Fairness and Exploitation in the $N$-player Ultimatum Game

Garrison Greenwood, Portland State University, Portland, OR, USA
Daniel Ashlock, University of Guelph, Guelph, Ontario, Canada
The **ultimatum game** (UG) is a sequential game devised by economists to test whether maximum individual payoffs equate to maximum utility.

The 2-player game is rather simple. One player is a *proposer* and the other player is a *responder*.

The proposer offers how to split some monetary amount and the responder then accepts or rejects the offer.

If accepted, the money is split as proposed but if rejected both players receive nothing.

A “fair” offer would be a 50-50 split.
basic game format

Two players. In each round a player could be a proposer or a responder (role randomly chosen).

Each player has a strategy $S(p,q)$ where $p, q \in [0,1]$.

The proposer offers $p$ (a percent of the amount to be split) that would go to the responder. The responder has a $q$ value, which is the minimum acceptable offer. Any offer $p \geq q$ is accepted and any offer $p < q$ is rejected.

It is not necessary that $p+q=1$ and $q > p$ is okay.
player $a$: $p_a = 0.3$  $q_a = 0.1$
player $b$: $p_b = 0.2$  $q_b = 0.4$

money to split: $100$

Example 1: (player $a$ proposer; player $b$ responder)

$p_a < q_b$ player $b$ rejects the offer; both players get 0

Example 2: (player $b$ proposer; player $a$ responder)

$p_b \geq q_a$ player $a$ accepts the offer; gets $20$, player $b$ gets $80$
What would a rational player do??

A rational *proposer* would offer as little as possible to maximize his payoff.

A rational *responder* would accept even a small offer because getting something is better than getting nothing.

Yet, human experiments show irrational behavior is the norm. Roughly half of the players considered offers below 30% unfair and rejected them. Nevertheless, offers as low as 30% were routinely accepted.
In the UG, the inevitable outcome is $p,q \to \varepsilon$. The smallest proposal that is accepted is the only sub-game perfect Nash Equilibrium.

At least that's what the models predict....

But human experiments indicate players act irrationally in practice.

They tend to reject low offers because they consider them unfair.
This illustrates a fundamental difference between social dilemmas (e.g., prisoner’s dilemma) and ultimatum bargaining games.

Irrational behavior in social dilemmas is the growth in cooperation.

In UGs it is the growth of fairness.

Note: “fairness” in UGs is said to exist in a population if in player interactions, on average, $p > q$.

**Fundamental question:** under what conditions will fairness persist in an UG?
Game Trees

Game theory provides a framework for getting some answers.

In our case in each round we must choose how \( p \) and \( q \) should change (or maybe even not change) to promote fairness.
Even more importantly, how do we find $p$ and $q$ values that promote fairness in a population of $N > 2$ players?

Rather than develop some formula or equation for determining $p$ and $q$ we used an *evolutionary algorithm* to search for good values.

Evolutionary algorithms model Darwinian evolution found in nature.

Essentially, they search for good solutions to difficult optimization problems by *evolving* the solutions.
(μ,λ) Evolution Strategy
Virtually all researchers use a heaviside function to determine whether to accept an offer.
This sigmoid type of function models the likelihood of accepting a slightly less offer.

\[
\text{prob(accept)} = \begin{cases} 
1.0 & p \geq q \\
\exp(-\alpha(p-q)^2) & p < q 
\end{cases}
\]
The amount to be split is set to 1. A player’s strategy is encoded in a genotype containing two real parameters $p, q \in [0, 1]$.

$$S(p,q)$$

The evolutionary algorithm used $\mu = 10$ parents to produce $\lambda = 80$ children.

The $p$ values from parents are averaged to create a "center of mass" parent $p_m$. Similarly the $q$ values are averaged to create $q_m$.

The $i$-th offspring is generated by

$$p_i \sim N(p_m, 0.2) , \quad q_i \sim N(q_m, 0.2)$$

where $N(\eta, \sigma)$ represents a normally distributed random variable with mean $\eta$ and standard deviation $\sigma$. 
Offspring are evaluated in tournaments and the proposer/responder roles in the interactions are randomly chosen.

The $\mu$ parents in the current generation form the tournament set.

The accumulated payoff from the tournament is the offspring’s fitness.

The evolutionary algorithm ran for 150 generations (iterations) and results are averaged over 20 runs.
Consequently, some offspring may see a majority of tournament members as proposers while others may see the same tournament members as responders.

A strategy with high $p, q$ values would probably die if most tournament set interactions are as a responder.

However, just the opposite happens if most interactions are as a proposer.

This survivor carries the high $q$ value into the next generation which would help drive the population away from the $S(0,0)$ rational strategy.
$\alpha = 25$
Lessons Learned

If you want to promote fairness in an UG,

• Don’t expect unrealistic returns. That is, choose a realistic $q$ value

• Be flexible. Accept “reasonably close” proposals

• But don’t be too flexible to avoid exploitation
Now we're going to add one more player ($N=41$)

This new player also has the same genome

$$S(p,q)$$

But this player is different. He is **self-interested**.

Self-interested players are greedy; they put their interests above those of others.

Thus, this new player wants to **exploit** the other 40 players by maximizing his profit.
$$S(p, q)$$

How do we set the $p$ and $q$ values for this new player???

Obviously cannot use the evolution strategy used by the other players....

$q = 0$ because getting something is better than getting nothing.

To set the $p$ value we will use a technique called

**Monte Carlo Tree Search (MCTS)**
MCTS tries to find optimal decisions by building a search tree in a given problem domain.

MCTS is an AI method which has had a profound impact in problem domains such as games and planning problems.

Indeed, it can be used in any domain that can be represented as trees of sequential decisions.
A best of five game series, $1 million dollars in prize money. Between 9 and 15 March, 2016, the second-highest ranked Go player, Lee Sidol, took on a computer program named AlphaGo designed by Google.

AlphaGo won the series 4-1.
The selection function is applied recursively until a leaf node is reached.

A node is created.

One simulated game is played.

The result of this game is backpropagated in the tree.
The multi-armed bandit problem (MAB)

- You are in a room with $k$ slot machines.
- at time $t$ pick you pull the arm on machine $i_t$
- if you play arm $i_t = i$, you collect a a payoff or reward $r_t(i)$
- Goal: maximize your accumulated reward (after $T$ trials)

$$\text{cumulative reward} = \sum_{t=1}^{T} r_t(i_t)$$

Can think of a slot machine as a "one-arm" bandit
Why is MAB so difficult??

- if you never try arm $i$, you'll never know how good (or bad) it is
  - should try each arm at least once
  - probably more than once (can't learn much with one pull...)

- if you think arm $j$ is best
  - maybe play only arm $j$....
  - but what if arm $m \neq j$ is best????
  - every time you didn't pull the best arm, you feel some regret
So what does MAB have to do with MCTS??

![Diagram showing a tree structure with a current state at the top and move options below.](image-url)
If we have a policy for picking the best arm in a MAB, then we can use that same policy to pick the next move in a MCTS iteration!!

To describe such a policy we need to introduce the idea of **Regret**.

Recall in each iteration of the MCTS we pick the next move with some probability.

Regret measures our disappointment in not picking the best arm
CUMULATIVE REGRET

Let $\xi_i \in [0,1]$ be the expect payoff from machine $i$

\[
CR_n = \sum_{t=1}^{n} (\xi^* - \xi_{I(t)}) \quad \text{where} \quad \xi^* \overset{\text{def}}{=} \max_{1 \leq i \leq K} \xi_i
\]

and $I(t)$ is the arm pulled in round $t$. $CR_n$ expresses disappointment in not picking the best machine over the $n$ rounds.

SIMPLE REGRET

Let $J(n)$ be the recommended best arm to pull after all rounds are played

\[
SR_n = \xi^* - \xi_{J(n)}
\]

only reflects regret for not recommending best arm to pull.

NOTE: no sampling technique minimizes simple regret and cumulative regret at the same time.
Upper Confidence Bound (UCB1)

picks the arm $j$

$$j = \arg \max_i \left( \bar{X}_i + c \sqrt{\frac{\ln(n)}{n_i}} \right)$$

- **exploitation term**: $\bar{X}_i$
- **exploration term**: $\sqrt{\frac{\ln(n)}{n_i}}$

- $\bar{X}_i$ is the sample mean of arm $i$ with support $[0,1]$
- $n$ is the number of plays so far
- $n_i$ is the number of times arm $i$ has been played
Upper Confidence Bound for Trees (UCT)

$$\text{UCB1} = \bar{X}_j + c \sqrt{\frac{\ln(n)}{n_j}}$$

This policy selects child $j$ such that

$$j = \arg \max_i \left( \bar{X}_i + c \sqrt{\frac{\ln(n_p)}{n_i}} \right)$$

- $n_p$ is the number of times parent visited
- $n_i$ the number of times child $i$ visited
- $c$ balances exploitation and exploration
Simple Regret (\( \text{UCB}^{\sqrt{\cdot}} \))

\[ j = \arg \max_i \left( \bar{X}_i + c \sqrt{\frac{\sqrt{n_p}}{n_i}} \right) \]

has higher UCB (better exploration than UCT)
Simple Regret

$\epsilon$ – greedy

samples the arm with the largest sample mean $\bar{X}$ with probability $1 - \epsilon$ and a random arm with probability $\epsilon$

With $\ell$ children nodes the selection probability is

$$p = \begin{cases} 
1 - \epsilon + \epsilon/\ell & \text{highest sample mean node} \\
\epsilon/\ell & \text{other nodes}
\end{cases}$$

Has no exploration and exploitation tradeoff

Sub-optimal nodes are selected with some non-zero probability regardless of their sample mean. It therefore promotes more exploration than UCT.
where $\Delta p$ is a random variable between 2% and 5% of $p$
The *tree policy* controls how the game tree is partially expanded.

Studies indicate a two-stage approach is best:

- sample at the root node to minimize simple regret
- sample all other nodes to minimize cumulative regret

![Game tree diagram]

- UCT
- $\epsilon$-greedy+UCT
- $\text{UCB}^\sqrt{+}\text{UCT}$
for some constant $\phi \geq 0$, after $n$ visits the expected simple regret is bounded from above by

$$\epsilon - \text{greedy : } \mathbb{E}r_n \leq O(e^{-\phi n})$$

$$\text{UCB} \sqrt{ } : \quad \mathbb{E}r_n \leq O(e^{-\phi \sqrt{n}})$$

$$\text{UCT : } \quad \mathbb{E}r_n \leq O(n^{-\phi})$$

UCT simple regret decreases only polynomially at best with $n$ whereas the other two sampling methods decrease exponentially.
Rollout

A rollout consists of using the $p$ value of the newly expanded node in a tournament against $\mu = 10$ randomly selected offspring out of the current population.

The expanded game tree node always acts as the proposer.

A total of 200 rollouts were performed each UG round.

$$b = \sum_{s=1}^{\mu} \mathbb{1}(p > q_s)$$ number of times an offer is accepted

Then the value backpropagated is

$$\Delta = \begin{cases} 
0 & \text{if } p < q_s \ \forall s = 1 \ldots \mu \\
1.0 - \frac{\sum (p - q_s)}{b} - (\mu - b)\gamma & \text{otherwise}
\end{cases}$$

$\Delta \in [0,1]$
Results

Each offspring and each MCTS version player was evaluated via a tournament.

The $\mu$ parents in each round of the evolutionary algorithm formed the tournament set.

Proposer/responder roles were randomly chosen in the tournament.

The payoffs of the $\lambda$ offspring were averaged each round and compared against each MCTS version player.
Conclusions

- MCTS is quite successful in creating a rational (greedy) UG player that exploits a population of fair players
- the two phase approach was better than the UCT-only approach which is consistent with previous studies
- the $\epsilon$ – greedy + UCT version is most responsive of the 3 MCTS versions. It has the smallest upper bound on expected simple regret.
Future Work

To the best of our knowledge this is the first instance where MCTS is used for making strategy decisions in an economic or social dilemma game.

• In a 2-player IPD, defecting every other round is a good strategy. Can MCTS "discover" this strategy? (Good initial test)

• In a multi-player, multi-strategy IPD evolved finite state machines can find good strategies. Can MCTS be competitive against evolved strategies?

• In a public goods game where free riders are punished, can MCTS find acceptable contribution levels that maximizes payoffs while avoiding punishment?
Any questions?