Monte Carlo Strategies for Exploiting Fairness in N-player Ultimatum Games

Garrison W. Greenwood
Dept. of Electrical & Computer Engineering
Portland State University
Portland, OR 97207–0751 USA
Email: greenwd@pdx.edu

Daniel Ashlock
Dept. of Mathematics and Statistics
University of Guelph
Guelph, Ontario, Canada
Email: dashlock@uoguelph.ca

***Abstract*—The Ultimatum Game (UG) is studied to see how people respond in bargaining situations. In the 2-player version each round a player can be a proposer or a responder. As a proposer an offer is made on how to split a monetary amount. The responder either accepts or rejects the offer. If accepted, the money is split as proposed; if rejected both players get nothing. Studies have found over time the offers decrease but are still accepted (getting something is better than nothing) until a subgame perfect Nash equilibrium is reached where the lowest possible offer is accepted. In the N-player version the object is to see if the population can reach a state of fairness where, on average, offers are accepted. We have previously shown that a $(\mu/\mu, \lambda)$ evolution strategy can evolve offers and acceptance thresholds that promote fairness. In this paper we report an extension to this previous work. One player is added to the population who interacts in the same manner with the other N players. However, this new player is rational—i.e., he ignores fairness and instead exploits the other players by maximizing his payoffs. We used three different versions of Monte Carlo Tree Search (MCTS) to adaptively control this rational player's offer levels during the game. The results indicate payoffs for this player can be as much as 40% higher than the population average payoff. Our MCTS introduces a novel rollout approach making it ideally suited for the play of mathematical games.**

I. Introduction

Game theorists commonly believe humans act rationally in any situation where there are financial gains. Many studies involving social dilemmas—i.e., situations where individuals choose to cooperate for the benefit a group or to selfishly defect for personal gain—indicate mutual defection is the eventual outcome if cooperation or defection are the only strategy choices. The well-known Prisoner's Dilemma game (PDG) is frequently used to study social dilemmas. A number of PDG models suggest other dynamics, such as altruistic punishment, can increase cooperation levels in a population. Human experiments indicate altruism and emotions both strongly influence cooperation levels [1]. In particular, players observe the payoffs others receive and react accordingly in future encounters based on a perceived notion of fairness.

The Ultimatum Game (UG) was developed by economists to see if maximum individual payoffs produce maximum utility [2]. The UG models many real-world economic situations such as pay negotiations and pricing decisions. In its simplest form a proposer makes an offer on how to divide a monetary amount. A responder either accepts or rejects the offer. The money is divided as proposed if the offer is accepted but both individuals get nothing if the offer is rejected. In anonymous, one-shot UGs a self-interested proposer offers as little as possible because he then keeps the bulk of the money. Getting something is better than getting nothing so a self-interested responder accepts any nonzero offer. In iterated games rational behavior leads to the eventual state where the minimum possible offer is made and it is accepted. This condition is a subgame perfect Nash equilibrium [2].

There is ia fundamental difference between social dilemmas and N-player UGs. Irrational behavior in social dilemmas is the growth in cooperation whereas in an UG it is the growth of fairness. Here fairness in a population means responders have reasonable expectations on what constitutes a fair offer and proposers make offers likely to be accepted. Humans tend to reject unfair offers. For example, in one human experiment nearly half of the players rejected offers below 30% because they considered them unfair [3].

In our previous work [4] we used a $(\mu/\mu, \lambda)$ evolution strategy to evolve offers and acceptance levels for a population of $N = 40$ individuals. Our results indicate if responders are somewhat flexible in deciding if offers are reasonable, then fairness can rapidly evolve and will persist in the population.

In this paper we report an extension to that previous work. One additional player is added to the population. This new player has the same genome as the other players and competes with the other players in the same manner. However, this new player has a completely different objective: he is strictly self-interested. His objective is to exploit the other players trying to achieve fairness by maximizing his personal payoff. The question is how to find good offers and acceptance levels for this rational, self-interested player. As a responder it makes sense to accept any nonzero offer because getting some payoff is always better than getting nothing. However, as a proposer it is not so obvious how to choose the amount to offer. Ideally this offer would be as low as possible to maximize his payoff while not so low it is likely to be rejected. Clearly the $(\mu/\mu, \lambda)$ evolutionary strategy cannot be used to find a good offers because it searches for offers and acceptance levels that

978-1-7281-1884-0/19/$31.00

promote fairness in the entire population. The rational player has no intention of being fair.

We used three different versions of *Monte Carlo tree search* (MCTS) to find good offers for a rational, self-interested player. Our results indicate all three versions can find offers yielding significantly higher payoffs for this rational player than the population average. To the best of our knowledge this is the first time MCTS has been used to find strategies in an economic game.

The paper is organized as follows. Section II provides a brief overview of MCTS, discusses regret in multi-arm bandit problems and reviews some previous UG work. Section III describes our model but the results and discussion are deferred until Section IV. Finally, Section V discusses other mathematical games where MCTS may be beneficial.

II. Background & Prior Work

In this section a brief overview of MCTS and regret is given plus a review of our previous work using a $(\mu/\mu, \lambda)$ evolution strategy used to evolve fairness. More detailed information can be found in [5] and [4], respectively.

A. MCTS overview

The basic algorithm iteratively builds a search tree showing potential moves or strategy choices in the game. Each node represents a particular game state and its K immediate children represent next moves. Effectively each node and its immediate children can be thought of as a MAB problem. The iterative search continues until some computational budget such as an iteration constraint is achieved. At that point the best child of the tree root node is deemed the best possible next move from the current game state.

The search consists of four steps [5]. Starting at the tree node a selection policy recursively works down the game tree until an expandable node is reached. The expansion step then adds a new leaf node[1]. A simulation from this leaf node (called a rollout) is then run and the outcome produces a value. This outcome value is then backpropagated up the tree updating the statistics of each node in the path. The visitation count of each node in the path is also incremented. This process is repeated until the computation budget is exhausted. The action $a(\cdot)$ associated with best child of the root node is the preferred next move or strategy choice in the game.

It is common to group these four steps into two policies: a *Tree Policy* that uses the selection and expansion steps to select or create a new leaf node, and a *Default Policy* that performs the rollout. The pseudocode shown in Algorithm 1 shows the MCTS steps. s_0 is the current game state while v_0 is its associated root node in the game tree. *TreePolicy* finds the leaf node v_l and $\triangle$ is the outcome value from the rollout. *Backup* then updates all ancestors nodes in the path. The game tree is recursively expanded until the computational budget is exhausted, which in our work means a fixed number of rollouts were done. A(BestChild) returns the next strategy choice associated with the root node child with the highest mean value[2].

Algorithm 1 MCTS

```
function MonteCarloTreeSearch(s_0)
    create root node v_0 for current game state s_0
    while computation budget not exhausted do
        v_l ← TreePolicy(v_0)
        △ ← DefaultPolicy(s(v_l))
        Backup(v_l, △)
    end while
return a(BestChild(v_0))
end function
```

In our work we evaluated three different MCTS versions. Each version has a slightly different Tree Policy which affects how the game tree is expanded. This will be explained in more detail in Section III.

B. Regret

A slot machine can be considered a one-armed bandit. In a multi-armed bandit (MAB) problem there are $K > 1$ slot machines and machine i produces an expected payoff $\xi_i \in [0, 1]$ each time its arm is pulled. Let $I(n)$ denote the arm some policy P decides to pull in round $n \in \{1, 2, \cdots, T\}$. P also makes a recommendation $J(n)$ on the best arm to pull after all rounds have been played.

P tries to accumulate the highest possible payoff which is equivalent to minimizing the *cumulative regret*

$$CR_n = \sum_{t=1}^{n} (\xi^* - \xi_{I(t)}) \quad \text{where} \quad \xi^* \stackrel{\text{def}}{=} \max_{1 \le i \le K} \xi_i \tag{1}$$

Cumulative regret expresses a disappointment in not necessarily having picked the best machine over a span of n rounds; regret accumulates over time. Conversely, the *simple regret* defined as

$$SR_n = \xi^* - \xi_{J(n)} \tag{2}$$

only reflects regret for not recommending the best arm to pull.

Many MCTS algorithms use the *upper confidence bound for trees* (UCT) policy proposed by Kocsis and Szepesvári [6]. This policy selects child j such that

$$j = \arg\max_i \left(\overline{X}_i + c \sqrt{\frac{\ln(n)}{n_i}} \right) \tag{3}$$

where n represents the number of times the parent node was visited and n_i the number of times child i was visited and the mean value at child i is $\overline{X}_i$ which is assumed to have support [0,1]. c is a constant that balances exploration and exploitation. UCT expands a game tree by trying to minimize cumulative regret.

[1] In this context a "leaf node" represents a nonterminal state that can be expanded during the simulation phase.

[2] Visitation count could also be used to identify the best child, but in our work the child with the best mean value is chosen.

Two policies that bound simple regret are ϵ-greedy and $\text{UCB}_{\surd}$. The ϵ-greedy method [7] samples the arm with the largest sample mean with probability $1-\epsilon$ and with probability ϵ samples a random arm. The $\text{UCB}_{\surd}$ method samples arm j where

$$j = \arg\max_i \left(\overline{X}_i + c \sqrt{\frac{\sqrt{n}}{n_i}} \right) \tag{4}$$

C. Prior work

In our prior work [4] each individual had an offer value p and and acceptance value q. A $(\mu/\mu, \lambda)$ evolution strategy with $\mu = 10$ and $\lambda = 80$ was used to evolve these values. The 10 parents formed a tournament set for evaluating the 80 offspring. Each offspring played one UG round against every member of the tournament set where proposer and responder roles were randomly chosen prior to each interaction. An offer was accepted if $p \geq q$ and rejected otherwise. The fitness of an individual equaled the accumulated payoffs.

In most UG models the acceptance probability is modeled as a Heavyside function—i.e., any $p < q$ is automatically rejected regardless of difference. But this is actually too restrictive. For example, suppose $q = 0.3$. Then an offer of $p = 0.295$ would always be rejected. However, a player might be tempted to at least consider an offer only slightly less than 0.3 if very little payoff was received over the past few rounds. We therefore modeled the acceptance function as sigmoid function to cover such situations. Specifically,

$$\text{prob(accept)} = \begin{cases} 1.0 & p \geq q \\ e^{-\alpha(p-q)^2} & p < q \end{cases} \tag{5}$$

where α is user selected. Figure 1 shows the probability of acceptance for various values of α.

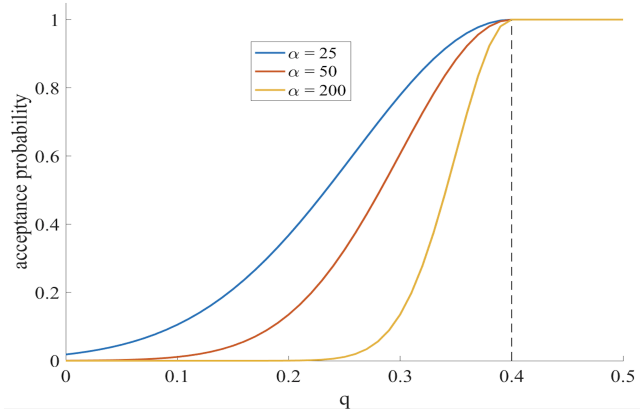

Fig. 1. Sigmoid function showing the probability of acceptance for $q = 0.4$. Notice in the limit, as $\alpha \to \infty$, the sigmoid function becomes a Heavyside function.

Figure 2 shows the population evolution over 150 rounds with $\alpha = 200$ in the sigmoid probability function. Notice that, except for a few outliers, $p > q$ indicating fairness was achieved early and persisted. Moreover, the median values are much larger than the subgame perfect NE.

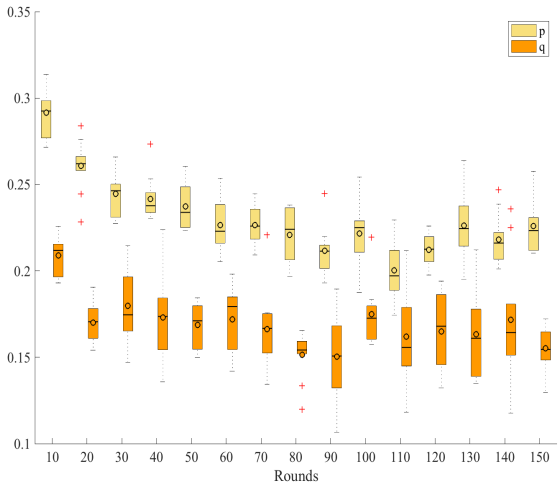

Fig. 2. p and q values averaged over 20 runs of 150 rounds with $\alpha = 200$. The circles (line) inside the boxes represent the mean (median) values.

III. The Model

The model described previously now has one additional player added to the population. This new player is rational meaning his goal is to maximize personal payoff at the expense of the rest of the population which is trying to achieve fairness. This rational player has the same genome and interacts with the same tournament set every round and acquires payoffs just like the rest of the population. What is different is how the p and q values are determined. The $(\mu/\mu, \lambda)$ evolution strategy cannot be used because its purpose is to chose p and q values that promote fairness. Some other method is needed for the rational player.

In this rational player q is permanently set equal to 0.0 so any offer, regardless of how small, will always be accepted. Hence, only some search method for a p value is needed. We use MCTS to conduct the search. The current game state s_0 is the current p value—i.e., $s_0 = p$. Every node in this game tree can have up to three children when fully expanded and the associated actions modify the p value of their parent node. The actions associated with the three children are listed in the table below

child node	action
v_1	$p + \zeta$
v_2	$p - \zeta$
v_3	no change

where ζ is a random variable that changes the parent's p value by 5–10%. A parent node samples a new ζ before expanding any children nodes. However, this sample is taken only once per parent because actions of the immediate children will use the same ζ.

The tree policy controls how the game tree is partially expanded. Any such policy should balance exploration, which

checks seldom visited branches, and exploitation which favors nodes with higher mean values. In this work we investigated three tree policies: the popular UCT and UCT augmented with two sampling methods designed to minimize simple regret. These policies are discussed in greater detail in the next section.

After the tree policy creates a new leaf node the default policy conducts the rollout. A rollout consists of using the p value of the newly expanded node in a tournament against μ randomly selected offspring out of the current population. In this tournament the leaf node always acts as the proposer. A total of 200 rollouts were performed each UG round. Every rollout used a newly sampled tournament set. Let

$$b = \sum_{s=1}^{\mu} \mathbb{1}(p > q_s)$$

be the number of times a proposal was accepted during the rollout. (Eq. (5) determines whether $\mathbb{1}(p > q_s)$ equals 0 or 1.) Then the rollout reward is

$$\triangle = \begin{cases} 0 & \text{if } p < q_s \quad \forall s = 1 \dots \mu \\ 1.0 - \frac{\sum(p - q_s)}{b} - (\mu - b)\gamma & \text{otherwise} \end{cases} \tag{6}$$

where $\gamma = 0.025$. The first term (1.0) is the maximum value while the second term is the average difference between a proposal and an acceptance threshold during the rollout. Only accepted proposals are averaged. The last term is a penalty that decreases the value depending on the number of times a proposal was rejected. The payoff reward $\triangle \in [0, 1]$ is added to the stored value of every node in the path to the root node during backpropagation. The visitation counts of all nodes in this path are also incremented.

After the 200 rollouts are finished, the action associated with the best child (based on highest mean value) determines the new p value for the rational player in the next UG round.

IV. Discussion& Results

In MCTS the root node represents the current game state and its immediate children potential next moves. UCT determines the values of these children by sampling in a manner that reduces cumulative regret. But if the highest value child of the root node is ultimately selected as the best next move, this suggests for games minimizing simple regret at the root node may make more sense. This is not necessarily the case for nodes at lower tiers. Cumulative regret probably provides more precise values for the children of the root node. Thus sampling has different objectives depending on the game tree tier. Tolpin and Shimony [8] recommend a two-stage approach: sample at the root node to minimize simple regret and sample at lower tier nodes to minimize cumulative regret. They conjectured this approach should do better than UCT alone. That is the approach we use. Specifically, ϵ-greedy and $\mathrm{UCB}_{\surd}$ are used to sample at the root node and UCT at other nodes. This is denoted by ϵ-greedy+UCT and $\mathrm{UCB}_{\surd}$ +UCT, respectively. These two versions are compared against a version where UCT sampling was used at all nodes.

Figure 3 shows scatter plots of the payoffs of each MCTS version as well as the average population payoffs. These are payoffs obtained in UG tournaments against the μ parents of the evolution strategy in each round. It is difficult to compare scatter plots so to facilitate comparison a 3rd-order polynomial fit of the scatter plots is shown. Notice all three MCTS versions produced significantly higher payoffs than the population average for the rational player. It is also worth noting the two versions that used simple regret to select children of the root node tended to outperform the UCT-only version.

To further evaluate the MCTS performance we arbitrarily selected one run out of the 20 runs discussed in Section II-C. This run is shown in Figure 4. p and q values for the μ parents and λ offspring in each round were recorded. The expected payoffs in each round for all three MCTS versions were determined by playing an UG against the μ parents. (Recall this is the same method used to compute offspring fitness in the $(\mu/\mu, \lambda)$-ES runs.) The offspring in each round were used during the MCTS rollouts. In the UCT version and the $\mathrm{UCB}_{\surd}$ +UCT version $c = 2$ to balance exploration and exploitation. $\epsilon = 1/2$ was used in the ϵ-greedy version.

Figure 4 indicates a spike in the average population q values at round 60 (about a 40% increase), and another increase of about 50% at round 90. Otherwise the average q values were roughly 0.2 or below. Figure 5 shows how the p values varied in the MCTS versions. All three versions track reasonably well except between rounds 70 and 125. Over these rounds the three versions show radically different behavior: UCT sharply decreased p, ϵ-greedy was roughly constant while $\mathrm{UCB}_{\surd}$ sharply increased p.

To help explain the different MCTS version behaviors between rounds 70 to 125 it is first necessary to look in more detail into how sampling choices in game trees affect regret. The sampling choice depends on whether the objective is to minimize simple regret or cumulative regret. It is worth noting no sampling method can simultaneously optimally lower both simple and cumulative regret [9].

Every node and its immediate children in the game tree is a separate MAB problem; pulling arm i corresponds to selecting child i. Selection favors nodes with higher estimated gains. This estimated gain is the sum of a value associated with the selection and an upper confidence bound of the estimate. UCT and $\mathrm{UCB}_{\surd}$ both use the sample mean of previous rollouts for the value term but use different upper confidence bound terms: $\sqrt{\frac{\ln(n_p)}{n_i}}$ for UCT and $\sqrt{\frac{\sqrt{n_p}}{n_i}}$ for $\mathrm{UCB}_{\surd}$ where n_p is the number of parent node visits and n_i the number of visits to the i-th child node.

Selection has to strike a balance between exploration and exploitation. The upper confidence bound term component in the estimated gain equation promotes exploration. Figure 6 shows how these bounds change for UCT and $\mathrm{UCB}_{\surd}$. Both exhibit a logarithmic decrease as nodes are more frequently

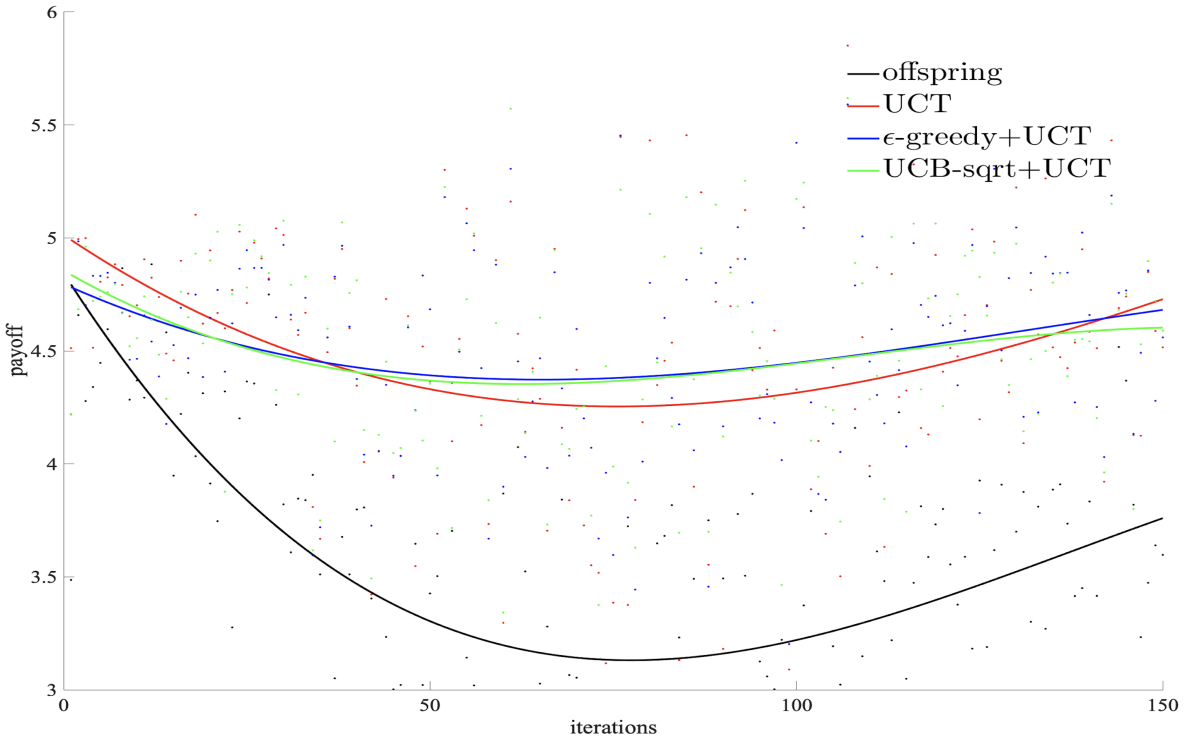

Fig. 3. Scatter plots showing average population payoffs and rational player payoffs using three different MCTS methods. Lines show a 3rd-order polynomial fit of the scatter plot points to help compare results.

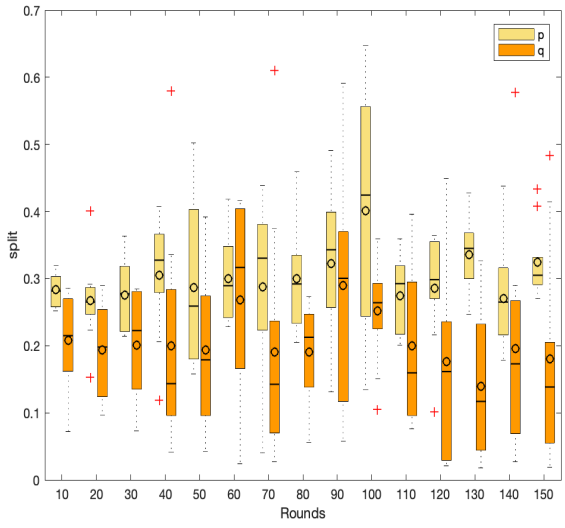

Fig. 4. Single evolution strategy run (out of 20) used to compare the three MCTS strategies for the rational player.

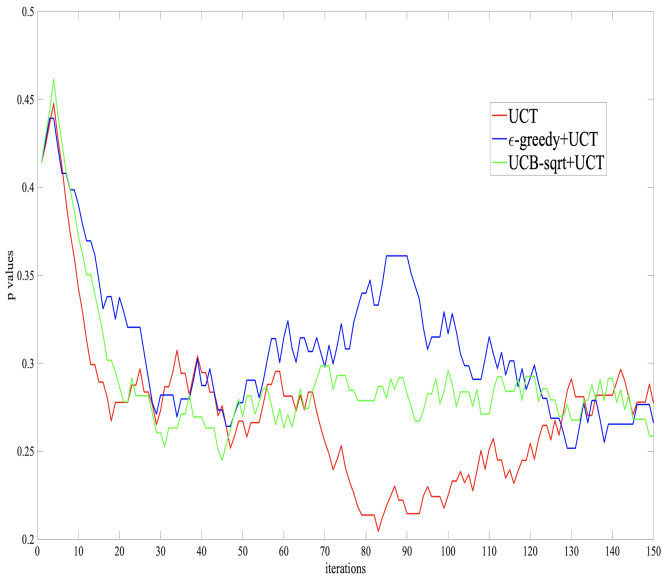

Fig. 5. Payoff differences for three MCTS strategies used in the Figure 4 run.

visited but the $\text{UCB}_{\surd}$ decreases slower. This means, for a given number of visits, the $\text{UCB}_{\surd}$ upper confidence bound is larger thereby contributing more to the estimated gain—i.e., it emphasizes more exploration than UCT.

ϵ-greedy sampling does not have an exploration term. Instead it samples the highest sample mean node with probability $1-\epsilon$ and a random node (including the highest sample mean node) with probability ϵ. Random node selection ignores the sample means. Thus, with ℓ children nodes the selection probabilities are

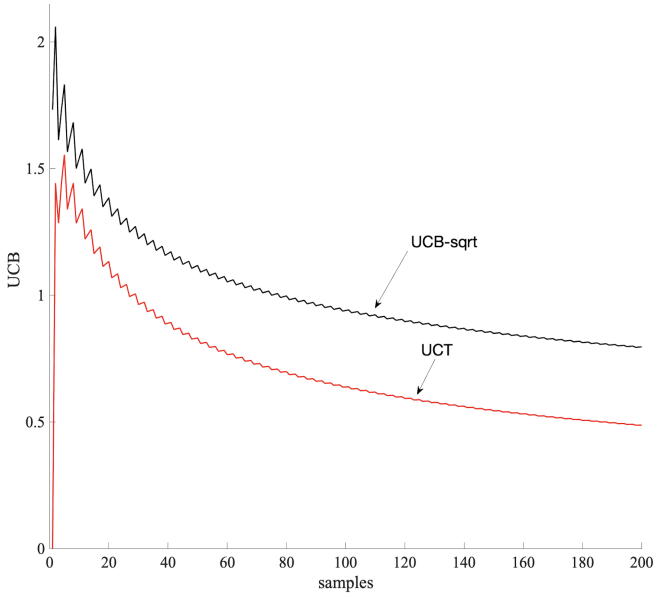

Fig. 6. shows sqrt vs uct

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

Notice sub-optimal nodes are selected with some non-zero probability regardless of their sample mean. It therefore also promotes more exploration than UCT.

UCT minimizes cumulative regret so it tends to sample the current highest sample mean node more frequently, which limits exploring less visited branches. Conversely, a sampling method that explores more and tends to choose the current highest sample mean node less often works better at minimizing simple regret. ϵ-greedy and $\mathrm{UCB}_{\surd}$ both put higher emphasis on exploration than UCT does. Consequently, while UCT minimizes cumulative regret, ϵ-greedy and $\mathrm{UCB}_{\surd}$ are better suited for minimizing simple regret. More specifically, it has been shown, for some constant $\varphi \geq 0$, after n visits the expected simple regret $\mathbb{E}r_n$ is bounded from above by

$$\mathbb{E}r_n \leq O\big(e^{-\varphi n}\big)$$

for $\frac{1}{2}$-greedy [8],

$$\mathbb{E}r_n \leq O\big(e^{-\varphi\sqrt{n}}\big)$$

for $\mathrm{UCB}_{\surd}$ [8], and

$$\mathbb{E}r_n \leq O\big(n^{-\varphi}\big)$$

for UCT [10]. Thus, with UCT simple regret only decreases at best polynomially with n whereas it decreases exponentially with n in the other two sampling methods.

This leads to a possible explanation for why the three MCTS methods exhibited different behavior between rounds 70 and 125. First consider the UCT behavior. Referring to Figure 4, there is an initial decrease in average q values between rounds 60 and 70 followed by a sharp increase at round 90. The q values then gradually return to a more moderate level for the remainder of the run where fairness begins to appear. Intuitively a rational player's p value should follow that same profile: decrease p initially to exploit the lower q average, then increase to compensate for the sudden increased q average and finally decrease again as q decreases. UCT sampling makes the initial decrease in p but recovers sluggishly. We conjecture this is due to UCT choosing the highest sample mean node too often which is more likely when minimizing cumulative regret. Limiting exploration makes it difficult to respond to transient q changes. Conversely, ϵ-greedy+UCT and $\mathrm{UCB}_{\surd}$+UCT sampling minimize simple regret. They explore more than UCT and are thus better able to respond to average q fluctuations. ϵ-greedy+UCT sampling responded similar to the way a rational player would: as the q averages peaked near round 90 and then fell, so did the p value. By increasing p to compensate for the increased q average it is more likely offers will be accepted thereby increasing the player's payoffs. The better ϵ-greedy+UCT response is expected given that it has the greatest ability to explore among the three MCTS versions. It has the smallest upper bound on $\mathbb{E}r_n$ and visits sub-optimal nodes in this particular problem with probability 0.33. $\mathrm{UCB}_{\surd}$+UCT sampling seems to equally balance exploration and exploration which makes it relatively immune to transient q changes.

It is perhaps worth noting that the implementation of MCTS in this paper has a novel character. Most MCTS is used to generate moves for a two player game of substantial length like checkers, chess, or go. The ultimatum game is a single-shot game. In a game like chess, the rollout consists of playing the game until a win, lose, or draw result is obtained. In this study, this is replaced by competition against a tournament set. Both sorts of rollout provide the information needed to continue expanding the MCTS partial game tree, but the use of a rollout that samples against a tournament gives this information a different character that make MCTS a natural choice for the play of mathematical games.

V. Conclusions & Future Work

We have shown that MCTS can construct a rational player that can exploit a population of players trying to achieve fairness in an UG. Payoffs received were significantly larger than the population average (nearly 40% larger). Sampling which minimizes simple regret at the root node and UCT at the other nodes performed better than using the UCT policy at all nodes. ϵ-greedy+UCT appears to respond the best to rapidly changing responder q values.

To the best of our knowledge this is the first instance where MCTS is used for making strategy decisions in an economic game. The novel rollout approach used in this work opens up the possibility of using MCTS to adapt strategies in mathematical games in general and social dilemma games in particular.

In the public goods game, players contribute from their own resources an investment in the public purse, a common pool which is then multiplied by a factor greater than one and divided evenly among the players. It is a substantially different game from the ultimatum game, but also possesses a similar conflict between its Nash equilibria and what human players might perceive as fair. The highest aggregate payoff arises from each player contributing all their resources. Any player who retains some of their own resources obtains a total of their retained wealth plus the common payout. If fairness represents equal contribution then rational play tends directly away from fairness. This, in turn, means that the Nash equilibria is for each player to retain all of their own resources, at which point fairness return, but at a very low level of public investment. In this game, a rational MCTS agent would rapidly discover the strategy of hoarding its wealth. If a penalty for insufficient contribution to the public purse is added the the game, as a rule or by permitting punishment, the MCTS agent would be useful for discovering this statutory or social boundary for acceptable contributions.

For mathematical games, like the IPD, an MCTS agent would be useful for rapidly discovering an optimally exploitative counter strategy, using the simple move set "cooperate" and "defect". The PD strategy tit-for-two-tats, for example, defects if its opponent has defected on the last two rounds. An optimal counter strategy is to defect every other round. This would make a nice initial test of an MCTS agent for prisoner's dilemma.

A more complex situation arises in a multi-player version of PD in which a group of agents make a move and then receive the score against each of their opponents, based on the pairwise moves. Finding strategies to play against various mixtures of opponents was examined in [11], using an evolutionary algorithm to train a finite state agent against the group of opponents. This situation, maximizing score of an agent against a fixed collection of opponents, creates an optimization problem from IPD. AN MCTS agent seems likely to substantially outperform an evolved agent for speed in locating high-scoring strategies against such groups of agents. This notion is framed for IPD, but could be used to train a rational agent in any simultaneous N-player game.

If a novel mathematical game were under investigation, then pitting pairs of MCTS agents, with different tree policies or stochastic default policies, against one another would be a good way to rapidly estimate the richness of the strategy space of the game. If the sequences of moves generate by the MCTS agents formed a relatively restricted group, the games have a relatively simple strategy.

An interesting variant of using MCTS would be to explore strategies for mathematical games with stochastic or non-stationary payoffs. The ability of MCTS to continue exploration would permit it to adapt rapidly to games where the payoff matrix changed over the course of the game. This ability requires that the MCTS agent be given advanced knowledge of the changes of the scoring system over time, in order to perform its rollouts. If no such forward model of the change of the scores over time existed, then a strategy that maintains a diverse evolving population would probably be superior.

REFERENCES

[1] P. Van Lange, J. Joireman, C. Parks, and E. Van Dijk. The psychology of social dilemmas: a review. *Org. Behav. and Human Decis. Proc.*, 120:125–141, 2013.

[2] W. Güth, R. Schmittberger, and B. Schwarze. An experimental analysis of ultimatum bargaining. *J. Econ. Behav. Organ.*, 3(4):367–388, 1982.

[3] C. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton Univ. Press, 2003.

[4] G. Greenwood and D. Ashlock. On the evolution of fairness in N-player ultimatum games. In *Proc. 2018 IEEE Cong. on Evol. Comput.*, pages 17–22, Rio de Janeiro, Brazil, 2018.

[5] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, S. Samothrakis, and S. colton. A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intell. and AI in Games*, 4(1):1–42, 2012.

[6] L. Kocsis and C. Szepesvári. Bandit based Monte Carlo planning. In *Proc. Eur. Conf. Mach. Learn.*, pages 282–293, Berlin, Germany, 2006.

[7] R. Sutton and A. Barto. *Reinforcement learning, an introduction*. Cambridge: MIT Press/Bradford Books, 1998.

[8] D. Tolpin and S. Shimony. MCTS based on simple regret. In *26th AAAI Conf. on Artif. Intell.*, pages 570–576, Toronto, ON, Canada, 2012.

[9] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in finitely-armed and continuously-armed bandits. *Theo. Comp. Sci.*, 412:1832–1852, 2011.

[10] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandit problems. In *Proc. 20th Int'l. Conf. on Alg. Learning Theo.*, LNAI 5809, pages 23–37, 2009.

[11] D. Ashlock, J. Brown, and P. Hingston. Multiple opponent optimization of prisoner's dilemma playing agents. *IEEE Trans. Comput. Intell. and AI in Games*, 7(1):53–65, 2015.