The City College of New York

CSC 59866-E: Senior Project I *AI Agents for Decision Making in the Real World*

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu, sbandyopadhyay@gc.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

Game Theory for AI Agents

Logistics and Motivation for Control Theory

Recall: We looked at Model Predictive Control (MPC) to guarantee safety and strict constraints for a single agent operating in a physical environment.

What happens when there are multiple intelligent agents in the same environment, and their actions mutually affect each other's success?

Today’s Agenda

Game Theory Fundamentals: Formalizing multi-agent environments.

Paradigms of Interaction: Pure Competition vs. Pure Coordination.

Nash Equilibrium: The mathematical core of AI strategy.

Interactive Problem: Solving a Mixed Strategy Equilibrium live.

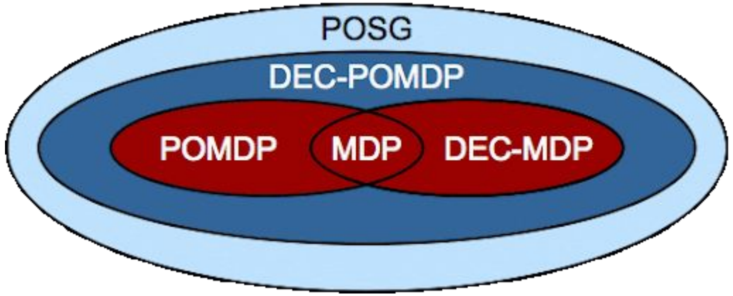
Fundamental Algorithms for AI Agents: How AI Agents actually learns these equilibria (Fictitious Play & CFR).

Formalizing a “Game”

What is the "Game" in "Game Theory"?

Recall:

- **Single-Agent RL (MDPs):** The agent takes an action, the environment transitions to a new state, and gives a reward. The environment is stationary.
- **Multi-Agent RL (Games):** The "environment" now contains other thinking, adapting agents.
- **The Core Issue:** You cannot optimize your reward without predicting what the other agents are going to do. And they are simultaneously trying to predict what *you* are going to do!



Normal-Form Game

To solve these interactions, we formalize them mathematically as a **Normal-Form Game**. A game is defined by a tuple: $\langle N, A, u \rangle$

Players (N): A finite set of agents $N = \{1, 2, ..., n\}$.

Action Space (A): $A = A_1 \times A_2 \times ... \times A_n$. This represents all possible joint actions. A specific joint action is $a = (a_1, ..., a_n)$.

Utility Function (u): $u_i(a)$ gives the payoff for player i when the joint action a is played.

Notation Trick: We often write a joint action from the player i perspective as (a_i, a_{-i}) , where a_{-i} means "the actions of absolutely everyone else except me".

Payoff Matrices

For 2-player games, we visualize $\langle N, A, u \rangle$ using a **Payoff Matrix.**

Row Player (Player 1) chooses the row. **Column Player (Player 2)** chooses the column.

The cell contains the tuple of utilities: (u_1, u_2) .

Example: If Row chooses "Top" and Column chooses "Left", we look at the top-left cell. If it says $(3, -1)$, Player 1 gets a reward of 3, and Player 2 gets a reward of -1.

Pure Competition (Zero-Sum Games)

The Paradigm: What is good for me is strictly bad for you. Our goals are perfectly misaligned.

The Math: For any joint action a, the sum of all utilities is zero.

$$\sum_{i \in N} u_i(a) = 0$$

(Or $u_1(a) = -u_2(a)$ *for 2 players).*

Coordination and Mixed-Motives

Pure Coordination (Common Payoff): Agents share the exact same utility function $u_1(a) = u_2(a)$ .The only challenge is communicating or agreeing on the best joint action. (e.g., two robots carrying a heavy table).

Mixed-Motive (General Sum): The most complex and realistic paradigm. Agents have independent goals, but can benefit from temporary cooperation.

The Classic Failure - Prisoner's Dilemma: Even if mutual cooperation yields a high reward for both $(3, 3)$, individual rationality forces both agents to defect, leading to a terrible outcome $(1, 1)$. How do we program AI to avoid this trap?

The Nash Equilibrium

Nash Equilibrium

How do we define a "solved" game? We use the **Nash Equilibrium**.

Definition: A joint strategy profile π^* is a Nash Equilibrium if no player has an incentive to *unilaterally* deviate from it.

The Mathematical Condition: For all players i , and all possible alternative actions a_i :

$$u_i(\pi_i^*, \pi_{-i}^*) \geq u_i(a_i, \pi_{-i}^*)$$

Intuition: If you know exactly what your opponent is going to do, and you still wouldn't change your move, you are in a Nash Equilibrium.

Pure vs. Mixed Strategies

Pure Strategy: The agent chooses a single action deterministically (100% chance to play "Rock").

The Problem: Many games (like Rock-Paper-Scissors) *do not have* a Pure Strategy Nash Equilibrium. If you always play Rock, I will just play Paper.

Mixed Strategy (π): The agent plays a probability distribution over its available actions.

Nash's Theorem: *Every* game with a finite number of players and actions has at least one Mixed Strategy Nash Equilibrium.

Interactive Example

The Problem

The Game: A non-symmetric zero-sum game (like a penalty kick in robot soccer).

Row Player (P1) plays *Up* with probability p , and *Down* with probability $(1-p)$.

Column Player (P2) plays *Left* with probability q , and *Right* with probability $(1-q)$.

The Payoff Matrix (P1, P2):

(Up, Left) = (2, -2); (Up, Right) = (-3, 3); (Down, Left) = (-1, 1); (Down, Right) = (2, -2)

Your Task: Find the exact probabilities p and q that form the Nash Equilibrium.

Solution

Step 1: Make Column Player (P2) indifferent. Calculate P2's expected payoff for playing *Left* vs *Right*, based on P1's probability p.

- Expected value of *Left*: $-2(p) + 1(1-p) = 1 - 3p$
- Expected value of *Right*: $3(p) - 2(1-p) = 5p - 2$
- Equate them: $1 - 3p = 5p - 2 \implies 8p = 3 \implies \mathbf{p = 3/8}$

Solution

Step 2: Make Row Player (P1) indifferent. Calculate P1's expected payoff for playing *Up* vs *Down*, based on P2's probability q.

- Expected value of *Up*: $2(q) - 3(1-q) = 5q - 3$
- Expected value of *Down*: $-1(q) + 2(1-q) = 2 - 3q$
- Equate them: $5q - 3 = 2 - 3q \implies 8q = 5 \implies \mathbf{q = 5/8}$

The Result: P1 must play Up 37.5% of the time, and P2 must play Left 62.5% of the time. This is the Nash Equilibrium!

Fundamental Game Theoretic Algorithms for AI Agents

How Do AI Agents Find the Equilibrium?

In our class problem, we used algebra to *calculate* the equilibrium.

The Reality: For massive games like Poker or Autonomous Driving, the matrices have trillions of rows. We cannot use algebra.

The AI Approach: Agents must *learn* the equilibrium through iterative self-play algorithms. They repeatedly play the game, update their strategies based on the outcomes, and gradually converge to the Nash Equilibrium.

Fictitious Play

One of the oldest and most fundamental multi-agent learning algorithms.

The Mechanism:

1. Each agent keeps a historical count of every action its opponent has ever taken.

2. The agent forms an empirical probability distribution (a "belief" about the opponent).

3. The agent calculates the Best Response to that belief and plays it.

Convergence: In zero-sum games, if both agents use Fictitious Play against each other, their empirical action frequencies are mathematically guaranteed to converge to the Nash Equilibrium!

Counterfactual Regret Minimization (CFR)

The Problem with Fictitious Play: It struggles with *Sequential* games (games with turns, like Poker, rather than simultaneous matrices).

CFR: The gold-standard algorithm that solved Heads-Up No-Limit Texas Hold'em.

The Intuition: Instead of tracking opponent histories, the agent tracks its own **Regret**. "How much better would I have done if I had played *Fold* instead of *Raise* across all past games?"

CFR - Regret Matching

At step T, the regret for not taking action a is the difference between the utility of a and the utility of the action actually taken:

$$R_i^T(a) = \sum_{t=1}^{T} \left(u_i(a, a_{-i}^t) - u_i(a^t, a_{-i}^t) \right)$$

Regret Matching Strategy: The agent updates its policy π for the next game by making the probability of taking an action directly proportional to its positive accumulated regret:

$$\pi_i^{T+1}(a) = \frac{\max(0, R_i^T(a))}{\sum_{a'} \max(0, R_i^T(a'))}$$

Actions that you deeply regret *not* taking in the past get played more often in the future!

Summary + Relevance to Projects

Game Theory provides the strict mathematical framework for multi-agent interactions.

Agents must balance **Pure Competition (Zero-Sum)** and **Coordination (General-Sum)**.

The **Nash Equilibrium** is the target state where no agent can unilaterally improve.

Algorithms like **Fictitious Play** and **CFR** allow AI to converge on these equilibria through self-play, avoiding impossible algebraic calculations.

For your Senior Projects: If your project involves multiple agents (e.g., decentralized routing, autonomous vehicle networks), you must identify your game paradigm. Are your agents fully cooperative, or competing for bandwidth/space?

Questions?