On Partially Controlled Multi-Agent Systems

By: Ronan I. Brafman and Moshe Tennenholtz

Presentation By:
Katy Milkman

CS286r - April 12, 2006
Partially Controlled Multi-Agent Systems (PCMAS)

- **Controllable Agents**: agents that are directly controlled by a system’s designer (e.g. punishing agents, conforming agents)
- **Uncontrollable Agents**: agents that are not under the system designer’s direct control
- **PCMAS**: systems containing some combination of controllable and uncontrollable agents
- **Design Challenge in PCMAS**: ensuring that all agents in the system behave appropriately through adequate design of the controllable agents
The Purpose of This Paper

• Suggest techniques for achieving satisfactory system behavior in PCMAS through the design of controllable agents

• Examine two problems in this context:
  – The problem of enforcing social laws in partially controlled multi-agent systems
  – The problem of embedded teaching of reinforcement learners in PCMAS where the teacher is a controllable agent and the learner is an uncontrollable agent
Problem 1: Enforcing Social Laws in PCMAS
Problem 1: Motivating Example

You have been hired to design a new working environment for artificial agents. Part of your job involves designing a number of agents that will use and maintain a warehouse. Other agents, designed by different designers, will be using the warehouse to obtain equipment. To make sure that different agents, designed by different designers, can operate efficiently in this environment, you choose to introduce a number of social laws, i.e., constraints on the behavior of agents, that will help the agents coordinate their activities in this domain. These rules include a number of ‘traffic laws’, regulating motion in the domain, as well as a law that specifies that every tool that is used by an agent must be returned to its designated storage area. Your robots are programmed to follow these laws, and you expect the others to do so. Your laws are quite successful, and allow efficient activity in the warehouse, until a new designer arrives. Pressed by his corporate bosses to deliver better performance, he decides to exploit all your rules. He designs his agent to locally maximize its performance, regardless of the social laws. What can you do?
Problem 1: The Game

- Uncontrollable agents and controllable agents face each other in an infinite sequence of two-player games
  - Given n uncontrollable and controllable agents, this is an n-2-g game.
- Uncontrollable agents never know what type of opponent they face (i.e. punishing, or conforming) in a given game
- Players are randomly matched in each game
Problem 1: A Few Assumptions

• The system designer’s goal is to maximize the joint sum of the players’ payoffs
  – The strategy that achieves this is called efficient
• Agent utility is additive
• The system is symmetric
• Uncontrollable agents are “rational” expected utility maximizers
Problem 1: The Strategy

• Assume that the system designer controls a number of reliable agents
• Design these reliable agents to punish agents that deviate from the desirable social standard
• Hard-wire this punishment mechanism into the reliable agents and make it common-knowledge
• Design this punishment mechanism so that deviations from the social standard are irrational for uncontrollable agents (assuming these agents are expected utility maximizers)
Problem 1: The Benefits of Reliable, Programmable Agents

- The fact that the designer can pre-program his agents means that he can make any punishment, however “crazy,” a credible threat.
- This gets around the problem from traditional game theory that a threat may only be credible if it is an SPNE.
- This idea ties into Schelling’s Nobel Prize winning work, which acknowledges that a player can sometimes strengthen his position in a game by limiting his options. In other words, by committing through some means (such as programming agents) to play a response that might not be credible without commitment, a player can improve his situation.
- Because programmable agents allow the designer to make credible threats, punishments never actually have to be executed!
Problem 1: When it is Solvable?

• **Minimized Malicious Payoff**: the minimal expected payoff of the malicious players that can be guaranteed by the punishing agents
  – This is just the minimax payoff!

• **When Are Social Laws Enforceable?**
  A punishment is said to “exist” when each uncontrollable agent’s minimized malicious payoff is lower than the expected payoff he would obtain by playing according to the social law
Problem 1: Theorem 1

Theorem 1: Given an n-2-g iterative game, the minimized malicious payoff is achieved by playing the strategy of player 1 prescribed by the Nash Equilibrium of the projected game, \(^1\) \(g_p\), when playing player 1 (in \(g\)), and the strategy of player 1 prescribed by the Nash Equilibrium of the projected game \((g^T)p\) when playing player 2 in \(g\). \(^2\)

\(^1\)The projected game of \(g\), \(g_p\), is a game where the first agent’s payoff equals the opposite of the second agent’s payoff in the original game. This is just a zero-sum game constructed to reflect the payoffs to player 2.

\(^2\)The transposed game of \(g\), \(g^T\), is a game where players’ roles are switched.

This theorem just says: minimax = NE in zero-sum games (we knew this already!)
Problem 1: Corollary 1

**Corollary 1** Let n-2-g be an iterative game, with p punishing agents. Let v and v’ be the payoffs of the Nash equilibria of gp and gp^T respectively (which, in this case, are uniquely defined). Let b,b’ be the maximal payoffs player 1 can obtain in g and g^T respectively, assuming player 2 is obeying the social law. Let e and e’ be the payoffs of player 1 and 2, respectively, in g, when the players play according to the efficient solution prescribed by the social law. Finally, assume that the expected benefit of two malicious agents when they meet is 0. A necessary and sufficient condition for the existence of a punishing strategy is that:

\[
(\mathbb{M} - I - \mathbb{I}) \cdot (\mathbb{P} + \mathbb{P}_i) - \mathbb{I} \cdot (\mathbb{N} + \mathbb{N}_i) < (\mathbb{S} + \mathbb{S}_i).
\]

(Expected Utility for Malicious Agent < Expected Utility Guaranteed by Social Law)
Problem 1: 
Prisoner’s Dilemma Example

• **Design Goal:** convince uncontrolled agents to “cooperate”

• **Maximal expected loss for an uncontrolled agent that a punishing agent can guarantee:**
  \[
  7 = (2 - (-5))
  \]
  ▪ if punishing agents play “defect”

• **Gain uncontrolled agent expects when playing an agent who follows the social law:**
  \[
  8 = (10 - 2)
  \]

• For a punishing strategy to be effective, it must hold that:
  \[
  \text{สมมุติฐาน} - \text{สมมุติฐาน} > 8
  \]

• Given a choice between: (a) fewer punishers and harsher punishments and (b) more punishers and gentler punishments, it is better to have fewer punishers and harsher punishments.
Problem 2: Embedded Teaching of Reinforcement Learners in PCMAS
Problem 2: Motivating Example

As an example, consider two mobile robots without any means of direct communication. Robot 1 is familiar with the surroundings, while Robot 2 is not. In this situation, Robot 1 can help Robot 2 reach its goal through certain actions, such as blocking Robot 2 when it is headed in the wrong direction. However, Robot 1 may have only limited control over the outcome of such an interaction because of uncertainty about the behavior of Robot 2 and its control uncertainty. Nevertheless, Robot 2 has a specific structure, it is a learner obeying some learning scheme, and we can attempt to control it indirectly through our choice of actions for Robot 1.⁷
Problem 2: The Strategy

• Assume that the system designer controls a single agent, the teacher

• If possible, design the teacher to always choose the action (or to always play according to the mixed strategy) that will make the desired action most appealing to the student
Problem 2: A Few Assumptions

• Assume that the system designer’s goal is to maximize the number of periods during which the student’s actions are as desired (other goals might be interesting to investigate too)
• Assume the teacher does not know when the game will terminate
• Assume there is no cost of teaching (is this reasonable?)
• Assume the student can be in a set of $\Sigma$ possible states, his set of actions is $A_s$, and the teacher’s set of actions is $A_t$
• Assume the student’s state at any time is a function of his old state, his current action, and the teacher’s current action
• Assume the student’s action is a stochastic function of his current state, where the probability of choosing “a” at state “s” is $p(s,a)$
• Assume the teacher knows the student’s state, state space, and policy
  – This assumption is relaxed in later experiments
• NOTE: agent rationality is no longer assumed
Problem 2: When is it Solvable?

- Given a two-player game where both players have two actions available to them, assume (for this example) that the teacher’s goal is to teach the student to play action 1:

  - **Case 1**: If $a > c$ and $b > c$, any teaching strategy will work (desired strategy is strictly dominant)

  - **Case 2**: If $a > c$ or $b > d$:
    - **preemption**: the teacher always chooses the action that makes action 1 look better than action 2 to the student
    
  - **Case 3**: If $c > a$, $c > b$, $d > a$, and $d > b$, teaching is impossible

  - **Case 4**: Otherwise, teaching is possible but preemption won’t work (e.g. Prisoner’s Dilemma)

**Case 4 is the focus of this section.**
Problem 2: Optimal Teaching Policies

- $u(a) =$ the value the teacher places on a student’s action, $a$
- $\pi =$ the teacher’s policy
- $Pr_{\pi,k} =$ the probability distribution over the set of possible student actions at time $k$ induced by the teacher’s policy
- The discounted expected value of the student’s actions ($val(\pi)$) = $\sum_{k=0}^{\infty} \gamma^k E_k(u)$
- The expected value of $u$ ($E_k(u)$) = $\sum_{a \in A_s} Pr_{\pi,k}(a) \cdot u(a)$
- View teaching as an MDP
- The teacher’s goal is to find a strategy, $\pi$, that maximizes $val(\pi)$. This is just a dynamic programming problem, and it happens to have a unique solution, $\pi^*$. 
**Problem 2: Theorem 2**

**Theorem 2** The optimal teaching policy is given by the $\gamma_0$ optimal policy in TMDP $= \langle \Sigma, A_t, P, U \rangle$.

Given:  

$$P(s, s', a_t) \overset{\text{def}}{=} \sum_{a_s \in A_s} \rho(s, a_s) \cdot \delta_{s', \tau(s, a_s, a_t)}$$  

$$U(s) \overset{\text{def}}{=} \sum_{a_s \in A_s} \rho(s, a_s) \cdot u(a_s)$$

The $\gamma_0$ optimal policy in TMDP is the policy $\pi$ that for each $s \in \Sigma$ maximizes:

$$\sum_{k=0}^{\infty} \gamma_0^k \left( \sum_{s' \in \Sigma} P_{s, \pi, k}(s') \cdot U(s') \right)$$

This policy can be used for teaching when the teacher can determine the current state of the student. When the teacher cannot determine the current state of the student, this policy can be used to calculate an upper bound on the success $\text{val}(\pi)$ of any teaching policy $\pi$.  

The probability of a transition from $s$ to $s'$ under $a_t$ is the sum of the probabilities of the student’s action that will induce this transition.
Problem 2: Experiments with a Prisoner’s Dilemma Game

- **Blind Q-Learner (BQL):** learner can perceive rewards but cannot see how the teacher has acted or remember his own past actions
  - update rule: $q_{new}(a) = (1 - \alpha) \cdot q_{old}(a) + \alpha \cdot R$

- **Q-Learner (QL):** learner can observe the teacher’s actions, has a number of possible states that encode past joint actions, and maintains a Q-value for each state-action pair
  - update rule: $q_{new}(s, a) = (1 - \alpha) \cdot q_{old}(s, a) + \alpha \cdot (R + \gamma V(s'))$
Problem 2: Experiments with a Prisoner’s Dilemma Game

• Both types of students choose their actions based on the Boltzmann distribution, which associates a probability $P_s(a)$ with the performance of an action $a$ at a state $s$.

\[
P_s(a) \stackrel{\text{def}}{=} \frac{\exp(q(s, a)/T)}{\sum_{a' \in A} \exp(q(s, a')/T)} \quad (QL)
\]

\[
P(a) \stackrel{\text{def}}{=} \frac{\exp(q(a)/T)}{\sum_{a' \in A} \exp(q(a')/T)} \quad (BQL)
\]

• Agents explore the most at high values of $T$, and their actions become stickier (making Q-values play a greater role in their decisions) as $T$’s value drops.
Problem 2: Experiments with a Prisoner’s Dilemma Game

- The Prisoner’s Dilemma game used in experiments:

<table>
<thead>
<tr>
<th></th>
<th>Cooperate</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td>(10, 10)</td>
<td>(-13, 13)</td>
</tr>
<tr>
<td>Defect</td>
<td>(13, -13)</td>
<td>(-6, -6)</td>
</tr>
</tbody>
</table>
Problem 2: What’s Wrong With These Reinforcement Learners?

• There is a serious computational complexity problem here:
  – A discretized representation of the BQL’s state space requires 40,000 states
  – Solving this problem with 40,000 states took 12 hours (in 1995)
  – A discretized representation of the simplest QL’s state space requires $10^{18}$ states
  – Solving this problem was not possible
Problem 2: The BQL Experiment

Figure 2: Fraction of Coops as a function of temperature for the approximately optimal policy (left) and for “teaching” using an identical Q-learner (right). Each curve corresponds to Coop rate over some fixed number of iterations. In the approx. optimal policy the curves for 1000, 5000 and 10000 iterations are nearly identical.
Problem 2: The BQL Experiment

Figure 3: Fraction of Coops as a function of temperature for the teaching strategy based on TFT (left) and 2TFT (right).
Problem 2: The QL Experiment – A Workaround

• As learners get more sophisticated, they are easier to teach. Because they remember their past actions and outcomes, they can recognize patterns of punishments.

• The authors argue that a tit-for-tat strategy should work quite well against a Q-learner with one memory state:

“While the immediate reward obtained by a QL playing defect may be high, he will also learn to associate a subsequent punishment with the defect action.” (page 22)
Problem 2: The QL Experiment

Figure 5: Each curve shows the fraction of Coops of QL as a function of temperature for a fixed number of iterations when TFT was used to teach (left) and when an identical Q-learner was used to teach (right). Values are means over 100 experiments.
**Problem 2: Block Pushing Example**

- In some games, a teacher wants to maximize a function that depends *both* on his behavior and on his student’s behavior.

- Example: A teacher and a student must push a block as far as possible. Pay is based on how far the block is pushed, and each agent may choose to push hard (expending lots of energy) or gently (saving energy) in each time period.

<table>
<thead>
<tr>
<th>Teacher</th>
<th>Student</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>hard</td>
<td>(3, 3)</td>
<td>(2, 6)</td>
</tr>
<tr>
<td>gentle</td>
<td>(6, 2)</td>
<td>(1, 1)</td>
</tr>
</tbody>
</table>

- A simple teaching strategy would be for the teacher to always play “gentle,” but this would not maximize the joint outcome.
Problem 2: Block Pushing Example

- Model the student as a BQL
- Teacher strategy in plot: push gently for $K$ iterations and then start to push hard
- Results that would be obtained with a “naïve” teaching strategy – 10,000 hard push instances in 10,000 iterations
- Results that would be obtained if two reinforcement learners played against each other – 7,618 hard push instances in 10,000 iterations

Figure 8: Teaching to push hard

results averaged over 50 trials
Problems With This Paper

- The theorems presented are all for two-player games. What about games with more players?
- Representing all states in even the simplest QL space is problematic, so how do the theorems about optimal policies help us with QLs?
- Many assumptions are made, some of which are very restrictive (e.g. known payoffs). What happens if some are relaxed/altered?
- Only two example problem domains are examined.
- This paper’s “general findings” essentially tell us what we already know: in a repeated game, you should use a combination of punishments and rewards to manipulate your opponents’ actions.
- On page 8, the authors note that the design of punishers may be complex, but they do not discuss this further.
- The authors should have discussed the Folk Theorems and tied their work directly to the relevant game theory literature.
Conclusions

• An interesting twist on the AI learning literature – how do we design mechanisms to manipulate agents that learn?
• In the first part of this paper (enforcement of social laws), there is no learning.
• In the second part of this paper (embedded teaching of reinforcement learners), there is no game theory.
• Even the authors acknowledge that their work only beings to examine the world of PCMAS, that more domains need to be explored, and that it would be ideal if more general conclusions could be drawn in the future.