.

Satisficing Algorithm

Littman and Stone's algorithms play cooperatively until the other agent does not, at which point the algorithms retaliate until the other agents does play cooperatively. Such a strategy can be effective, but it can also lead to repeated cycles of retatiliation. The next algorithm (we call it the S-Algorithm) we discuss is based on the principle of satisficing. The S-Algorithm (Stimpson & Goodrich 2003) is applicable to n-agent, m-action games and is requires no knowledge of the game structure, nor the actions of the other agents. Despite these knowledge restrictions, Stimpson showed that, in the multiagent social dilemma (MASD),[3] if aspirations are set sufficiently high and similar for all n agents and learning rates are sufficiently slow,[4] then the agents will likely converge to the Nash bargaining solution.

The general algorithm is shown in Table 2. In step 1 of the algorithm, the agent's aspirations are initialized and the agent is designated to be unsatisfied ($sat = false$) with its current rewards. Initial aspirations should be at least as high as the highest payoff an agent can receive playing the game. In step 2, the agent plays the game repeatedly. First, the agent selects its next action according to the criteria of step 2(a), which says that the agent will repeat its action of the previous iteration (a_{t-1}) if it is satisfied, but act completely randomly otherwise. In step 2(b), the agent updates it internal state. It does so by a) determining if it is satisfied with the reward r_t that it received as a consequences of its action (a_t), which is done by determining if r_t met or exceeding its aspirations (α_t), and, then, b) updating its aspirations.

The S-Algorithm never explicitly seeks to maximize rewards, yet it learns (with high probability) to play pareto optimal solutions in self play and learns to avoid being exploited by selfish agents. As does tit-for-tat and leader algorithms, the S-Algorithm resorts to a fallback strategy if its aspirations are not met. Its fallback strategy, however, differs from the other two in that instead of completely defecting (to punish other agents) when it is not satisfied, it resorts to random behavior, which, in effect, leaves cooperation as

[3]The MASD is equivalent to public goods games in (Camerer 2003).

[4]Small changes in the algorithm eliminate these restrictions, but we omit these changes in the interest of space.

a possibility while still punishing other agents (although not as severely). Such a fallback encodes a reputation which implicitly says: "I am willing to cooperate, if you are too."

Social and Payoff Maximizing Agent (SaPMA)

The last algorithm (SaPMA) we discuss is a new algorithm that performs by learning two different utility functions. To illustrate the algorithm and how it relates to HMS, we present a game which typifies many human-machine interactions. When then show how SaPMA plays the game against various agents in the next section.

Extensive-Form Prisoners' Dilemma An extensive-form prisoners' dilemma (EFPD) is shown in Figure 2. 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, as the payoff each agent receives at the end of its turn 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 make the game easier for the automated agents to learn, we restricted movements to those that could possibly move an agent closer to its goal, 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).
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 reward $r = 40 - n$, 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). 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 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 $S < P < R < T$ (Axelrod 1984).

	C	D
C	(R, R)	(S, T)
D	(T, S)	(P, P)

Table 3: Generalized payoff matrix for the EFPD.

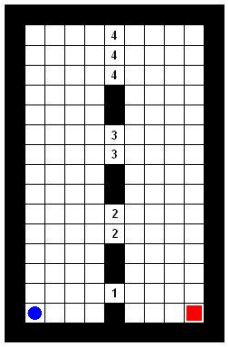

Figure 2: Prisoners' dilemma game in extensive form.

The EFPD game is an abstraction of some HMS. Consider, for example, a human-robot system where a single human operator interacts with multiple robots that can adjust their autonomy. In such a game, the attention of the human can be viewed as a resource that must be shared by the robots. A robot that adjusts its own autonomy mode in a way that requires high human attention/workload is essentially defecting against the other robots in the system (i.e., it attempts to move through gate 1). This means that the other robots may not get enough human attention, so they they must also change their autonomy modes to demand more of it. In such a case, no robot receives enough attention from the human. If all (or most) robots, however, select more appropriate autonomy modes (i.e., they cooperate) then all can receive sufficient attention from the human operator, and the whole system benefits. Clearly, this game demonstrates that social dilemmas can arise in HMS that affect long-term success.

Another important element of this game is that it also allows implicit communication. A move toward or away from Gate 1 communicates to the other agents in the game what it plans to do (assuming that agents can observer each others state). This helps the agents to coordinate their actions, and, if leader-like algorithms are employed, to enforce cooperation.

Since the EFPD has the above properties, it shows the relationship maintenance requirements of two agent interactions in two ways. First, it shows the outcome (reputational equilibrium) of agents building their individual reputations. If agents converge to mutual cooperation, then relationship maintenance is lower than if agents converge to mutual defection. Second, it shows how long it takes for different combinations of algorithms to converge. Longer times until convergence generally mean higher relationship maintenance because lower average payoffs are generally received while agents are learning (and have not converged).

The Algorithm The gist of the algorithm is to compute for each action from each state both a payoff maximizing utility (selfish) and a social utility. The payoff maximizing component (M) is concerned only with computing the expected payoff ($U_s^M(i)$) of taking action i from each state

Repeat steps 1. and 2. while game lasts

1. Repeat while state s is not an end state:
   (a) Compute $U_s^{So}(i)$ and $U_s^M(i)$ for all actions i available from state s.
   (b) Form $S_s = \{i : U_s^{So}(i) \geq 0\}$
   (c) 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.
   (d) Take action i and observe next state s
2. Receive reward pair (R_1, R_2) and
   (a) Update internal state.
   (b) Transition to start state.

Table 4: The SaPMA algorithm.

s. The social component (So) is concerned about computing the social value ($U_s^{So}(i)$) of taking action i from state s. The decision of which action to take is made by computing the two components and then selecting the action with the highest expected utility in the payoff maximizing component from a set of actions which the social component can endorse. The general algorithm is shown in Table 4.

Step 1 of SaPMA repeats until an agent reaches its goal state. It first calculates the utilities $U_s^M(i)$ and $U_s^{So}(i)$ for all actions i that agent 1 can take from state s. $U_s^M(i)$ is calculated according to standard ficticious play (FP) (Fudenberg & Levine 1998) and $U_s^{So}(i)$ is calculated using a leader algorithm-like technique.After it has computed $U_s^M(i)$ and $U_s^{So}(i)$, the SaPMA agent computes (in Step 1b) its satisficing set S_s as the set of actions i that have positive social utilities (i.e., $U_s^{So}(i) \geq 0$). In Step 1c, the action with the highest payoff maximizing utility (U_s^M) in S_s is the action chosen by the SaPMA agent with probability η.[5] Otherwise, the agent selects an action randomly (i.e., it explores). After choosing its action, the agent moves and waits for the state of the world to be updated. Thus, SaPMA's action selection algorithm consists of selecting the action that returns the highest reward that is good for the group.

In Step 2, when the agents reach their goal states (they may reach at different times, but the rewards are not issued until both agents have reached their respective goals), the rewards (R_1, R_2) are assigned. The agents take this reward pair to update their internal state (Step 1a). The agents are then returned to their initial starting positions, and a new round begins.

Results

In this section, we present two sets of results. First, we present results on what happens when two automated agents, FP and SaPMA, interact with each other in the EFPD. Second, we present results on what happens when humans interact with these automated agents in the EFPD.

[5]If two or more actions in the set S_s yield the same expected payoff then the algorithm randomly selects between these actions.

Machine-Machine Interactions

Table 5 shows results from interactions between serveral automated agents in the EFPD. The results represent an average of ten experiments each.[6] The table tells how long it took for the agents to converge[7] (*# Iters*) to their final strategies. The table also tells the average payoff that each agent received during the learning period (*Learning Ave.*) and the equilibrium strategy (*Strategy*). The results illustrate the need to take into account reputational equilibrium.

Agents	# Iters	Learning Ave.	Strategy
SaPMA-SaPMA	224	(19.4, 18.0)	(C, C)
SaPMA-FP	529	(19.3, 18.0)	(C, C)
FP-FP	299	(17.5, 15.7)	(D, D)

Table 5: Results from interactions between automated agents. Results are the averages of ten experiments.

SaPMA agents in self play establish reputations that lead to mutual cooperation. It took 224 iterations, on average, for the agents to converge. During those 224 iterations, the agent with the highest payoff received an average payoff of 19.4, while the other agent average 18.0. These payoffs are significantly less than the payoff that each agent received after convergence (about 24.0), however, it is the highest average of any of the other automated agents. Since they also took the least amount of time to converge (of the three different pairings), SaPMA in self play has the lowest relationship maintenance.

FP agents in self play establish reputations that lead to mutual defection. The average convergence time was 299 iterations, during which the average payoff was 17.5 for one agent and 15.7 for the other. These payoffs are actually higher than those received by the agents after they converged (about 15.0). Thus, the relationship maintenance actually increases after convergence in this case.

When SaPMA and FP associate in EFPD, reputations lead to mutual cooperation. This is because SaPMA uses a leader-like algorithm to make cooperation more desireable for FP than defection. To do this, SaPMA learned that it must approach gate 1 until FP moved away from it (towards gate 2), after which SaPMA also moved towards gate 2. Because of this, convergence was slower; nevertheless, the reputational equilibrium is good since relationship maintenance becomes low in the end. However, relationship maintenance is higher than in the case of SaPMA self play since it took longer to converge.

Human-Machine Interactions - A User Study

We next explain results of what happens when humans interact with machines in the EFPD. To do so, we performed a user study using six subjects, each having various exposure

[6]High variance, especially in the number of plays needed for convergence, was seen for all combinations of agents.

[7]If agents played the same solution 17 out of 20 iterations, then they were considered to have converged.

to game theory. Each subject played with SaPMA and FP, three playing against SaPMA first and three playing against FP first. These results (along with the averages) are shown in Table 6.

Subject	SaPMA		FP	
	# iters	payoff	# iters	payoff
A	85	(21.1, 18.8)	254	(19.9, 17.5)
B**	62	(23.7, 18.7)	181	(21.5, 18.8)
C	119	(19.3, 19.7)	300*	(16.9, 16.4)
D**	72	(21.5, 19.6)	65	(20.1, 19.3)
E	91	(19.2, 21.0)	229*	(15.4, 19.7)
F**	72	(20.3, 19.1)	207	(19.9, 17.2)
Ave.	84	(20.9, 19.5)	206	(19.0, 18.2)

* indicates that after the stated iterations, defection was still the primary action taken by the agents.

** indicates that the subject associated with FP first, then SaPMA.

Table 6: Results from humans interaction with SaPMA and FP in the EFPD.

When a human associated with SaPMA, the result was always mutual cooperation in the end. This was the case even though humans played against SaPMA differently. For example, those subjects that played with FP before playing with SaPMA tended to try to teach/force SaPMA to cooperate, just as they learned to do with FP. The subjects that played with SaPMA first, however, tended to trust SaPMA more and tended to cooperate without forcing cooperation. SaPMA learned to cooperate in both cases by adapting to the humans repututation and portraying it own reputation.

Results were varied, however, when humans interacted with FP. Two of the subjects decided that FP would never cooperate with them, so mutual defection was the result. The other four subjects eventually learned how to force FP to cooperate, but this generally took a long time and caused a good deal of frustration. Typical comments during the learning process with FP were: "I'd cooperate with him if he would just cooperate with me" and "He has no incentive to cooperate." In the end, it took much longer for humans and FP to converge than for humans and SaPMA, and the average payoffs during learning were not as high. Taking this into account and the fact that a third of the subjects never learned to cooperate with FP shows that relationship maintenance is much higher when a human associates with FP then when a human associates with SaPMA.

It should be noted that both subjects that could not figure out how to teach FP to cooperate played with SaPMA before playing with FP. This could have influenced their inability to teach FP to cooperate. However, this demonstrates how FP is inflexible to different strategies. This differs from SaPMA, as it was able to adapt to the different teaching methods of the subjects.

These results show that reputation is an important fact to consider in the design of multiagent systems. Humans should not be required to think like machines in order to interaction with them effectively. Thus, purely best response strategies are not sufficient, especially when machines interact with humans.

Conclusions

Multiagent learning can be very important in HMS. In social dilemma contexts such as those arising when the human is a scarce resource shared by multiple agents, learning algorithms that employ only best-response strategies are not acceptable. Learning algorithms used in such systems should take into account the reputations that they portray to other agents. Successful learning algorithms establish successful reputational equilibrium, which lowers the relationship maintenance of human-machine interactions. *fiddle* In this paper, we presented examples of such learning algorithms and presented an abbrieviated case study in which the relationship maintenance of a human-machine interactions.

References

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

Bowling, M., and Veloso, M. 2001. Multiagent learning using a variable learning rate. In *Preprint submitted to Artificial Intelligence*.

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

Colman, A. M. 2003. Cooperation, psychological game theory, and limitations of rationality in social interaction. In *Behavioral and Brain Sciences*, volume 26, 139–198.

Crandall, J. W., and Goodrich, M. A. 2003. Measuring the intelligence of a robot and its interface. In *Performance Metrics for Intelligent Systems (PerMIS'03)*.

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

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

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

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

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

Stimpson, J. R., and Goodrich, M. A. 2003. Learning to cooperate in a social dilemma: A satisficing approach to bargaining. In *The Twentieth International Conference on Machine Learning (ICML-2003)*.