Mathematical Modeling of N-IPD Game

Aubrey Leung, Michael Liston, Matt Shaw, Dongping Wu

University of San Francisco

Abstract

We discuss the results of Iterated Prisoners Dilemma (IPD) simulations with n-agents unaware of the finite, fixed number of iterations on three types of random networks: Configuration, Erdos-Renyi, and Barabasi-Albert. Throughout all simulations, the number of nodes are held constant at n=1000 and the Prisoners Dilemma (PD) is repeated 1000 times. The average degree is arbitrarily manipulated for observation purposes and the temptation to defect is varied across each network. As expected, we observe a negative relationship between the percent of nodes in cooperation and the temptation to defect for every simulation. For every network, simulations show this negative relationship tends to decrease more rapidly as average degree is increased. Comparisons between the three random networks are drawn. Percentage of cooperation tends to decrease more rapidly as variation in degree distribution decreases.

1. Introduction

A plethora of situations follow the structure of the Prisoners Dilemma (PD) including: arms races, antibiotic resistance, or cartel price setting. These real world phenomena are best modeled using an iterated version in which agents play the PD repeatedly and are able to obtain payoff results after each round. In an Iterated Prisoners Dilemma (IPD) the appropriate strategy for rationally self-interested agents is no longer obvious, as agents who cooperate in one round can be rewarded by cooperation in subsequent rounds while those who defect can be punished by defections. When n-agents instead of two engage in an IPD, the PD now has an evolutionary aspect; biological models, in fact, often employ the IPD to study how an agents individual fitness depends on who they interact with within the population. The assumption of homogeneity (i.e. well-mixed populations) fail to reflect empirical observations of PD applications, which lends to applying IPD dynamics on different network structures.

1.1. Prisoner's Dilemma

The simple Prisoners Dilemma involves two agents (x and y) who independently choose to defect or cooperate. The objective of a PD is to reach the Pareto-optimal outcome (C,C) in which both agents have maximized their individual payoffs such that an individual cannot increase their payoff without decreasing the payoff of anothers. The symmetric matrix below identifies the payoffs associated with all four possible outcomes:
$[(C, C); (C, D); (D, C); (D, D)]$

(x, y)	**Cooperate**	**Defect**
Cooperate	R, R	S, T
Defect	T, S	P, P

Table 1: Prisoner's Dilemma

- R = reward payoff each agent receives for both cooperating
- P = punishment each agent receives for both defecting
- T = temptation payoff an agent receives as sole defector
- S = sucker payoff an agent receives as sole cooperator

1.2. Strong Prisoner's Dilemma

A Strong PD is defined as a PD in which $T > R > P > S$.

Both agents have a dominant strategy of playing D. Any two rational agents will choose D and receive payoff P. Any two irrational agents will choose C and receive payoff R, achieving the Pareto-optimal outcome (C,C). D strictly dominates C for both players. The only strong Nash Equilibrium, where each player can only do worse by unilaterally changing its move, is outcome (D, D).

1.3. Weak Prisoners Dilemma

A Weak PD is defined as a PD in which $T > R > P \geq S$ or $T \geq R > P > S$.

The difference in a weak PD is that, for both agents, D does not strictly dominate C. However, D is still weakly dominant in the sense that each player always does at least as well, and sometimes better than the other agent, by playing D. Under these conditions it still seems rational to play DD, which again results in a payoff not Pareto-optimal for both players. Nash Equilibrium cannot be defined in the strong sense, but remains at outcome (D, D) in the weak sense, in which neither agent can improve its position by unilaterally changing its move.

1.4. n-Agent Prisoners Dilemma

The n-Agent Prisoners Dilemma is an extension of the simple PD following the same framework with n agents instead of two. All n agents or nodes will play the PD game with each of its neighbors then subsequently calculate its accumulated payoff.

1.5. Iterated Prisoners Dilemma

In an Iterated Prisoners Dilemma the simple PD is played repeatedly with agents updating their strategy after each round until an equilibrium is reached. Since it is ideal to reach an outcome as quickly as possible, it is often assumed that agents are unaware of how many rounds they will play.

- Suppose all agents knew the game would last exactly n rounds. Using backward induction, it is apparent that (D,D) is the only Nash equilibrium. Regardless of previous rounds, any rational individual agent at round $(n-1)$ faces the simple PD and will thereby defect. At round $(n-2)$ each agent is aware that they will both defect in the next round regardless of what strategy they select in the current round, so it would be rational for them to defect now and in the next round. Rational agents deduce that they should defect in every round by repeating the argument sufficiently many times. Defection is a best response to any move at any round thus no other equilibria exists.

2. Simulation Structure

In each simulation, we had n = 1000 agents or nodes who were all unaware that they would be playing a weak PD 1000 times. All 1000 nodes are initially assigned a strategy. The nodes then play a weak PD with all of their neighbors, compare their cumulative payoff with a randomly selected neighbor and update their strategy for the next round.

2.1. Random Assignment of Strategies

Before any interaction occurs, all one-thousand nodes are independently assigned a strategy to either cooperate or defect. A node will be assigned to cooperate with $P(Cooperate) = 0.5$ or defect with $P(Defect) = 1 - P(Cooperate) = 1 - (0.5) = 0.5$. Thus approximately 500 nodes will start off with strategy C and the other 500 will start with strategy D.

2.2. Engagement of Prisoners Dilemma

Suppose the individual node of interest is x and their single neighbor of interest is y. Without loss of generality, all nodes will engage in a weak PD according to their previously assigned strategy with every node they share an edge with (i.e. all of their neighbors) with the following symmetric payoff matrix:
where $0 < r < 1$.

(x, y)	**Cooperate**	**Defect**
Cooperate	1, 1	0, 1+r
Defect	1+r, 0	0, 0

Table 2: Table caption

From the perspective of a rational node, for whom D is its weakly dominant strategy, the cost of mutual cooperation is $T - R = (1+r) - (1) = r$. As sole cooperator, a node must pay $R - S = (1) - (0) = 1$ to the sole defector. Therefore, the cost-to-benefit ratio is given by $\frac{T-R}{R-S} = \frac{(1+r)-(1)}{(1)-(0)} = r$.

2.3. Comparing Cumulative Payoffs with Randomly Selected Neighbor

All one-thousand individual nodes will randomly select a neighboring node with equally likely probability. Suppose the individual node x shares an edge with 5 other nodes. Each neighboring node of node x has a one-fifth probability of being selected by node x.

All one-thousand individual nodes will compare their cumulative payoffs with the randomly selected neighbor in order to decide to either continue playing the same strategy for the next round or to adopt their neighbors strategy for the next round if their neighbors strategy differs.

2.4. Updating Strategy

Suppose the individual node of interest is x and their randomly selected neighbor of interest is node y. The probability that node x will adopt node ys strategy for the next round is given by the following equation:

$$p_{x\to y} = \frac{P_y - P_x}{(1+r)k_{max}} \tag{1}$$

Where:
P_x = node x's cumulative payoff from all neighboring nodes (similarly for y);
$(1+r)$ = temptation payoff as sole defector, $0 < r < 1$;
k_{max} = the largest degree between x and y (x or y's degree depending on which is greater).

3. Simulation Results on Random Networks

Over three different random networks (Uniform Configuration, Erdos-Renyi, Barabasi-Albert) the weak IPD was simulated with varying average degrees for each graph, as well as varying r-values (range 0-1 in increment of 0.025). Over the course of testing, we held the number of nodes constant at n = 1000 and while repeating the IPD 1000 times per value of r.

Due to a programming oversight, a new network was generated for each simulation with a different value of r and average degree. Although this makes simulation results less accurate for comparisons, the general trends still hold regardless.

3.1. Uniform Configuration Model

The Uniform Configuration Model was structured such that we specified the same degree for each node; thus there is no variation in degree distribution. Suppose the degree $k_i = c$ for each node i=1,2,...,1000is specified where c is the average degree of the network. Each node i is given a total of $k_i = c_i$ stubs of edges. With equally likely probability of any pair occurring, two stubs are randomly selected and an edge is created by connecting the pair of stubs. Another pair is selected then connected from the remaining 2m - 2stubs until all stubs are connected. It is important to note that all pairings of stubs occur with equal probability, but not all networks appear with equal probability.

Since the degree $k_i = c_i$ for each node i = 1, 2, ..., 1000 is always selected to be an even number, the possibility of dangling stubs is eliminated. However, there is still a possibility of self-edges and/or multi-edges occurring. This is not an important issue in practice, since the average number of such edges remains constant as the network becomes larger (i.e. the density tends to zero as n increases).

The way we construct the configuration network in Python:

- Construct a list of stubs (the size of the list depends on the number of players n, and the degree of each player k).
- Create an empty dictionary, the keys are the players, and then fill in the neighbors of each players as the values of each key.
- Assign the first round of the strategies at random, and update them in the later rounds.
- Compute the cumulative payoffs for each player each round.
- Compare the payoffs with one of the neighbors at random, adopt his strategy if he has better payoff with a probability

$$p_{x \to y} = \frac{P_y - P_x}{(1+r)k_{max}} \tag{2}$$

- Start next round.

A negative relationship between the value of r and the percentage of cooperating nodes is observed as expected. Furthermore, the visualization of results show that the percentage of cooperating nodes monotonically decreases faster as the average degree increases. Cooperating nodes are more successful on Uniform Configuration networks with smaller average degrees.

3.2. Erdos-Renyi Model

The Erds-Rnyi Model was structured such that edges were added between each pair of distinct nodes with an independent probability $p = \frac{c}{(n-1)} = \frac{c}{999}$. The total number of edges are no longer fixed and the degree of individual nodes are allowed to vary. For a network of $c << n = 1000$, the degree distribution is well approximated by the Poisson distribution such that $p_k = e^{-c}\frac{c^k}{k!}$.

We are constructing the Erdos-Renyi network based on the probability p:

- Create a dictionary with n keys.
- Appending neighbors into the value of the key by probability p.
- Assign the strategy for the first round and compute the payoffs.
- Compare the payoffs with one of the neighbors and make the decision for the next round.

- Start next round.

3.3. Barabasi-Albert model

The first two networks were random networks consisting of n=1000 nodes where each pair of nodes is connected with independent probability p. There are a number of differences between random and real networks. However, we will only highlight the two primary characteristics that distinguish a Barabasi-Albert (BA) real network: growth and preferential attachment.

Starting with an arbitrary initial network consisting of m_0 nodes such that no dangling nodes exist, growth occurs as the Barabasi-Albert (BA) is generated by adding a new node with $m = \frac{1}{2}c \geq m_0$ stubs to be connected to the m_0 nodes that already exist in the network one at a time until the network had a total of $n = 1000$ nodes. The probability (pi) of a new node connecting to a node i in the existing network is directly proportional to the degree of the existing node (preferential attachment) and is given by $p_i = \frac{k_i}{\sum k_j}$ where k_i = degree of node i and $\sum k_j$ = sum of degrees of all pre-existing nodes = 2current number of edges in network. This process was done entirely by the barabasi_albert_graph(n = 1000, m = 12c) function in the NetworkX Python package.

Due to the property of preferential attachment, new nodes are more likely to connect to nodes with larger degrees than those with small degrees. This is sometimes referred to as the rich-get-richer phenomenon; highly connected nodes obtain new links at the expense of less connected nodes and grow into hubs. The degree distribution has a heavy right skew such that there are many nodes with low degrees and very few hyper-connected hubs. Analytical calculations predict degree distributions follow a power law with pk1k3. However, we will simply characterize the degree distribution qualitatively using the following figure of a typical degree distribution we observed generated by NetworkX.

The same negative relationship seen on the previous network models between the value of r and the percentage of cooperating nodes is observed as expected. Visualization of results again show that the percentage of cooperating nodes decreases faster as the average degree increases. However, cooperating nodes are extremely successful on this network model than on the two previous random network models. When r=1, cooperation went to zero, regardless of average degree, on the previous two random network models. For c=4, the majority of nodes are still cooperating when r=1. For c=4 and c=6, the percentage of nodes cooperating still do not reach zero when r=1.

4. Conclusion

After comparing all three models, the Barabasi-Albert network displayed the highest values of percent cooperation. This held true while playing across the range of r values tested. These results can be explained by the prevalence of highly connected nodes (hubs) which have a higher influence on their neighboring nodes strategy. All three models display a negative relationship between R and total percent cooperation. This was expected as r is our reward variable. In analyzing the average percent cooperation of all three networks, it is important to observe the relationship of percent cooperation and Heterogeneity of a model. In our tests, percent cooperation of models was consistently sorted in decreasing order by the heterogeneity of each model: Barabasi-Albert Erdos Renyi Configuration model. Based on these results, it appears there is a relationship between percent cooperation and heterogeneity of the model. When modeling most real-world phenomena it is best to use Barabasi-Albert network because of its property of high heterogeneity.

5. Reference

[1] Dixit, A. K., Skeath, S. *Games of strategy.*. New York: Norton., 1999.

[2] Newman, M. E. *Networks: An introduction.* Oxford: Oxford University Press, 2010.