An Approach to Cooperation in General-Sum Normal Form Games

Steven Damer
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455, USA
damer@cs.umn.edu

My primary research interest is social behavior for software agents to achieve cooperation in general-sum normal form games. An agent can easily be programmed to constantly cooperate in a normal form game, but such an agent is not suitable for environments with potentially hostile opponents. For a rational agent, the main reason to cooperate is to induce reciprocation; to reciprocate it is necessary to determine which moves are cooperative. In constant-sum games cooperation is impossible because any gain by one agent is a loss by the other agent. In other games (such as Prisoner's Dilemma) it is easy to identify cooperative moves because the opponent's payoffs for that move strictly dominate the other moves of the agent. In general it is not easy to identify cooperative strategies in arbitrary general-sum games.

1 Attitude

Given a normal form game, how do you determine which moves are cooperative? One model that has been used to explain cooperative behavior in humans is that players act as if they receive a share of opponent payoffs [Frohlich, 1974]. This model can be used to generate cooperative behavior for software agents. We say agents are cooperating when they select joint strategies which provide better outcomes than they could achieve individually.

Given a two-player general-sum normal form game G, with utility functions U_1 and U_2, cooperative moves are found by creating a modified game G' with the same set of moves and utility functions $U_1' = U_1 + A_1 * U_2$ and $U_2' = U_2 + A_2 * U_1$ where A_1 and A_2 are the *attitudes* of player 1 and player 2. The Nash equilibria of G' define strategies for the players which reflect that attitude values used to create G'. A_1 and A_2 can take any value, but we confine our attention to the range $[-1, 1]$. Values lower than 0 create strategies where the agent harms itself to harm the opponent, and values higher than 1 create strategies where the gain to the opponent isn't worth the cost to the agent.

Figure 1 shows the expected outcome of the calculated strategy in randomly generated games. The main influence on an agents payoff is the attitude of the opponent, but the agents own attitude also affects its payoff. Adopting a selfish attitude generally helps the agent, but if the opponent's attitude is positive reciprocating will improve the agent's payoff.

To use this technique to cooperate agents must coordinate on an appropriate set of attitude values. This can't be done by communication; a hostile agent would lie about its attitude. Therefore agents must learn the attitude of their opponent by observing the opponent's actions.

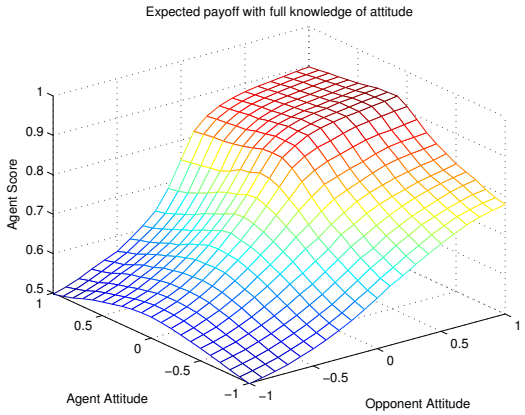

Figure 1: This graph shows the effect of attitude values on the performance of an agent in randomly generated games.

It's possible to learn attitude values using a particle filter to estimate the parameters used by the opponent to determine its strategy. This is complicated by the possibility of multiple Nash equilibria, but that can be handled by adding another parameter to the particle filter. One advantage of using particle filters in this way is that they can learn from observations from different games as long as the opponent stays the same.

Figure 2 shows the performance of two agents which estimate attitude values using a particle filter and reciprocate the attitude of the opponent with a slightly larger attitude value. In each round agents play against their opponent in a new randomly generated game. Agents can rapidly learn to cooperate and achieve good outcomes for both agents.

2 Restricted Stackelberg Response with Safety

Using a particle filter to estimate attitude allows agents to cooperate, but is counterintuitive because the particle filter provides a prediction of opponent behavior which is not used by the agent. We would like the agent to respond rationally to the prediction (given its chosen attitude). A best-response is

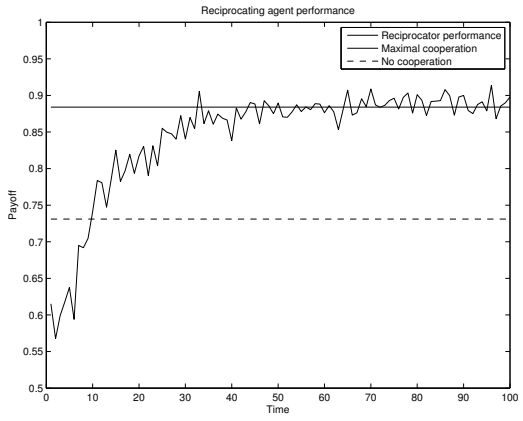

Figure 2: This graph shows the performance of 2 agents using attitude to cooperate. Performance occasionally goes above the maximum expected performance because the games are randomly generated.

easy to calculate, but without performance guarantees. On the other hand, playing strategies which provide guarantees can reduce performance against the prediction. A number of approaches have been taken to solving this problem [McCracken and Bowling, 2004] [Johanson *et al.*, 2007] [Bowling, 2005]; they provide different guarantees and are not all suitable for general-sum games.

We have developed Restricted Stackelberg Response with Safety (RSRS), a parameter driven approach which can smoothly adjust between best-responding to the prediction, dealing with an exploiting opponent, and providing worst-case guarantees. RSRS uses two parameters, w, the prediction weight, which controls the tradeoff between exploiting the prediction and dealing with an exploiting opponent, and r, the risk factor, which determines how much of the value of the game to risk in hopes of exploiting the prediction.

RSRS is calculated by creating a modified game, in which the moves are the same as the original game, but the payoffs are adjusted to a weighted average of the original payoffs and the expected payoff of the calculating players move against the prediction $U_1'(m_i, m_j) = (1 - w) * U_1(m_i, m_j) + w * U_1(m_i, p)$. Using the modified game, the RSRS is the strategy which maximizes performance against a best-responding opponent while guaranteeing a payoff within r of the safety value of the game.

Figure 3 shows the effect of r and w when using RSRS in a general-sum game. The risk factor causes the strategy to gradually shift from the minimax solution to a best-response to the prediction. The prediction weight causes discontinuous changes in the strategy at points where a best-responding opponent changes strategies. It can be proven that when the strategy changes at a point w the ratio of gain against the prediction to the loss against a best-responding opponent is $\frac{1-w}{w}$.

Using the method outlined in [McCracken and Bowling, 2004] to set r values and a combination of gradient descent with an exponential opponent response model to set w values RSRS can perform well using a flawed predictor against a wide variety of opponents. It is able to coordinate in self-play to arrive at strategies which are beneficial to both agents. However, it is not fully cooperative. RSRS will never select a dominated strategy, so it can't cooperate in games such as Prisoner's Dilemma, where the cooperative move is strictly dominated.

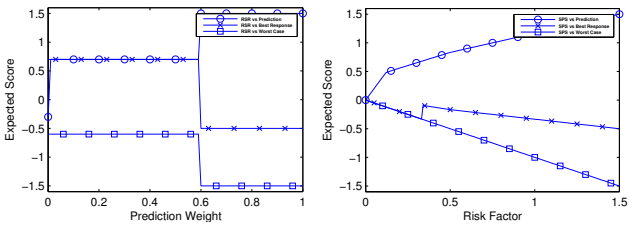

Figure 3: Performance of the w (left) and r (right) parameters in a general-sum game, against the prediction($\circ$), against a best-responder($\times$), and in the worst-case($\square$).

3 Future Work

We can use attitude to cooperate in self-play at a cost of not responding rationally to our prediction. We can use RSRS to respond effectively to a prediction without cooperating. The next stage is to incorporate RSRS into an attitude-based reciprocating agent to create an agent which can use reciprocation to achieve a cooperative outcome in any general-sum game while avoiding exploitation. We will do this by creating a particle filter which combines attitude and the RSRS model to identify attempts to cooperate.

The final problem is to formalize the process of reciprocation. In symmetric games, it's easy to aim for equal outcomes, but not every game is symmetric, and even when the game appears to be symmetric inaccuracies in the utility function may make an asymetric outcome desirable. For example, in a game with cash payoffs one player may have an immediate need for a specific amount while the other simply wants as much money as possible - this would affect the achievable cooperative agreements. We don't believe it's possible to fully resolve this problem in the absence of an oracle to provide true utility functions, but formalizing the tradeoffs being made will be an important step.

References

[Bowling, 2005] Michael Bowling. Convergence and no-regret in multiagent learning. *Advances in neural information processing systems*, 17:209–216, 2005.

[Frohlich, 1974] N. Frohlich. Self-Interest or Altruism, What Difference? *Journal of Conflict Resolution*, 18(1):55–73, 1974.

[Johanson *et al.*, 2007] Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In *Advances in Neural Information Processing Systems (NIPS)*, pages 721–728, 2007.

[McCracken and Bowling, 2004] Peter McCracken and Michael Bowling. Safe strategies for agent modelling in games. In *AAAI Fall Symposium on Artificial Multi-agent Learning*, October 2004.