First Draft of MATH 320 Final Paper

Robert Maura and Natalie Weiss

Introduction

Game theory has a rich literature that spans over many disciplines and its applications are endless. Game theory aims to understand strategies of players in various types of games and scenarios, competition, and whether there are stable equilibrium points for strategies of players. The most common application of game theory is in modeling economic markets and how firms will interact with each other; however game theory can also be used in biology to understand interactions between species and within individuals of the same species. Competition for resources and cooperation are the central themes in evolution and ecology, and game theory has offered a novel way to understand these interactions. As one of us is an economist by training and the other a biologist, game theory provided an intersection point for us to experiment and learn about our fields from different perspectives.

There are various types of classical games in game theory: zero-sum games, cooperative and non-cooperative games, perfect information games, and repeated games. A game where one player's losses equals another player's gains would be a zero-sum game, aptly named because the sum of payoffs will equal zero. Cooperative games model a scenario where the players can work together to optimize their payoffs and create joint strategies. Non-cooperative games model the opposite--each player is on their own in deciding strategies to play and predicting what the other player may do. There is no contracts or alliances between players in non-cooperative games. Lastly, games of perfect information model games where each player knows the move of player preceding him. Each player moves sequentially and therefore, each player knows the exact move and strategy of the other person. A game of imperfect information shows a scenario where each player moves at the exact same time and each player does not know what the other may do. These games could be played over many rounds, as repeated games, or only once, as a one-shot game. Repeated games are interesting because it allows us to test whether or not strategies change or evolve over many rounds.

Our project will explore the classical game of Prisoner's Dilemma. First, we will describe the basic two person Prisoner's Dilemma, the optimal pure strategies and Nash equilibriums. We will extend these results to include an n-person Prisoner's Dilemma game and the time and space complexity of our code will be computed. Second, we will find the mixed strategies based on the minimax theorem of a two player Prisoner's Dilemma game and the resulting payoffs. Third, we will add a more realistic spin on these games by adding utility functions and motivations for certain actions for each player. Game theory assumes rational play, but humans do not always act rational and may be inclined to act in a certain way based on their genetic parameters or past experiences. We will explore a biological utility function, based on the 2016 paper by Dridi and Akcay, and an economic utility function that models risk-averse and risk loving players.

Prisoner's Dilemma

Prisoner’s Dilemma is one of the most classic and well known games described in game theory. It was initially developed by Flood and Dresher in the early 1950’s and then formally solved by Tucker (SOURCES). The game is played like this:

Imagine two prisoners, both part of a criminal gang, and taken into custody. Each prisoner is taken into a different room 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 larger crime and longer sentence. The prisoners have two options: to either rat out their partner, cooperating with the investigator, or deny that they have anything to do with the crime. If both cooperate with the investigator, then they both receive a shortened sentence for the larger crime. If one cooperates and the other denies, the one that cooperates 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 prisoner for the minor crime.

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

Prisoner 1/Prisoner 2	Confess	Deny
Confess	(-3,-3)	(0,-5)
Deny	(-5,0)	(-1,-1)

Analyzing the payoff matrix, you would think that both would choose to condess because it gives a higher payoff for both strategies, however you would receive the highest payoff if both deny. Because it is advantageous for both to defect, this is illustrative of a Nash equilibrium for the Prisoner’s Dilemma. There is a Nash equilibrium for a game if both players are playing a strategy that will maximize payoff and switching strategies will not further maximize for the player. In a stable Nash Equilibrium, there is no incentive to change strategies, as seen in the Prisoner’s Dilemma.

Our take on Prisoner’s Dilemma:
(n players--only pure strategies)
Given the fact that the 2-players version of problem is computationally very simple, to increase its complexity, implement new interesting functions, we have created n-players prisoner’s dilemma programs. Our programs will take as an input the matrix of payoffs and will give the Nash-equilibriums and dominants strategies of the players as an output.

Before explaining the program, we will define a few concepts:

Strategy

In game theory, a strategy is any of the options that a player can choose in a setting. More formally, for any ‘i’ of the ‘n’ players, given a finite set τ_i = {X$_1$, …, X$_m$} called set of feasible strategies, each of its elements X$_k$ is a strategy. For example, in the Prisoner’s dilemma, the 2 possible strategies for any of the players are ‘deny’ and ‘confess’.

Dominant strategy

Given a finite strategy set τ_i = {X$_1$, …, X$_m$}, X$_k$ is a dominant strategy for the i-th player if, and only if, the payoff of the strategy is greater than the payoff of any other strategy, despite the decisions of the other players. It is important to say that there is an existence of a dominant strategy for any player because dominant strategies are not guaranteed. Add an example

Dominant strategy equilibrium

(X$^1_{k1}$, X$^2_{k2}$, ..., (X^n)$_{kn}$) is a dominant strategy equilibrium if for any player ‘i’ , (X^i)$_{ki}$ is a dominant strategy.

Nash equilibrium

(X$^1_{k1}$, X$^2_{k2}$, ..., (X^n)$_{kn}$) is a Nash-equilibrium if, and only if, for any player ‘i’, the payoff of (X$^1_{k1}$, X$^2_{k2}$, ..., (X^n)$_{kn}$) is not going to increase for him if only he changes his strategy.

Description of the program

Each player has a 2x2x...(n)...x2 payoffs matrix. We considered that it would be much easier to present the problem instead with a (2^(n-1))x2 matrix with all the possible combinations reordered. As an example, let's imagine a 3-player case, and each player payoff matrix:

Player 1 payoff matrix:

	2 confess, 3 confess	2 confess, 3 deny	2 deny, 3 confess	2 deny, 3 deny
1 confess	3 years in prison	2 years in prison	2...	0
1 deny	9	3	3	1

Player 2 payoff matrix:

	1 confess, 3 confess	1 confess, 3 deny	1 deny, 3 confess	1 deny, 3 deny
2 confess	3	2	2	0
2 deny	9	3	3	1

Player 2 payoff matrix:

	1 confess, 2 confess	1 confess, 2 deny	1 deny, 2 confess	1 deny, 2 deny
3 confess	3	2	2	0

3 deny	9	3	3	1

In our nomenclature, this 3 matrices will be elements of a n 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<j<2^n + 1
Prisoner(k, j , i) = k row (confess or deny row) of the j-th column of the i-th matrix

```
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

 % 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 don’t 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 could show here a graph of the time that this program spended depending on “n”, and maybe calculate the complexity

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 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 that we want, looking at the vector, and we observe whether in that column he would change his decision.

E.g, lets take a ‘z’ from [0, 2^n - 1]
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 can be calculated taking the binary vector, transforming it into the decimal base, and adding 1. In this case (., 1, 0) -> 10_2 = 2_{10}-> j = 2 + 1 = 3rd column)

	2 confess, 3 confess (ie (. , 0, 0))	2 confess, 3 deny (ie (. , 0, 1))	2 deny, 3 confess (ie (. , 1, 0))	2 deny, 3 deny (ie (. , 1, 1))
1 confess (ie (1, ., .))	-3	-2	-2	-0
1 deny	-9	-3	-3	-1

He prefers to confess (-2 (2 years in jail) > -3 (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) -> 1st column

	2 confess, 3 confess (ie (. , 0, 0))	2 confess, 3 deny (ie (. , 0, 1))	2 deny, 3 confess (ie (. , 1, 0))	2 deny, 3 deny (ie (. , 1, 1))
1 confess (ie (1, ., .))	-3	-2	-2	-0
1 deny	-9	-3	-3	-1

He prefers to confess (-3 > -9), but in the vector that we were checking, the second element was 1 (deny). Then, (0,1,0) is not a Nash equilibrium.

```
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 (ie dec2bin(z, n) =
  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, let’s write 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
```

As you can see, the dominant strategies and strategies that provide Nash equilibrium strategies found in our program are all pure strategies. Pure strategies are when the player chooses the same action each time and this action optimizes payoff. However, in some scenarios, mixed strategies, or where the probabilities of playing more than one action are active (0<p<1). In the next section we explore mixed strategies for zero-sum games.

Mixed Strategies for zero-sum games:

For zero-sum games, there may not always be a pure strategy that maximizes payoffs, but rather a mixed strategy. A mixed strategy is a strategy that plays different options of the game at different probabilities. To find a mixed strategy for the players, we can use the minimax theorem. This theorem states that each zero-sum game has a value that will minimize the maximum losses for player 2 and maximize the minimum games for player 1. A saddle point occurs when there is a payoff that simultaneously achieves minimax for both player 1 and player 2. This saddle point will be played as a pure strategy for both players.

Our code allows us to check for saddle points and mixed strategies for a two person, zero sum game. Using the fact that player 1 plays their first strategy with probability p and their second strategy with probability (1-p), 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 without considering the action of the other player, this represent 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];
```

(Write and add code for mixed strategies for non-zero sum games???)

Utilities

A very important concept in modelling more realistic games is the concept of utilities. In reality, not every player interprets payoffs the same way. Some may put higher value, or find higher utility, in smaller payoffs and vice versa. The way a player interprets their payoffs will influence their future strategy and actions in future rounds of repeated games. For our experiments, we looked at two different types of utility functions in the same basic model: one utility function adapted from economics and one from biology.

The basic model is taken from Dridi and Akcay's paper (2016) and is representative of other learning models. Each player starts off with initial motivations to either cooperate or defect. These motivations are updated at each round based on pure payoffs (a model without utilities) or the utility function for that payoff and the learning rate. The learning rate is (1/t)+1, where t is the current round of the game. Learning declines as more rounds of the games are played because the players have essentially learned the other player's strategies and there is less to learn from each round. One round of the game is then simulated, where each player picks an action based on the probability function below.

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

Where a is the action taken by player i and b are the actions taken by the other players. Lambda is the exploration parameter and M is the motivation for that action for the specific player. As you can see the probability to take an action is based on the motivation to take that action. Since motivations are dictated by the utility gained from the payoff of an action, the players are more motivated to keep playing a strategy that provided a high utility in past rounds. We are able to input different utility functions into this model to understand how utility functions change the strategies and interactions of repeated games.

The biological utility function is also adapted from the Dridi and Akcay's paper (2016), in which they describe a model for freely evolving utilities based on other-regarding genetic parameters. The idea is that player's find utility in payoffs based on certain genetic parameters that capture a person's altruistic tendencies. Some players may be naturally more cooperative and willing to cooperate than others. In the Prisoner's Dilemma scenario, the outcome of repeated game may not be the Nash Equilibrium of defect-defect. Since the scenario of both players cooperating results in the highest payoff, two players that are not purely rational, but rather genetically inclined to cooperate, would be more advantageous for both and lead to cooperation over time. We see that in an iterated Prisoner's Dilemma game for infinite rounds the equilibrium is cooperate-cooperate. This fact has strong implications for the evolution of cooperation within and between species because it suggests that individuals with a higher tendency to cooperate will receive a larger overall payoff overtime and cooperation would evolve in certain populations. Hamilton (1964) demonstrated that relatedness to other individuals can positively affect altruism and cooperation within populations of the same species, and this model provides a game theory approach to explain cooperation in populations.

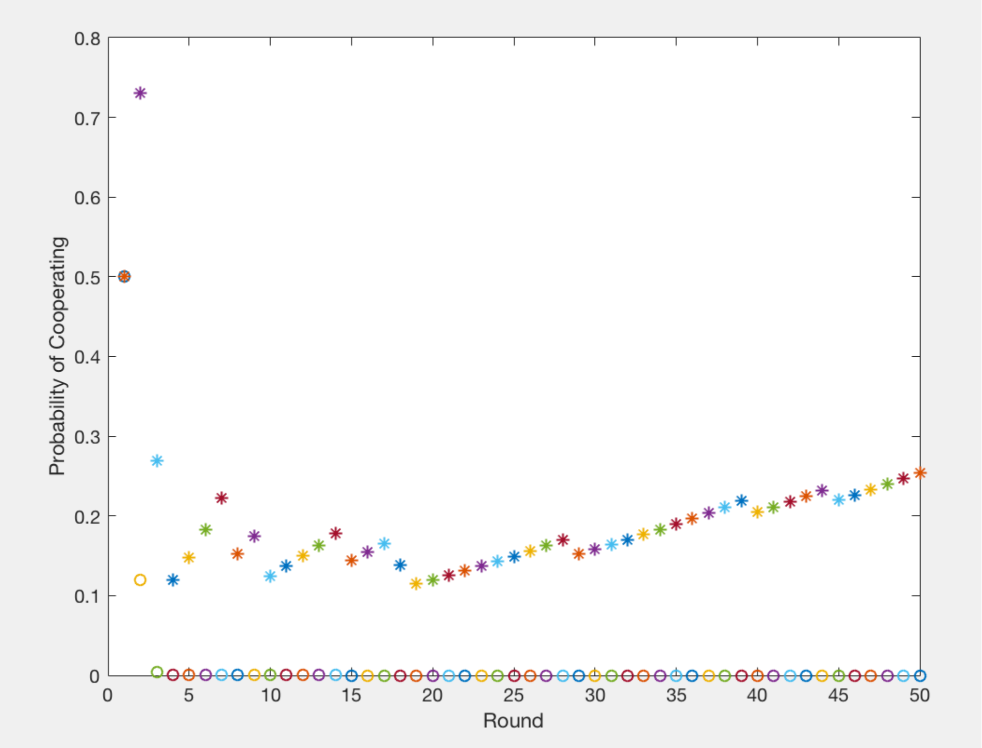
The Dridi and Akcay (2016) function for freely evolving takes this form:

$$u_i(\mathbf{a}) = \pi_i(\mathbf{a}) + \beta(\pi_i(\mathbf{a}) + c + k)(\pi_{-i}(\mathbf{a}) + c + k) + \alpha\pi_{-i}(\mathbf{a}) - \gamma|\pi_i(\mathbf{a}) - \pi_{-i}(\mathbf{a})|.$$

Pi is the payoff for player i for the action that they took during the round. Alpha, beta, and gamma are all genetic parameters (between [-1,1]) that capture different perspectives on how someone may interpret a payoff. Alpha represents purely altruistic characteristics, where a person only cares about others success. The gamma parameters show aversion to inequity, and beta captures multiplicative preferences. Beta can be thought of as the idea that a player wants to do well when others also do well.

(player 1 is o and player 2 is *)

Simulation 1: alpha1=alpha2=beta1=beta2=gamma1=gamma2=0, lambda1=lambda2=1



More interesting simulations...

The economic utility function models player's aversion to risk--either a player is risk averse or risk loving.

$$u(c) = \frac{c^{1-\sigma} - 1}{1 - \sigma}$$

From this utility function, you can see that when sigma=1 and simplifying using L'Hopital's Rule, u(c) goes to log(c). The logarithmic shape of the utility function aptly models risk taking behaviors: for risk loving players, they will be rewarded for their risk and continue to take risks up to a certain threshold when the risk is too great and will start to level off. The same is true for players who are more risk averse, except that their threshold will be lower

Use linear utility function for risk indifferent, exponential for risk averse (log(c) where c is the payoff for that round)

****should we do monte carlo simulations of the utility experiments?? ****

<u>Conclusion</u>