GAME THEORY AND APPLICATIONS

ROBERT MAURA AND NATALIE WEISS

1. Introduction

Game theory has a rich literature that spans many disciplines due to its endless applications. Economists, psychologists and biologists all utilize game theory to approach questions of behavior, competition, and cooperation in their respective fields. Game theory is useful in modeling economic markets and firm interactions. In biology, game theory is used to evaluate species' strategy while competing for scarce resources. Because competition is a driver of natural selection, game theory has provided a novel way to approach evolution and ecology. As one of us is an economist by training and the other a biologist, game theory provided an intersection for us to experiment and learn about our fields from different perspectives.

The general approach for applying game theory is to model a game that accurately reflects the scenario of interest. Once the game is constructed with various players, feasible strategies, and payoffs, the goal is to find equilibriums in the game and expected payoffs for each player. There are many classical games that have known equilibriums and many applications often modify these games by taking into account the priorities of the players. Equilibriums illustrate expected outcomes of games with rational players. Moreover, games allow us to understand what is rational and irrational behavior and can provide a framework to analyze human behavior that deviates from the known equilibriums.

In our paper, we are going to expand on the classic game, the Prisoner's Dilemma, from theory to application. We will describe the basic two person Prisoner's Dilemma, including the optimal strategies and equilibriums and extend these results to an n-person Prisoner's Dilemma game. We would like to get a deeper understanding of the complexity of identifying equilibriums in a Prisoner's Dilemma with more than 2 players. The time complexity of our code will be computed and analyzed. We will continue to develop our understanding of the Prisoner's Dilemma by exploring a repeated Prisoner's Dilemma game, rather than the typical one-shot game. Instead of varying the number of players, we will vary the number of times the interaction occurs. Analyzing how strategies change over multiple interactions simulates real life interactions. The addition of a biological utility function from Dridi and Akcay (2016) in our repeated game will further capture more realistic human behavior by considering players' personal preferences in their payoffs. Overall, we aim to show how dynamic game theory is through the extension and application of a familiar, classical game, the Prisoner's Dilemma.

2. Definitions

2.1. **Game.** A **game** is defined by 3 objects

- $N = \{1, \ldots, N\}$ set of of players.
- The sequence $A_1 = \{a_1^1, ..., a_1^m\}, ..., A_n = \{a_n^1, ..., a_n^m\}$ represent all the feasible strategies that a player can choose.

- A set of functions $f_i : A_1 \times ... \times A_n \mapsto \mathbb{R}^n, i \in [1, ..., n]$ that represent the payoff functions of the player.

2.2. **Pure Strategy.** Player i has a pure strategy if for a feasible strategy vector $A_i = (a_i^1, ..., a_i^n)$, the element a_i^j will be played with probability 1, for $1 \leq j \leq n$, and element a_i^k will be played with a probability 0, for all k $\neq$ j.

2.3. **Mixed Strategy.** Player i has a mixed strategy if for a feasible strategy vector $A_i = (a_i^1, ..., a_i^n)$, the element $a_i^j > 0$, $1 \leq j \leq n$, and $a_i^1 + ... + a_i^j + ... + a_i^n = 1$.

Note: This implies that actions are played with varying probabilities.

2.4. **Dominant Strategy.** We will say that $a_i^{k_i} \in A_i$ is a dominant strategy for player i, iff, $\forall(a_1^{k_1}, ..., a_{i-1}^{k_{i-1}}, a_{i+1}^{k_{i+1}} ..., a_n^{k_n}) \in A_1 \times ... \times A_{i-1} \times A_{i+1} ... \times A_n$, and $f_i(a_1^{k_1}, ..., a_n^{k_n}) = \max_{b \in A_i} f_i(a_1^{k_1}, ..., a_{i-1}^{k_{i-1}}, b, a_{i+1}^{k_{i+1}} ..., a_n^{k_n})$

Note: The intuitive idea behind this definition is that a strategy is dominant when we would always choose that action (because we will obtain the maximum payoff), regardless of the strategy other players choose. The dominant strategy is an example of a pure strategy.

2.5. **Dominant Equilibrium.** $(a_1^{k_1}, ..., a_n^{k_n})$ is a dominant equilibrium $\Leftrightarrow \forall i \in [1, n], a_i^{k_i}$ is a dominant strategy for player i.

2.6. **Nash Equilibrium.** $(a_1^{k_1}, ..., a_n^{k_n})$ is a Nash equilibrium iff
$\forall i \in [1, n]$ $f(a_1^{k_1}, ..., a_n^{k_n}) = \max_{b \in A_i} f(a_1^{k_1}, ..., a_{i-1}^{k_{i-1}}, b, a_{i+1}^{k_{i+1}} ..., a_n^{k_n})$

Note: The intuitive idea behind this definition is that if all of the players had a default strategy and all players knew everyone's default strategy, none of the players would defect from their strategy.

Lemma: every dominant equilibrium is also a Nash equilibrium.

2.7. **Non-Zero sum game.** $(N, A_1, ..., A_n, f_1, ..., f_n)$ is a zero sum game iff $\sum_{i=1}^{n} f_i(a_1, ..., a_n) \neq 0, \forall(a_1, ..., a_n) \in A_1 \times ... \times A_n$

Note: A zero sum game is a game where the totals gains of the players are equivalent to the total losses of the rest of the players.

3. Prisoner's Dilemma

Prisoner's Dilemma is one of the most classic and well-known games described in game theory. The game is played like this:

Imagine two prisoners, both part of a criminal gang, and taken into custody. Each prisoner is put into different rooms and questioned by the investigator. The investigator has enough evidence to charge each prisoner with a lesser crime, but not enough evidence to charge the pair for the major crime. The prisoners have two options: to either rat out their partner, cooperating with the investigator and confessing their crime, or deny that they have anything to do with the crime. If both confess to the investigator, then they both receive a shortened sentence for the larger crime. If one confesses and the other denies, the one that confesses is let free and the other prisoner is sentenced to the full length of the larger sentence. If both deny

participation in the crime, they are both sent to prison for the minor crime.

The game can be illustrated as a payoff matrix for the two players shown below.

	Player 2 deny	Player 2 confess
Player 1 deny	(1,1)	(5,0)
Player 1 confess	(0,5)	(3,3)

Analyzing the payoff matrix, the dominant strategy for both players is to confess. This is because the prisoner would always receive a lower payoff regardless of the action of the other player. The action confess/confess is a dominant equilibrium in the game and the players would each receive a sentence of 3 years. The caveat is that if either player knew the other was going to deny, they would also choose deny. There is temptation for one player to switch to confess and go free, but that opens the risk for the other player to also switch to confess. There is no impetus to switch because then both would receive a higher sentence at the confess/confess equilibrium. Deny/deny is therefore a Nash Equilibrium because there is no incentive to change strategies. The Prisoner's Dilemma game is interesting because it is paradoxical and displays the power of equilibriums.

3.1. **N-Players Prisoner's Dilemma with Pure Strategies.** Given the fact that the 2-players version of problem is computationally very simple, we have created n-players prisoner's dilemma programs to increase its complexity. Our programs will take the payoff matrix as an input and will give the Nash-equilibriums and dominant strategies of the players as an output.

3.1.1. *Description of the program.* Each player has a $2\times2\times...(n)...\times2$ payoff tensor. We decided it would be much easier to present the problem instead as a $(2^{(n-1)})\times2$ matrix with all the possible combinations reordered. As an example, let's imagine a 3-player case, and each player's payoff matrix:

Player 1 payoff matrix (note: P1 = C⇒player one confess;P1 = D ⇒player 1 deny):

	P2=C, P3=C	P2=C, P3=D	P2=D, P3=C	P2=D, P3=D
Player 1 deny	3 years in prison	2 years in prison	2	0
Player 1 confess	9	3	3	1

Player 2 payoff matrix:

	P1=C, P3=C	P1=C, P3=D	P1=D, P3=C	P1=D, P3=D
Player 2 deny	3	2	2	0
Player 2 confess	9	3	3	1

Player 2 payoff matrix:

	P1=C, P2=C	P1=C, P2=D	P1=D, P2=C	P1=D, P2=D
Player 3 deny	3	2	2	0
Player 3 confess	9	3	3	1

In our nomenclature , these 3 matrices will be elements of a vector called Prisoner:
Prisoner(. , . , i) = the payoff matrix of the i-th payer, for any 0<i<n+1
Prisoner(. , j , i) = j-th column of the i-th player's payoff matrix, for any $0 \leq j < 2^{n+1}$

Prisoner(k, j , i) = k row (confess or deny row) of the j-th column of the i-th matrix. Our MATLAB code for checking dominant equilibriums is below.

```
function n_players(n, Prisoner)
  # Every player has a 2x2x...(n)...x2 payoffs matrix
  # instead we are going to use a (2^(n-1))x2 matrix
  #the (2^(n-1))x2 matrix is the Prisoner input

 # we will have a dominant equilibrium
 #if and only if all players have a dominant strategy
 # We assume that there exist a dominant
 #strategy until we find a prisoner without it
  Dominant = 1


  PrisonerDominant = [ ]


    # Dominant strategy analysis for Prisoner i:
  for i = [1:n]
    confess = 1; # we are assuming that he confess
    #-> ie, he prefers to confess in all columns
    deny = 1;  # we are going to assume also that  he deny
    #-> he prefers to deny in all columns

    for j = [1:2^(n-1)]
      if max(Prisoner( : ,j, i)) == Prisoner( 1 , j, i)
      #comparing the maximum with the first row,
      #if they are the same, then you confess
      # so you do not deny:
        deny = 0;
      else
        confess = 0;
      end
      if deny == 0 & confess == 0
          #you don't have a dominant strategy
        break
      end
    end
    if confess == 1
      fprintf('Prisoner%ds dominant strategy is confess \n', i);
      PrisonerDominant = [PrisonerDominant, 'confess, '];
    elseif deny == 1
        fprintf('Prisoner%ds dominant strategy is deny \n', i);
        PrisonerDominant = [PrisonerDominant, 'deny, '];
```

```
    else
        fprintf('No dominant strategy for Prisoner%ds \n', i)
        Dominant = 0;
        break
    end
end

# Dominant strategy equilibrium analysis for the game:
if Dominant
    fprintf('(');
    for word = PrisonerDominant
      fprintf(word);
    end
    fprintf(' ) is a dominant strategy equilibrium \n')
end
```

We also created a program that would find Nash-equilibriums in a n-players game. To solve this problem, we utilized binary numbers and booleans. We noticed that we could identify each combination of strategies with a binary vector (e.g.(1, 0, 0, 1) = (player 1 deny, player 2 confess, player 3 confess, player 4 deny)). So what we did was create a program that, given the matrix 'Prisoners' as an input, checked if any of the 2^n possibles combinations was a Nash-equilibrium. First it creates the vector that we want to check. Then, we check for each player the column of interest, search the vector, and observe whether in that column he would change his decision.

E.g, lets take a 'z' from [0,2^{n-1}] (*note* : $(X)_b$ = number X in base b):
In this example we will use $Z = (2)_{10} = (0, 1, 0)_2$ = (player 1 deny, player 2 confess, player 3 deny)
Let's check if player 1 would change his strategy.

(Note that the column that we must check can be calculated taking the binary vector, transforming it into the decimal base, and adding 1. In this case $(., 1, 0) \rightarrow (10)_2 = (2)_{10} \rightarrow j = 2 + 1 =$ 3rd column)

	P2=C, P3=C	P2=C, P3=D	**P2=D, P3=C**	P2=D, P3=D
Player 1 deny	3	2	**2**	0
Player 1 confess	9	3	**3**	1

He prefers to confess ("2 years in jail" is better than "3 years in jail"), and in the vector that we were checking, the first element was 0 (confess). So far, everything is correct. Let's check if player 2 would change his strategy:
(0, ., 0)$\rightarrow (0)_2 \rightarrow (0)_{10} \rightarrow j = 0 + 1 \rightarrow$ 1st column

	P1=C, P3=C	P1=C, P3=D	P1=D, P3=C	P1=D, P3=D
Player 1 deny	**3**	2	2	0
Player 1 confess	**9**	3	3	1

He prefers to confess ("3 years in jail" better than "9 years in jail"), but in the vector that we were checking, the second element was 1 (deny). Then, (0,1,0) is not

a Nash equilibrium. Below, is our MATLAB code.

```
function []= n_playersNE(n, Prisoner)


  #Nash equilibrium analyses for Prisoner i
  # nomenclature -> 1 = deny; 0 confess

  confess=0;
  deny=0;

for z = [0:2^n - 1]
#we check each combination
#of strategies of the n players
  itIsNash = 1;
  # we assume that this
  #is a Nash-equilibrium until we prove the contrary
  str = dec2bin(z, n);
  # transform the number in a n-binary vector
  for i = [1:n] #check if the i-th player would change his strategy
      column = [str(1:i-1), str(i+1: n)];
      j = bin2dec(num2str(column)) + 1;
      #identifying the column that we need to check
      a = (Prisoner(1, j, i)<Prisoner(2, j, i));
      #booleans. What he prefers to do?
      #note that row 1= confess; row 2 = deny
      # a=0 he prefers  'confess'; a=1 he prefers 'deny'
      b = bin2dec(num2str(str(i)));
      # b is what player i is doing. Would he change that?

      if a ~= b # notice a = b iif it is a Nash equilibrium
          itIsNash = 0;
          break
      end
  end
  if itIsNash #if z was finally a Nash - equilibrium, we will writte it
      fprintf('\nwe have this Nash equilibrium:\n')
      player = 1;
      for k = str
          k = bin2dec(num2str(k));
          if k
              fprintf('player%d deny\n', player);
          else
              fprintf('player %d confess\n', player)
          end
```

```
            player = player + 1;
  end
end

end
end
```

3.1.3. *Complexity.* The function that finds dominant equilibriums will enter in a n-loop with 2 operations, an *if-elseif-else* (that we will count as 2 operations) and a 2^{n-1}-loop. The 2^{n-1}-loop has an *if* and an *if-else* that we will count as 2 operations; and finally an *if* with a n-loop. Therefore, we can count $n \times (2+2+2^{n-1} \times 2)+n = 5n + n2^n$. Thus, the order of complexity is $O(n2^n)$, which means that the program is really complex and slow for a big n. Actually, only knowing that we had to enter in a 2^{n-1}-loop, was enough to say that the complexity was at least $O(2^{n-1})$, which is already quite complex.

The function that finds Nash equilibriums will do 2 operations, then enter in a 2^n-loop with 2 operations. Then will enter a n-loop (this includes 5 operations), and after that loop, an *if* (this includes 2 operations and a n-loop with 3 operations). Therefore, we can count $2+2^n \times (2+n \times (5))+1 \times (2+n3) = 2+2^{n+1}+5n \times 2^n+2+3n$. Thus, the order of complexity is $O(2^{n+1})$.

3.2. **Pure vs.Mixed strategies.** The dominant strategies and strategies that provide Nash equilibriums found in our program are all pure strategies (the player chooses the same action each time and this action maximizes payoff). The cause of this is the fact that we are modeling one-shot games. However, in repeated games, mixed strategies may provide a higher payoff then a pure strategy. In the next section we explore mixed strategies for zero-sum games.

4. Mixed Strategies for zero-sum games

For zero-sum games, there may not always be a dominant strategy for both players, and furthermore dominant equilibrium; however the Nash equilibrium can be achieved through mixed strategies. The mixed strategy solution to a zero-sum game may also be referred to as the minimax solution, because the payoff simultaneously minimizes the maximum losses for player 2 and maximizes the minimum gains for player 1. This value guarantees a maximum minimum gain for player 1 and a minimum maximum loss for player 2 and therefore, neither player would change their strategy.

For the special case when one player has a dominant strategy, the other player is forced to play a pure strategy as a best response to the other's dominant strategy. This strategy is not necessarily a dominant strategy for the player, but the best option given the other's choice. Each player can achieve the Nash Equilibrium by employing these pure strategies. The payoff at the Nash Equilibrium simultaneously achieves minimax for both player 1 and player 2 and is referred to as a saddle point. The example matrix below illustrates a matrix where a saddle point is present.

For the payoff matrix

$$\begin{pmatrix} 1 & 2 \\ 0 & 4 \end{pmatrix}$$

there is a saddle point at 1 because it is both the maximum in its column and the minimum in its row. Player 1 will win at least 1 and player 2 will lose at most 1. To check that this is true, it is clear that player 2 will choose column 1 with probability 1 because it is the dominant strategy. Column 1 will minimize losses, regardless of what player 1 chooses. Player 1, knowing that player 2 will choose column 1, will always play row 1 because it maximizes his gain in this column. Each player will play to the saddle point with probability 1. Furthermore, the value of the saddle point is the exact value player 1 will expect to win, and player 2 to lose.

Our code allows us to check for saddle points in a matrix, and if none exist, will find the mixed strategies for a two person, zero sum game. Using the fact that player 1 plays their first action with probability p and their second action with probability *1-p* and player 2 plays their first action with probability q and their second action with probability *1-q*, we can use the payoff matrix to find the strategy that ensures that player 1 maximizes their average payoff regardless of what the other person plays. Because we can solve for the average payoff for each player by just looking at their own payoff matrix, this represents a non-cooperative game.

```
Function = mixStrat(M)
    #M(1,1)*p + M(2,1)*(1-p) = M(1,2)*p + M(2,2)*(1-p)
#equalizing strategy
    #mix strategy for 2x2 matrix, (zero-sum prisoner's dilemma)

    p=(M(2,2)-M(2,1))/(M(1,1)+M(2,2)-M(2,1)-M(1,2));
    q=(M(2,2)-M(1,2))/(M(1,1)+M(2,2)-M(2,1)-M(1,2));

    dimensions=size(M);
    m=dimensions(2);
    n=dimensions(1);

    ##minimax thm
    if p>1
        for i=[1:n]
            for j=[1:m]
     if min(M(i,:))==max(M(:,j))
         saddle_pt=min(M(i,:))
     fprintf('there is a saddle point\n')
     end
            end
        end
    end

    prob1=[p,1-p];
```

```
prob2=[q,1-q];
```

5. Utilities

Utility functions are beneficial in modeling more realistic games without adding too much complexity. In real life, every player comes to the table with a different set of experiences and priorities. They do not interpret payoffs the same way and may not play to maximize their raw payoff. Some players may gain more reward, or find higher utility, in certain payoffs or outcomes. Utility functions allow us to more accurately understand the motivations of the players and overall behavior. A very basic scenario will illustrate the concept of utilities:

Let us pretend we have two friends, one is a mathematician and one is an economist. The mathematician makes $100 a day and the economist makes $1,000,000 a day (sorry, Professor). If each receives a pay raise of $100 per day, you would expect that the mathematician would appreciate this raise more, and therefore he would receive a higher utility from the raise.

In the previous example, the $100 raise is not valued equally between the mathematician and the economist due to their existing paychecks. Furthermore, the additional $100 bonus will affect their future actions; maybe the mathematician will splurge on the newest graphing calculator whereas the economist may not change his spending habits. The mathematician gained greater utility in the raise and his future motivations for choices have shifted. Utility functions can be simply any function and it is up to the modeler's discretion to employ a function that captures the behavioral characteristics of interest. For our experiments, we looked at a biological utility function in the same basic iterated Prisoner's Dilemma game. The utility function illustrates other-regarding preferences and altruistic actions in non-cooperative games. The model outlined below is taken from Dridi and Akcay's paper (2016) and is representative of biological learning models. Each player starts off with initial motivations for the two person prisoner dilemma game actions, confess or deny. Motivations for an action directly affect the probability of the player to choose an action. This makes intuitive sense because if a player has a high motivation for an action, they are more likely to take that action. The vector of probabilities corresponding to contribution levels are calculated by the equation,

$$p_{i,t}(a_i) = \frac{e^{M_i(a_i)\lambda}}{\sum_{b \in A_i} e^{M_i(b_i)\lambda}} \tag{1}$$

where $M_i(a_i)$ is the current motivation for the a_i, the action taken, λ is the exploration parameter, and $M_i(b_i)$ are the motivations for the actions not taken by the player at time, t. A high λ shows a player that is very sensitive to their motivations, and a low λ represents a player that would take each action with a uniform probability. This equation accurately reflects the influence of motivations on choosing actions and also how sensitive the player is to listening to these motivations through the λ parameter.

After each round, the player receives a payoff, and must update their motivations based on the utility they gained from their payoff. Motivations are updated each round under the form,

$$M_{i,t+1}(a_i) = (1 - \frac{1}{t})(M_{i,t}(a_i)) + \frac{1}{t}(u(a_i)) \tag{2}$$

where $M_{i,t+1}(a_i)$ is the updated motivation for player i for the action a_i, the coefficient $(1 - \frac{1}{t})$ is the dynamic learning rate for the player, $M_{i,t}(a_i)$ is the previous motivation for the action, and $u(a_i)$ is the utility gained from taking the action. As t approaches infinity, the learning rate converges to 1. This essentially means that the player has stopped learning and only bases his actions on his motivations. The exciting part is the utility gained from the action. Depending on whether the utility is positive or negative, the player will either be encouraged or discouraged to retake the action respectively. The utility function (Dridi and Akcay, 2016) that we analyzed is

$$\pi_i(a_i) + \beta(\pi_i)(\sum_{j \neq i} \frac{\pi_j(\mathbf{a})}{N+1}) + \alpha(\sum_{j \neq i} \frac{\pi_j(\mathbf{a})}{N+1}) + \gamma \left| \pi_I - (\sum_{j \neq i} \frac{\pi_j(\mathbf{a})}{N+1}) \right| \tag{3}$$

where $pi_i(a_i)$ is the payoff for the action taken by player i, $\sum_{j \neq i} \frac{\pi_j(\mathbf{a})}{N+1}$ is the average payoff for other players in the game, and α, β, and γ are genetic traits that capture different aspects of other regarding preferences. The parameter α represents purely altruistic characteristics; when α is positive, the player aims to maximize other people's payoffs. Conversely, when α is negative, the player aims to minimize other players' payoffs. The β parameter can be understood as a competitiveness factor. When β is positive, the player wants to do well when others are also doing well; and when β is negative, the player wants to do well when others are doing poorly. The γ parameters show aversion to inequality between the payoffs of players in the game.

5.1. **Monte Carlo Simulations.** We will use this model to simulate iterated Prisoner's Dilemma games with players who have varying other-regarding preferences. This will test if game action converges to the Nash Equilibrium with players that are not purely motivated by payoffs.

Each Monte Carlo Simulation simulates a 20-round iterated Prisoner's Dilemma game, 100 times with the 2 by 2 payoff matrix below.

	Player 2 deny	Player 2 confess
Player 1 deny	(5,5)	(2,1)
Player 1 confess	(1,2)	(3,3)

Each player starts with the initial motivation for each action set to 1. Player 1 is in blue, and player 2 is in red.

In Figure 1, all of the 100 simulations reach the Nash Equilibrium of both players denying by round 20. This is what we would expect and shows that the model can replicate the normal Prisoner's Dilemma scenario.

For the next simulation, we tested a game with players who have different β parameters, the competitiveness coefficient. You would expect that player 1's mindset in this game is to do well and also have his or her counterpart do well. This is illustrated in Figure 2 from the fact that in each simulation, player 1 either always plays a pure strategy of confess or a pure strategy of deny. The confess/confess and

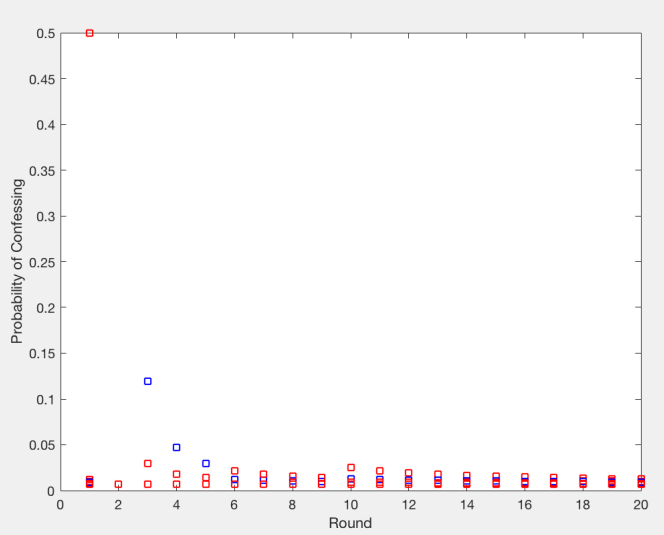

FIGURE 1. $\alpha_{1,2}$=$\beta_{1,2}$=$\gamma_{1,2}$=0 and $\lambda_{1,2}$=1
. This is a simulation of the normal, payoff based Prisoner's Dilemma game because all utility specific parameters are zero.

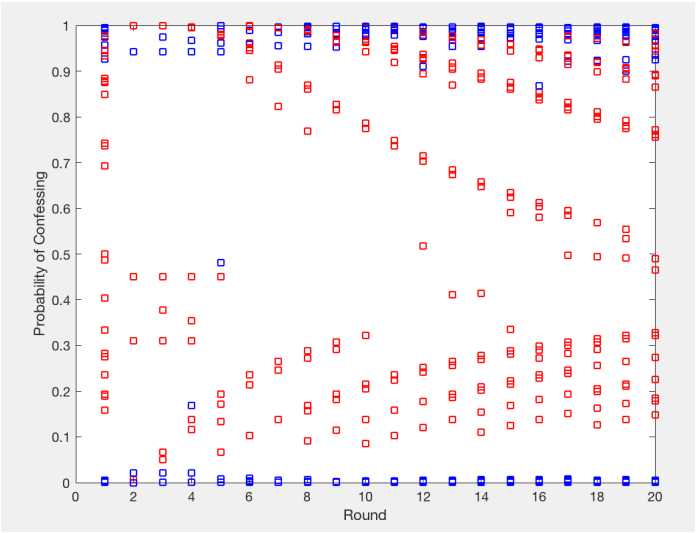

FIGURE 2. $\alpha_{1,2}$=$\gamma_{1,2}$=0, $\beta_1 = 0.5$, $\beta_2 = -0.5$, $\lambda_{1,2}$=1

deny/deny equilibriums offer both players the same payoff in the matrix. Player 1 is playing towards this set of outcomes because the payoffs for both are equally good or equally bad. On the other hand, player 2's competitiveness takes the form of doing well when others are doing poorly or doing poorly when others are doing well. Player 2 takes advantage of the fact that player 1 continues to play the same action by progressively defecting from the equilibrium and widening the payoff gap. This is illustrated by the fact that at the end of each simulation, player 2 is trending towards playing both confess and deny with probability 0.5. It is worth noting that player 2 tends to defect from the deny/deny equilibrium more than the confess/confess equilibrium.

Lastly, we performed a simulation with two players that are both averse to inequality and altruistic. Player 1 is more strongly averse to inequality compared to player 2, but both players are almost equally altruistic. From Figure 3, you can see that

they both tend towards the equilibrium of confess/confess, but in two simulations player 2 plays deny almost exclusively. In two simulations player 1 plays a mixed strategy of playing confess with probability of 0.8. This shows that players' parameters can alter the game play and that the actions may not reflect equilibriums. Out of 100 simulations, two deviated from the general pattern. These simulations are illustrative of the unpredictability of behavior.

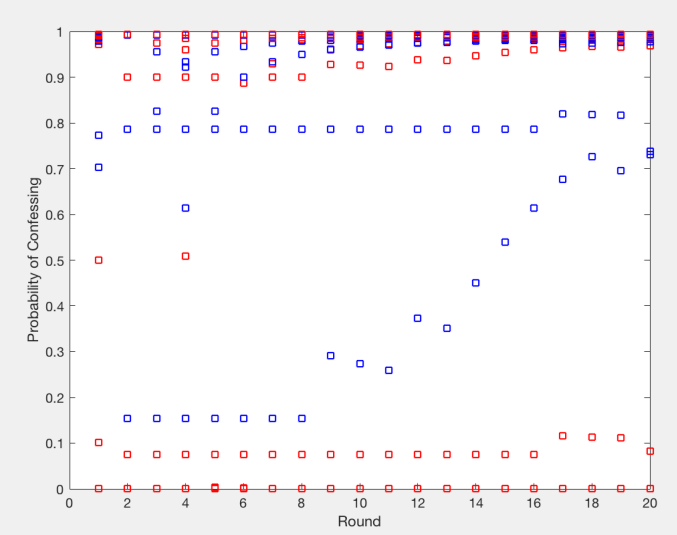

FIGURE 3. α_1=0.6, α_2=0.7, γ_1=-0.9, γ_2=-0.2 $\beta_{1,2}$=0, $\lambda_{1,2}$=1

As displayed in the simulations, utility functions can add a realistic spin to classical games. Utility functions can be crafted to test different behaviors players may exhibit and captures the variability in human decision making.

6. Conclusion

One of the main conclusions that we have extracted from this work is the enormous range of possibilities that game theory represents in models. Moreover, in the beginning of the project we showed how, even the most simple game like a n-players prisoner's dilemma, can be really complex and worthy to study (let's remember that it is very inefficient and slow even to answer a simple question like if there is a Nash equilibrium in the game).

Beyond the time complexity of solving games, the human behavior that games try to capture can be just as complex. Utility functions allow us to understand the priorities of the players, but the interactions between different utility functions can be unpredictable and difficult to analyze. Equilibriums offer insight to rational play based on pure payoffs, but when players' inner motivations are taken into account, the game play may not lead to the same rational play equilibriums. The profound discoveries in mathematics, economics and biology from studying game theory have revolutionized the way we think about human behavior, interactions, and strategy. We are excited to see where game theory will take us next!

7. Contributions

For this project, Robert focused on the Dominant Equilibrium and Nash Equilibrium codes and Natalie focused on the Mixed Strategy code, iterated Prisoner's Dilemma model, and simulations of the model.

8. References

Dridi, S. and Akcay, E. (2016). Learning to Cooperate: The evolution of social rewards in repeated interactions. Still in pre-print.

Geckil, I. and Anderson, P. (2009). Applied game theory and strategic behavior. Chapman and Hall/CRC.