ASYNCHRONOUS COALITION IN MULTI-PLAYER GAMES, WITH APPLICATIONS TO GUTS POKER

LEAH DICHTER AND DAVID JIANG

Abstract. Cooperation between players introduces an understudied dimension to multi-player game theory, and the effects of asynchronous coalitions are particularly unknown. To understand Guts poker and build on the study of synchronous coalition strategies for recursive games introduced by Buck, Lee, Platnick, Wheeler and Zumbrun, we carry out an analysis of 1 vs. n player asynchronous coalitions. With the goal of developing a practical numerical algorithm to approximate optimal asynchronous coalition strategies for Guts, and for the generalized analysis of multiplayer games, we discuss and implement diverse optimization and sorting algorithms for comparison to perform analysis on 3-player Rock-Paper-Scissors. Applying our chosen algorithm to 3 player Guts, we find that the optimal asynchronous coalition play are pure strategies of $1/\sqrt{2}$, resulting in a payoff of zero to player 1. Significantly, we find that player 1 may, at best, force a payoff of approximately -0.0058 against the coalition, revealing an inequality between the minimax and maximin and that Guts does not have a value.

1. Introduction

In this paper, we study asynchronous coalitions in multiplayer games with target applications to the Guts poker problem introduced in [CCZ, BLPWZ] and with the aim of producing a practical and generalized algorithm to optimize multiplayer strategy. We begin with an analysis of 3-player adaptations on the classic game Rock-Paper-Scissors to develop our methodology and intuition.

1.1. Classical Game Theory Preliminaries.

First, we introduce basic definitions and calculations for classical game theory.

Definition 1.1. A *Zero-sum game* is any game in which the gains of one player are equivalent to the losses of another, so in which the payoffs to all players sum to zero.

Zero-sum is a useful game property for our analysis, because we may evaluate the expected outcomes of a 2-player game purely in terms of the payoff to one player. Rock-Paper-Scissors is a classic example of a zero-sum game.

Definition 1.2. The set of *pure strategies* denotes the set of distinct, strategic actions available to players within a singular round of game play.

Definition 1.3. A *mixed strategy* is a probability distribution over the set of pure strategies, representing the frequency of their use by a particular player over an extended period of game play.

We can represent players, and their mixed strategies, as vectors in the form V = (v_1, v_2 ,..., v_n), where v_i represents the probability that a given player will choose the i-th strategy from the set of pure strategies enumerated 1 to n on any given round, and $\Sigma_{i=1}^{n} v_i = 1$.

For example, enumerating the strategies in classic Rock-Paper-Scissors as {Rock, Paper, Scissors} $\to$ {1, 2, 3}, a player of the game choosing to throw either Rock or Paper with equal probability will be depicted by the vector:

$$p_2 = (1/2, 1/2, 0)$$

On the other hand, a player choosing to use a pure strategy of scissors with 100% probability shall be notated as follows:

$$p_1 = (0, 0, 1)$$

Definition 1.4. A $n \times m$ *payoff matrix* contains the possible outcomes of a 2-player game in which player 1 may choose from n different pure strategies and player 2 may choose from m different pure strategies. For zero-sum games, we may write the outcome in terms of the payoff to player 1.

Valuing a win as a payoff of 1 unit and a loss as -1, the payoff matrix for 2-player Rock-Paper-Scissors is shown below, where the strategy options for player 1 are represented by each row:

$$A = \begin{pmatrix} & R & P & S \\ R & 0 & -1 & 1 \\ P & 1 & 0 & -1 \\ S & -1 & 1 & 0 \end{pmatrix}$$

We can also use payoff matrices to calculate the expected payoff of a game, given any two player strategies. Given a payoff matrix A and mixed strategies (x, y), we will have the following function ψ for the expected payoff of one round:

$$\psi(x, y) = xAy^\top = \sum_{i,j} a_{i,j} x_i y_j$$

We can evaluate this function using the payoff matrix for Rock-Paper-Scissors (RPS) and our example player strategies (p_1, p_2), described earlier:

$$\psi(p_1, p_2) = \begin{pmatrix} 1/2 & 1/2 & 0 \end{pmatrix} \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = 0$$

With this calculation, we observe that the expected payoff for player 1 is zero, but we may also derive this result intuitively. Knowing that player 2 will throw Scissors with 100% certainty, player 1 is guaranteed to win playing Rock and to lose playing Paper. Accordingly, player 1 will win or lose with equal probability, averaging to an expected payoff of zero. The simplicity of Rock-Paper-Scissors lends its self to our early research; as we transition to developing numerical algorithms for optimization, we can compare computationally derived results with our intuitive understanding of the game.

1.2. **Fundamental Theorem of Game Theory.** In this section, we introduce John von Neumann's Fundamental Theorem of Game Theory, also known as the Minimax Theorem, as it pertains to our research.

Definition 1.5. A *Nash Equilibrium* is a game state represented by a set of strategies for all players, in which all players are incentivized to maintain their chosen strategy, since no player may improve their payoff through an individual strategy deviation.

We will define and differentiate between the minimax and maximin as two different approaches to strategy for zero-sum, $n \geq 2$-player games.

Definition 1.6. The *maximin* is a strategy that yields the maximum payoff for one player that can be guaranteed. For $n \geq 2$-player games, the *maximin* is determined by assuming opponents will select a worst-case response and identifying the strategy choice with the highest, minimum payoff. Accordingly, the value of the *maximin* is the maximum payoff that a player can force.

Definition 1.7. The *minimax* is a strategy that minimizes payoff to a player's opponents. For $n \geq 2$-player games, the *minimax* is determined by identifying a strategy that forces the lowest possible payoff to opponents, assuming opponents will select a response to maximize their payoff.

Given mixed strategies (x, y) and a function $\psi(x, y)$ for expected payoff, the minimax and maximin are written as follows:

$$\min_y \max_x \psi(x, y)$$
$$\max_x \min_y \psi(x, y)$$

Definition 1.8. The *value* of a game, denoted V, is a property of any game in which the value of the minimax and maximin are equivalent, and it is equal to their same value.

Theorem 1.1. For any finite, zero-sum, 2-player game, let x and y be mixed strategies and $\psi(x, y)$ be a continuous, bilinear function for expected payoff. Then,

$$\max_x \min_y \psi(x, y) = \min_y \max_x \psi(x, y) = V$$

Upon proving his theorem in 1928, von Neumann famously stated, "As far as I can see, there could be no theory of games, without that theorem; I thought there was nothing worth publishing until the Minimax Theorem was proved" [vN]. Indeed, von Neumann's theorem proves the existence of optimal mixed strategies, the solutions x and y to the minimax problem, for any two-player, finite, zero-sum game. Accordingly, any strategy profile that is a solution to the minimax and maximin is a Nash Equilibrium.

As we will explore, the Minimax Theorem doesn't necessarily apply to multiplayer ($n > 2$) games, and the possibility of coalition adds further complexity to the problem. Particularly, it is likely that the coalition will have a stronger ability to force gameplay, thus breaking the minimax-maximin equality proved by Neumann for $n = 2$. We will evaluate and compare the minimax and maximin values for multiplayer games with coalitions, with a focus on "Guts" poker, to determine the existence of a game value and search for optimal strategies.

2. Multiplayer Games

In this section, we will apply classical game theory to multiplayer games, beginning with 3-player Rock-Paper-Scissors as an example to build intuition.

2.1. 3-Player Rock-Paper-Scissors.

First, we will outline the rules of two different multi-player adaptations on classic Rock-Paper-Scissors.

Odd-Man-In (OMI) and Odd-Man-Out (OMO) are both 3-player iterations of Rock-Paper-Scissors, which disregard the notions of hierarchy between pure strategies in the classic 2-player game. For both games, each player will select and play a pure strategy (either Rock,

Paper or Scissors) in unison. In OMI, if players select three distinct or equal strategies, there is a draw; otherwise, the player who selects a different strategy to both their opponents wins. Conversely, in OMO, the player who selects the unique strategy loses.

Just as with classic Rock-Paper-Scissors, we may code the strategies and payoffs for OMI and OMO into a payoff matrix. However, with three players as variables, a 2D matrix will no longer suffice and we will use a 3D tensor, instead. The payoff tensor for OMO is shown bellow:

$$\begin{pmatrix} \mathbf{R} & R & P & S \\ R & 0 & 1 & 1 \\ P & 1 & -2 & 0 \\ S & 1 & 0 & -2 \end{pmatrix} \quad \begin{pmatrix} \mathbf{P} & R & P & S \\ R & 0 & 1 & 1 \\ P & 1 & -2 & 0 \\ S & 1 & 0 & -2 \end{pmatrix} \quad \begin{pmatrix} \mathbf{S} & R & P & S \\ R & 0 & 1 & 1 \\ P & 1 & -2 & 0 \\ S & 1 & 0 & -2 \end{pmatrix}$$

Our 3-player adaptations of Rock-Paper-Scissors remain zero-sum-games, so we write the payoff tensor in terms of payoff to player 1. The strategy choices for player 1 are indicated in top, left-hand corners of each 2D matrix in bold.

Given three mixed strategies, (p_1, p_2, p_3) and a payoff tensor A, we may also calculate the expected payoff of a 3-player game with the following function ψ:

$$\psi(p_1, p_2, p_3) = p_1 \left(\sum_i^3 p_2 A_i p_3^\top \right)$$

To facilitate optimization, we derive a closed-form function for the expected payoff of player 1 in OMO as follows:

Let p_1, p_2, and p_3 represent mixed strategies for three players, where x_i, y_i, and z_i are the frequencies that respective players will choose the i-th pure strategy $\in \{Rock, Paper, Scissors\}$ on any given round.

$$\begin{aligned} p_1 &= (x_1, x_2, x_3) \\ p_2 &= (y_1, y_2, y_3) \\ p_3 &= (z_1, z_2, z_3) \end{aligned}$$

Inputting our mixed strategies (p_1, p_2, p_3) into the payoff function for 3-player games, using the payoff tensor for OMO, we can expand to get the following:

$$\begin{aligned} \psi(p_1, p_2, p_3) = {} & x_1(y_1z_2 + y_1z_3 + y_2z_1 - 2y_2z_2 + y_3z_1 - 2y_3z_3) \\ & + x_2(-2y_1z_1 + y_1z_2 + y_2z_1 + y_2z_3 + y_3z_2 - 2y_3z_3) \\ & + x_3(-2y_1z_1 + y_1z_3 - 2y_2z_2 + y_2z_3 + y_3z_1 + y_3z_2) \end{aligned}$$

Pulling out $-2y_iz_i$ from each term x_i, we may rewrite the expected payoff:

$$\begin{aligned} \psi(p_1, p_2, p_3) = {} & x_1(y_1z_2 + y_1z_3 + y_2z_1 + y_3z_1 + 2y_1z_1) \\ & + x_2(y_1z_2 + y_2z_1 + y_2z_3 + y_3z_2 + 2y_2z_2) \\ & + x_3(y_1z_3 + y_2z_3 + y_3z_1 + y_3z_2 + 2y_3z_3) \\ & - 2y \cdot z(x_1 + x_2 + x_3) \end{aligned}$$

We know that $x_1 + x_2 + x_3 = 1$, so factoring out y_i and z_i from each term, the expected payoff can be further simplified:

$$\begin{aligned}\psi(p_1, p_2, p_3) &= x_1(y_1 + z_1) + x_2(y_2 + z_2) + x_3(y_3 + z_3) - 2y \cdot z \\ &= \sum_{i=1} x_i(y_i + z_i) - 2y \cdot z\end{aligned}$$

We will optimize with the above closed-form function to evaluate the minimax and maximin for Odd-Man-Out. Similarly, we will use a derived closed-form function to compute the expected payoff for Guts and to optimize with over mixed strategies for Guts.

2.2. **Introduction to Coalition Strategies.** In $n \geq 3$ player games, players may form coalitions against other players. We differentiate between two different types of coalition strategies.

Definition 2.1. In a *synchronous coalition*, players may collude to select their strategies together throughout game play and may coordinate strategies to respond to real-time updates in the game.

Definition 2.2. In an *asynchronous coalitions*, players are limited to selecting mixed strategies in advance of game play, and they may not communicate to coordinate their strategies during the game.

Consider a coalition between two players, A and B, for a general game with pure, individual strategies enumerated $1 - n$.

We can define the possible pure, synchronous coalition strategies as the set:

$$S = \{(a_1, b_1), (a_1, b_2), ..., (a_n, b_n)\}$$

where $(a_i, b_j) \in S$ is the coalition strategy in which player A uses the i-th pure, individual strategy and player B uses the j-th pure, individual strategy. We will enumerate elements of S in order from $1 - n^2$. Moreover, we may represent a mixed, synchronous coalition strategy as a vector in the form:

$$V = (v_1, v_2, ..., v_{n^2})$$

where v_k is the probability of the coalition using the k-th strategy $\in S$, and $\Sigma_{k=1}^{n^2} v_k = 1$.

On the other hand, a mixed, asynchronous coalition strategy may be represented by a vector in the form:

$$V = (v_{A1}, v_{A2}, ..., v_{An}, v_{B1}, ..., v_{Bn})$$

where $v_A i$ is the probability of player A using the i-th pure, individual strategy, and $v_B j$ is the probability of player B using the j-th pure, individual strategy. Additionally, $\Sigma_{i=1}^{n} v_A i = 1$ and $\Sigma_{j=1}^{n} v_B j = 1$. For the efficiency of our optimization, we will rewrite mixed strategy vectors in the form:

$$V = (v_{A1}, v_{A2}, ..., v_{An-1}, v_{B1}, ..., v_{Bn-1})$$

where $1 - \Sigma_{i=1}^{n} v_A i = v_A n$ and $1 - \Sigma_{j=1}^{n} v_B j = v_B n$, and V has size $2n - 2$. We will optimize over vectors in this later form to search for mixed, asynchronous coalition strategies for Guts and Rock-Paper-Scissors.

2.3. **Minimax Problem for Mulitiplayer Games.** With the goal of evaluating von Neumann's minimax equality [see 1.2] for multiplayer games and solving for optimal asynchronous coalition strategies for Guts and Rock-Paper-Scissors, we rewrite the minimax and maximin for 3-player games with a coalition, given any 3 players (x, y, z) and a payoff function ψ:

$$\max_{x} \min_{y,z}(\psi(x,y,z)) \stackrel{?}{=} \min_{y,z} \max_{x}(\psi(x,y,z))$$

3. Optimization Methods

In the following sections, we discuss and compare mixed optimization methods for computing the minimax and maximin. We compute results for a 3-player iteration of Rock-Paper-Scissors called Odd-Man-Out [2.1] to evaluate accuracy and time complexity.

3.1. **Computing the Minimax.** We first develop an algorithm to compute the minimax. Recall that our payoff functions compute payoff for player 1, and we will refer to "payoff" in terms of player 1 from henceforth. We write the minimax for Rock-Paper-Scissors using our derived payoff function [2.1].

$$\min_{y,z} \max_{x} \left(\sum_{i=1} x_i(y_i + z_i) - 2y \cdot z \right)$$

We can distinguish between two optimization routines required to solve the minimax: the inner routine, $\max_x$, and the outer routine, $\min_{y,z}$. For the former, we will maximize payoff with respect to player 1's mixed strategy x, given any asynchronous coalition strategy vector containing y, z [see 2.2]. Since our payoff function for Rock-Paper-Scissors is linear with respect to x, we can use linear programming methods to compute the $\max_x$.

We use HiGHS, a simplex based linear programming method developed by Huangfu and Hall [HH], which is the default method in SciPy's linear programming module. For Odd-Man-Out, we find that player 1 should choose an entirely pure strategy to maximize payoff, such that:

$$x_{\max_i(y_i+z_i)} = 1 \tag{1}$$

In other words, player 1 should select the pure strategy that has the highest joint probability of being played by players 2 and 3. Intuitively, we expect this result; for OMO, players want to avoid throwing a unique strategy, which would result in negative payoff. For the purpose of improving time complexity while testing our $\min_{y,z}$ routine, we may substitute HiGHS to simply find the strategy index, $\max_i(y_i + z_i)$. However, it is important to note that this heuristic is distinct to OMO and cannot be generalized to other games.

3.2. **Computing the Maximin.** Now that we have a method to evaluate the inner function, we need to find a way to optimize over the outer function. The inner max function is not linear so it becomes a bit more of a challenge to optimize over. We will first apply **Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)** to solve for min over y, z. BFGS is a quasi-Newton method that allows us to do non-linear optimization which works out nicely for our problem.

We used SciPy's optimization method "minimize". This requires us to give it an objective function, a starting value, and constraints for the problem, in the form of inequalities.

$$\text{Objective function} = \max_{x} \psi(x,y,z)$$

$$\text{Starting value} = \text{Randomly generated}$$

$$\text{Constraints} = \begin{cases} y_1 + y_2 \leq 1 \\ z_1 + z_2 \leq 1 \\ y_1, y_2, z_1, z_2 \geq 0 \end{cases} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \leq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} yz \leq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Our objective function will take in our combined yz vector and find the best strategy for x which we showed previously is just the most probable yz throw. Then we used that strategy in our payoff function.

The starting value for yz are randomly generated used np.random_sample. The use of random starting values is beneficial because BFGS is not a global optimization method and commonly falls into the trap of local minima.

The constraints correspond to our yz vector, the first four rows correspond to the fact that $0 \leq y_1, y_2, z_1, z_2 \leq 1$ while the last two rows correspond to the fact that $0 \leq y_1 + y_2 \leq 1$ and $0 \leq z_1 + z_2 \leq 1$.

We will now run this optimization thousands of times and take the smallest payoff (min) and the corresponding yz vector for that payoff. The output that we end up getting is that yz play a pure strategy together so they will never lose, but then x has the ability to play there as well to guarantee that it will always end in a tie. So means that the minimax for OMO is 0 or:

$$\min_{y,z} \max_{x} \psi(x, y, z) = 0$$

This was also confirmed by discretizing yz with a very low mesh size. It is important to note that there are other strategies that yield 0 for the minimax. It is not necessarily unique.

3.2.1. *Maximin.*

$$\max_{x} \min_{y,z} \left(\sum_{i=1} x_i(y_i + z_i) - 2y \cdot z \right)$$

Similarly, we can think of this expression as players y, z being able to see player x's strategy first.

This function is no longer linear and we can't take advantage of the structure of the problem as easily. The approach that we took was continuing to use BFGS for the inner function and discretizing player x's strategy. Which essentially means that we will be trying a large number of different values for x and finding the one that maximized the payoff.

Upon running BFGS, we found values returned for x that we could easily show were not the global minima. As an example we got:

These values that we got are local minima, which is a problem because we are looking to globally minimize over y, z. This is one of the perils of using BFGS, since it is a local minimization method, it is prone to finding local minima.

Instead we want to find global optimization methods. The one we ended up settling with is **Basin-Hopping** [WD]. It is inspired by Monte-Carlo minimization and simulated annealing. For our sakes, it is essentially a black box. But what it aims to accomplish is trying a bunch of values for yz in a manner that doesn't require trying all values. Basin-Hopping does solve our issue and finds our global minima for given yz. The issue with this is that Basin-Hopping is a relatively slow method and won't be computationally/time feasible for a larger game

such as Guts poker. But, again, taking advantage of the fact that OMO is a small game, we can find the value of the maximin.

We find that the maximin is equivalent to $-4/3$. The strategy that corresponds to this is $x = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ and then y and z can just play a pure strategy together at any of the throws.

3.2.2. *Comparison.* After finding the answer for both sides, we know know that:

$$\max_x \min_{y,z}(\psi(x,y,z)) = -\frac{4}{3} < 0 = \min_{y,z} \max_x(\psi(x,y,z))$$

So this means that the value of our 3 player version of rock paper scissors does not exist beacuse the two sides of the minimax do not agree with each other. In particular, we know that in the maximin, player x has a negative expected value while in the minimax, player x has a 0 expected value.

4. Guts Poker

4.1. **Rules.** (For the full rules, check out the Wizard of Odds website.)

Guts poker is an old variant of poker where you can play with any number of players if the amount of cards dealt permits (26 players for 2 card Guts). You are able to play with multiple cards and create your own rule set. For sake of simplicity, we will look at 2 card Guts.

At the start of the game, every player will ante a set amount of chips to the pot. Then everyone will be dealt their 2 cards. Then each player will need to decide whether they want to hold or fold their hand. The dealer will count down and then everyone will reveal their decision.

Those who folded are automatically eliminated from winning the pot. In the case that everyone folded, the antes will reset back to the original ante from the start of the game. If anyone had held, then those who folded will not have to ante anything for the next round.

Those who held now have to compare their hands against each other. In the case for 2 card Guts, a pair beats a non-pair, a higher value pair beats a lower value pair, and a higher value high card in non-pair hands beat lower value high cards in non-pair hands.

The player who had the strongest hand wins the pot, while those who held their cards but did not win will have to match whatever the pot was. This allows for the pot to grow rapidly.

4.2. **Continuous Guts.** Following from the work of the previous years work of [CCZ, BLPWZ] we will map the strength of a hand to the continuous number line of $[0, 1]$. When playing 2 card Guts, this means that the best hand A, A would be a 1 while the worst hand, $2, 3$ would be a 0. The rest of the hands are placed in between these values.

4.3. **Strategy Vectors in Guts.**

Definition 4.1. *Pure Strategies in Guts* are a singular float from $[0, 1]$ in which the player holds if their hand exceeds that value.

Definition 4.2. *Mixed Strategies in Guts* is a vector that correspond to how often you play certain pure strategies.

An example of a pure strategy would be 0.5 in which the player would play any hand better than a 0.5 hand strength, or put simpler, they play a hand better than average.

An example of a mixed strategy would be $\left(0, \frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right)$ where there is a 1/2 chance that they play a pure strategy of 0.4, 1/3 chance that they play a pure strategy of 0.6, and a 1/6 chance that they play a pure strategy of 0.8.

4.4. **Guts Payoff Function.** The payoff function for Guts is a bit more difficult to calculate than it is for RPS. Since in Guts the strategy you play is on the continuous number line, that implies there are an infinite amount of strategies in which you can play. This makes impossible to work with the payoff matrix for the game. Instead, we use the work from [CCZ] in which they derived a formula for the one-shot payoff in Guts.

$$\alpha(x,y,z) = \begin{cases} 2x - y - z + z^3 + 3y^2z - 4xyz, & x < y < z \\ 2x - y - z + y^3 + 3z^2y - 4xyz, & x < z < y \\ 2x - y - z + z^3 - 3x^2z + 2xyz, & y < x < z \\ 2x - y - z + y^3 - 3x^3y + 2xyz, & z < x < y \\ 2x - y - z - 2x^3 + 2xyz, & y < z < x \\ 2x - y - z - 2x^3 + 2xyz, & z < y < x \end{cases}$$

This function is in terms of player x and takes in pure strategies for all of the players and returns the expected payoff for playing x.

4.5. **Optimizing Guts.**

4.5.1. *Minimax.* Recollect that the minimax function for a three player game looks like:

$$\min_{y,z} \max_{x} \psi(x,y,z)$$

So we want a quick way to find the best strategy for x given any mixed strategy y, z. We can do this by using the quicksort method we discussed previously. We can create a discretized vector for x and try that against all the possible values that y and z can play given their mixed strategies. This will give us all the possible payoffs that x can get. We then want to sort these values, the largest one of these is the one that x wants since it is maximizing over x.

On our outer loop, we then use BFGS to determine the best strategy for y and z, given that x is using the best response strategy, as calculated using quicksort, allowing y and z to take on mixed strategies. We find that the best coalition strategy for y and z are infact pure strategies of $\frac{1}{\sqrt{2}}$, which forces a minimized payoff of zero for player 1. Significantly, this result reveals that player 2 and 3 cannot force a loss for player 1 with an asychronous coalition.

4.5.2. *Maximin.* Now we are looking at the function:

$$\max_{x} \max_{y,z} \psi(x,y,z)$$

We will also use BFGS along with sorting for this function as well. We again generate a random initial guess for our BFGS.

In each iteration we will call on our max_x function to get our payout for each iteration of strategy x.

```
def max_x(p1_mixed):
    p1_list = p1_mixed.tolist()
    p1_list.append(1-sum(p1_list))

    global global_p1
    global_p1 = p1_list

    #Get best response strategies for player 2 and 3
    best_pair = min_yz(global_p1)
    return check_payoff(best_pair)
```

$p1\_mixed$ is the mixed strategy that we are currently iterating through in BFGS. Then we add the final term onto the mixed strategy, since by the structure of the problem we omit the final probability since our vectors have $n-1$ degrees of freedom. Then we will call on our inner function min_yz to determine the best strategies for y and z.

```
def min_yz(p1_mixed):
    strats = generate_strategy_pairs()
    d = dict()
    for strat in strats:
        d[strat] = check_payoff(strat)

    sorted_strats = sorted(d.items(), key=lambda x:x[1])
    best_pair = sorted_strats[-1][0]

    return best_pair
```

This method first generates all possible pairs of strategies that y and z can play with the given discretization originally. We then want to create a dictionary where the key is the strategy pair and the value is the payoff. We fill this dictionary as we iterate through the strategy pairs. Once the dictionary is full, we can sort the dictionary to find the best response such that we minimize the payoff/dictionary value. This will give us our inner function.

Once we have the best strategy pair that y and z can play, we just need to return the payoff given the pair as well as the current iteration of the strategy x is using. This output is our function that we are optimizing over for BFGS.

Running on high mesh sizes for our discretization we notice that the payoff is not 0. We find small payoff values around -0.007. Although this value might seem small, the important thing to realize is that it is non-zero, albeit very close.

```
Maximin processed in 23.1110 seconds

Best P1 mixed strategy...
Strategy: 0.52; Probability: 0.0287
Strategy: 0.54; Probability: 0.0194
Strategy: 0.56; Probability: 0.033
Strategy: 0.58; Probability: 0.0484
Strategy: 0.6; Probability: 0.1108
Strategy: 0.62; Probability: 0.2107
Strategy: 0.64; Probability: 0.1068
Strategy: 0.66; Probability: 0.157
Strategy: 0.68; Probability: 0.1106
Strategy: 0.7; Probability: 0.0773
Strategy: 0.72; Probability: 0.0958

Best (P2, P3) pure strategy pair: (0.68, 0.66)

Resulting payoff to P1: -0.0073
```

4.5.3. *Comparison.* Our results from Guts match up with our results from OMO. We found that the maximin is less than the minimax. The effects in this case are a lot less drastic than what we found for OMO.

$$\max_x \min_{y,z}(\psi(x,y,z)) \approx -0.007 < 0 = \min_{y,z} \max_x(\psi(x,y,z))$$

So this would mean that Guts does not have a value as it does not align with our minimax theorem when players are playing optimally.

4.5.4. *Time Considerations.*

4.6. $\mathbf{n > 3}$**.** Our algorithm looked primarily at 3 player guts where player y, z are colluding against x. There has not been an analytically determined payoff function for $n > 3$ yet. If and when it is determined, changing the objective function to it will result in the minimax and maximin strategies without much change in the code.

We could potentially look at a simplified 4 player game such as some sort of 4 player rock paper scissors to test this hypothesis. But we have not had time to do such analysis.

5. General Games

The hope of the project is that our methodology is generalizable to any games given either the payoff matrix or the objective function of a game. Where if given the matrix or the objective function, we will be able to return the maximin and the minimax values for the game when player 1 is playing against an asynchronous coalition of players 2-n.

There is some reason to believe it would not work in general cases. As seen earlier, BFGS runs into the issue of falling into local minima, and if the payoff function for a game skews heavily to contain local minima, then it is likely that BFGS will not find the values that

we are looking for. If instead we try using Basin-Hopping as we did for OMO, we may run into the issue of certain games being too big for it to be computationally feasible to run Basin-Hopping at large discretizations.

Finding better methods that reduce run-time and more reliable methods will be the next steps are progressing this project to be generalizable to all games.

References

[vN] J. Von Neumann (1928) Zur Theorie der Gesellschaftsspiele

[BLPWZ] K. Buck, J. Lee, J. Platnick, A. Wheeler, and K. Zumbrun (2022) Continuous guts poker and numerical optimization of generalized recursive games

[CCZ] L. Castronova, Y. Chen, and K. Zumbrun (2021), Game-theoretic analysis of Guts Poker

[HH] Q. Huangfu, J. A. J. Hall (2018), Parallelizing the dual revised simplex method

[WD] D. Wales, J Doyle (1998), Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms