The Crowd Dynamic
How Does Opinion Spread in a Network?

Mahika Calyanakoti, Arnab Sircar, Sruthi Kurada

Project Summary:

Project Title: The Crowd Dynamic– How Does Opinion Spread in a Network?

Project Description:
Our project idea is to create a mathematical model that can accurately model the spread of opinion in crowds. Crowd wisdom is especially difficult to model due to the seeming randomness of individual behaviors, but can we gain a glimpse into how wisdom forms through study of opinion formation? It is possible to design a network of individuals and consider a game played on each edge to determine the decisions that are made by each individual. Games on certain clusters and sequences of decisions can impact group behavior as well. Our goal is to create a weighted model that considers all of these phenomena that influence group behavior and predict group proportions on decisions (and see group decisions). Our plan is to test the model by creating a simulation and see if there exists a pattern of opinion spread.

Tasks: Model generation, Formalizing experiment, Determining formulae for model metrics, Creating simulations, Writing paper

Project Concepts: Graph/Graph Algorithms, Social Networks, Game Theory/Auctions/Matching Markets (specifically mixed strategy nash equilibrium)

We plan to generate a model driven by graph theory and degree measures and the influence of decisions of others on personal decisions (game theory). Games will be played on edges of the network between each player. In an iterative process, the payoff functions of each game will be updated after each round of "play" by considering the degree of each node in its respective game.

Changes:
Initially, we were interested in measuring the "wisdom" of crowds. While we are still interested in pursuing this greater goal of assessing crowd wisdom, it was difficult to find data that could be manipulated in a way that would allow us to test hypotheses relating to crowd wisdom. Thus, we decided to analyze the spread of opinion, which serves as *somewhat* of a proxy for crowd wisdom development (or at least a herd-mentality vs polarized opinions). We do not have any

metric on whether the wisdom is accurate– a step that we want to take in the future to understand whether crowds are reliable or provide the best plans of action regarding important decisions.

We also initially wanted to use real-world data to validate our initial model. However, finding time-series spread of opinion data across a network was quite difficult. Ideally, we would like to use statistical techniques to infer previous opinion distributions from stable-state opinion data on networks, but we decided that that could be done in future work.

Implementation and Empirical Analysis:

With the development and proliferation of inexpensive and powerful technologies, more and more people across the world have been able to access information. Increased information access and increased use of online social media platforms for voicing opinions has ushered in the era of crowd participation in various decision-making situations. Already, voting procedures in elections across the world have elicited crowd opinions by definition of democracy for centuries. However, with novel political innovations including referendums, many in the technological sector have questioned whether crowd input can be used for *general* decision-making scenarios. Often, in situations where crowd input is valuable, it is important to determine the attributes of the crowd's *wisdom*. Thus, in our study, we analyze the opinion dynamics of a crowd in which there are social pressures to align opinions with close connections.

Model: Our goal is to model the effect of the opinions of friends or close connections on personal opinions over time. We utilize an iterative method to determine how members of the crowd can change their opinions in response to their close connections in the crowd. In order to model opinion spread and social influence, we make simplifying, but justified assumptions. In this case, model simplicity indicates the existence of a phenomena– this can be studied further by relaxing assumptions. Thus, we construct our model in the following way (*we use the word "node" and "person" interchangeably throughout this described process*):

1. A proposition X is declared to a crowd. For each person in the crowd, we assume that there are only two possible responses to the proposition, and that everyone in the crowd holds a response: (1) Agree with the proposition, (2) Disagree with the proposition.

2. Next, we model our crowd using a graph representation. In our case, we assume a crowd of $N = 1000$ people which are represented by nodes. To model friendships or close connections in the graph, we use edges to connect a pair of nodes that share such a relationship. For the purpose of the study, we generate an Erdős–Rényi random graph to represent our crowd with model $G(N, p) = G(1000, 0.5)$, meaning that for each possible

edge that can exist in the graph, there is a 50% probability that we include it in our crowd model.

3. Now, we "issue" our proposition to the crowd and ask them to respond to the proposition with one of the two possible responses. Thus, we utilize a specific kind of randomization to assign crowd members a response in the beginning (which will be very similar to the way in which crowd members will potentially change their opinions in later iterations of the model, see step (4) below). This assignment of responses serves as the first iteration of the model. Thus, we do the following:

    a. We assume that people in the crowd will make decisions regarding what opinion to take in response to what their close connections (neighbors in the network) decide. Thus, a certain *payoff* can be associated with the decisions that a person makes. It is assumed that individual payoff should be maximized and positive if a person aligns their opinion with a neighbor, as this means that both people across an edge *agree with each other* and are likely to satisfy each others' perceptions of each other and like each other *more*, which is desired. Using similar logic, we realize that it is less desired for close connections to share opposing beliefs, so we determine that opposite opinion selections should have less or negative payoffs. Thus, this social situation can be represented by a game as follows:

		Person 2	
		Agree with X	Disagree with X
Person 1	Agree with X	A, B	-1, -1
	Disagree with X	-1, -1	C, D

    In the first iteration of this game, we assume that $A = B$ and $C = D$. Note that $A, B, C, D > 0$. Also, note that this game is played *on each edge* of the graph, meaning that the social dynamic described above is considered between each pair of individuals. We describe below how the social dynamic is further formalized in step (4).

    b. Notice that in the game described in step (3a), there exists a mixed strategy Nash equilibrium when solving a set of equations for the distributions on which Person 1 and Person 2 should select their opinions. Let p be the probability with which Person 1 agrees with the proposition and q be the probability with which Person 2 agrees with the proposition. The equations are described as follows:

$$Bp + (-1)(1-p) = (-1)p + D(1-p)$$
$$Aq + (-1)(1-q) = (-1)q + C(1-q)$$

Solving for probabilities p and q gives the probability with which Person 1 and Person 2 should select their opinions to maximize their payoffs. Note that a pure strategy Nash equilibrium does exist in some circumstances of the game (always in the first iteration when $A = B$ and $C = D$), but we assume that players will try to optimize for the mixed strategy as there is inherent uncertainty in the opinion they should choose, as well as other factors that influence opinions. We argue that randomization due to mixed strategy Nash equilibria can capture these confounders better than allowing for players to choose the pure strategy.

c. Note that in many instances, a person is connected to several other people, meaning its respective node has a degree greater than 1. This means that the node plays more than one game in one iteration of the model. Thus, how can we decide what opinion this node will assume in a given iteration if we choose separate opinions from different games? We solve this issue by considering the average probability p for agreeing with the proposition across all games that a node plays. Then based on that probability, we assign the node a boolean value that indicates whether its respective person agrees (true) or disagrees (false) with proposition X.

4. In step (3), we describe the set-up for the network and the set-up for the games played between pairs of people in the network. This initialization process also serves as a method for carrying out the first iteration. However, we must now consider the effects of social influence on the opinions that people in the network choose. To do this, we consider the opinions held by closely connected individuals represented as neighboring nodes to a given node. Note that after the first iteration is complete, all nodes are assigned a value that indicates what opinion the person at that node holds. In the next iterations of the model, we do the following to determine the effect of neighboring nodes:

    a. For each given node, we determine the proportion of neighboring nodes that agree with the proposition (y) and the proportion of nodes that disagree with the proposition ($1 - y$). For the node that it is paired with across the edge, we label its respective proportions as z and $(1 - z)$. Using these values as weights, we update the payoff matrix for the game played on a given edge in the respective iteration as follows:

		Person 2	
		Agree with *X*	Disagree with *X*
Person 1	Agree with *X*	A*y*, B*z*	-1, -1
	Disagree with *X*	-1, -1	C*(1 - y)*, D*(1 - z)*

The rationale behind the weights applied to the payoffs is that if a high proportion of nodes that a given node is connected to share a certain belief, this will have a greater impact on the probability with which the given node will select/change their opinion in the next iteration of the model. Thus, in a given iteration of the model, we update payoffs.

b. Using the same paradigm described above in step (3c), we determine a new boolean value for the node on whether a given node agrees or disagrees with the proposition.

5. The model can be run over several iterations. In our study, we run the model for 100 different iterations for different initial values of *A* and *B*, and *C* and *D*. At each iteration, we record the proportion of nodes that agree with the proposition and the proportion of nodes that disagree with the proposition.

Hypothesis: Upon a cursory glance, we believe that, if our model is run for a large number of iterations, no matter the initial conditions, there will be stable equilibria in which either all of the nodes will agree with the proposition or none of the nodes will agree with the proposition. Of course, we make sure that the experiment remains controlled by following the same exact process for each initial of *A, B, C,* and *D* used.

Method: We write files `Graph.java`, `Game.java`, and `Main.java` to run the iterations of our model. At each iteration, we print the true/false (agree/disagree) values of each node as well as the proportion of nodes that agree with the proposition. We use the following inputs for the model:

Erdős–Rényi graph input: *G(N, p) = G(1000, 0.5)*, 1000 nodes (people) and 0.5 probability of generating edge for each possible edge

We test our model on 7 different inputs of {*A, B, C, D*}:

1) {2, 2, 2, 2}
2) {8, 8, 2, 2}
3) {100, 100, 2, 2}
4) {200, 200, 2, 2}
5) {2, 2, 8, 8}
6) {2, 2, 100, 100}
7) {2, 2, 200, 200}

The rationale of these various inputs is to analyze the rate at which potential stable equilibria are reached for agree- and disagree-proportions in the graph. In the real-world, we believe that there could be stigmas associated with certain opinions that could lead to different payoff structures.

Data Set: Instead of using real-world data, we ran a simulation using our crowd network model and collected data by running our seven simulations, which each run several iterations of the model. We fed in a seed payoff matrix to start with, varying four parameters, as seen in the format below. As mentioned above, we set the payoff values for non-aligning opinions to be -1, -1:

Player 1, 2	Agree	Disagree
Agree	A, B	-1, -1
Disagree	-1, -1	C, D

Using these parameters, the model runs the games across the network several times and uses probabilistic models to simulate how each node, or person, acts in the next round of the game. This generates proportion-data at each iteration. This data can be seen in the CSV files provided as part of the project, where each row stores the agreement ratio (the proportion of the crowd that agrees with the given proposition) calculated at that iteration or round of the game. The CSV files also show the input parameters used to run the model. You can see the values used for the experiment in the data analysis section of this write up.

Results:
After running the simulations, we obtained the following data in a time-series format:

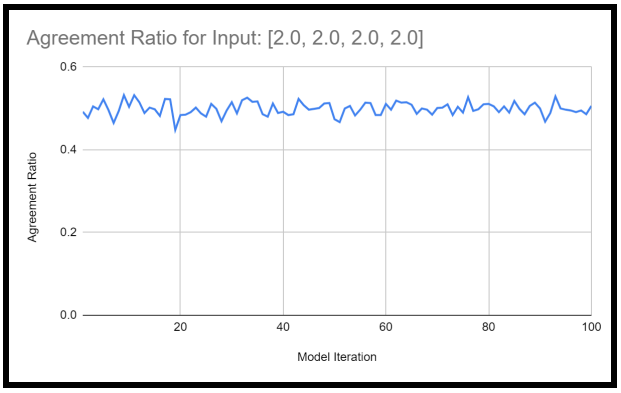
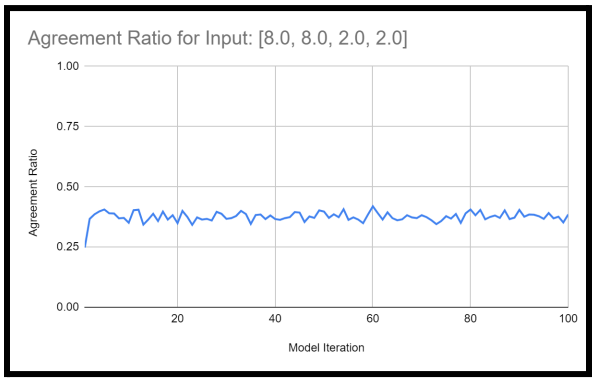
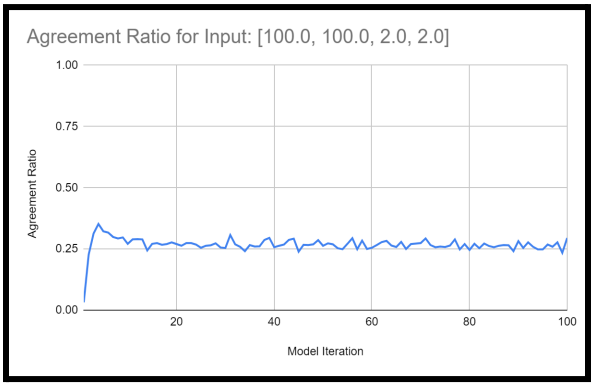


Fig 1. Agreement Ratio for Input: {2, 2, 2, 2}





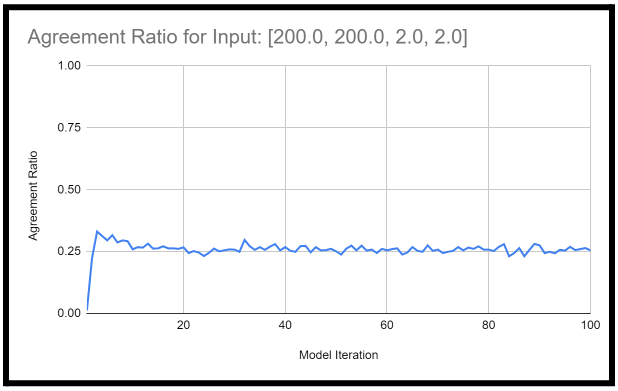


Fig. 2. Agreement Ratio for Inputs: {*A*, *B*, 2, 2}

Results (continued):

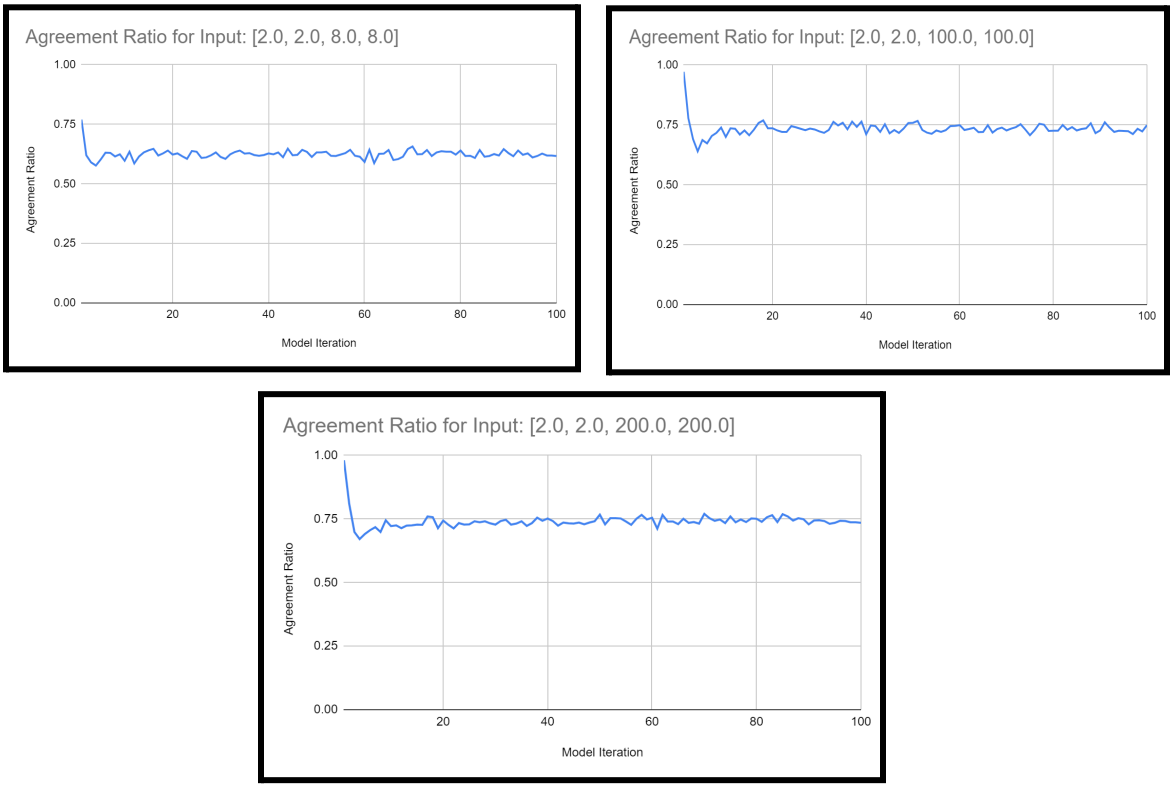


Fig. 3. Agreement Ratio for Inputs: {2, 2, *C*, *D*}

From our results, it appears clear that, over the course of several iterations of the model, a somewhat stable equilibrium is reached. However, we note that for each of our different inputs to the models, the stable equilibrium is not 0 or 1 for the agreement ratio– it is some other proportion. There appears to be some notion of stability in the formation of opinion based on our model generated to monitor and determine opinion dynamics. One thing to note is that for conditions in which we increased initial conditions for the "agree with proposition" payoff, the overall agreement determined at the end was far lower than the conditions in which we increased initial conditions for the "disagree with proposition" payoff. We discuss our interpretation of this result below. Also, another finding that we saw is that smaller *A, B, C,* and *D* lead to stable equilibria that were closer to 0.5. In fact, the input of {2, 2, 2, 2} was nearly at an equilibrium of 0.5.

Discussion and Conclusions:

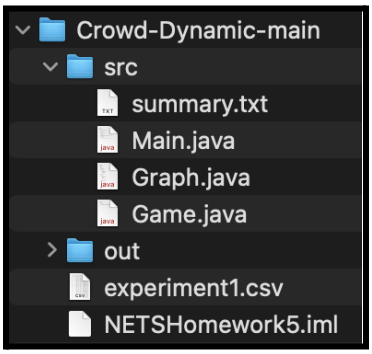
From the data collected from our simulation, it appears that certain specific stable equilibria are reached when inputting separate initial conditions in the model. Thus, our hypothesis correctly

predicted the existence of stable equilibria but did not predict the specificity of the values of these equilibria. From our results, it is clear that a stable agreement ratio develops, representing opinion formation similar to that of some real-world propositions that exist (those propositions that have staunch partisan support follow similar deadlock at a near-stable equilibrium). Thus, if we can determine specific payoff metrics regarding specific propositions, it may be possible to predict the proportion at which agreement/disagreement for a population may exist (with regard to that proposition). Of course, we realize that there are several other factors that exist that can affect opinion formation, but we argue that our model can serve as a simple representation of how social influence in opinion formation can lead to potential deadlock in agreement for a population.

With regard to drawbacks in our model, we wish to add an improved method for normalizing the payoffs in the payoff matrices after following our payoff-update process. This will shed true insight into a more precise value for the stable equilibria of the model given its initial conditions. We also hope to include more factors in the determination of the opinion that a node will form at a specific iteration– future work can involve survey inputs from real participants regarding how likely they are to maintain an opinion or how open they are to changing their opinion. In terms of model validation, we would like to compare our results with real-world data in the future in order to draw further conclusions on the accuracy of our model and what we should adjust. Finally, because we notice that stable equilibria are reached in the model, we would like to follow a method of proof in the future to show that stable equilibria will be reached as well as a mathematical model that will produce the expected value of the stable equilibria.

User Manual:

File Structure:



NOTE: Here, only experiment1.csv is shown, but you can find the rest of the experiments' CSVs in the project as well.

Files in SRC folder:

1. `Main.java`: This class contains the main runnable method to run multiple iterations of the simulation. The user can change the payoff matrix at the top. The four values $\{A, B, C, D\}$ represent the following values in payoff matrix format.
   a.

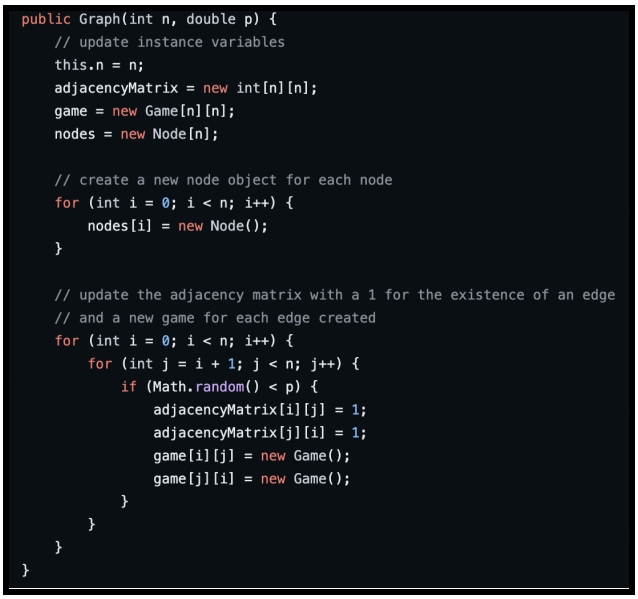
Player 1/2	Agree	Disagree
Agree	A, B	-1, -1
Disagree	-1, -1	C, D

   b. The user can also alter the number of edges and the probability of the presence of a possible edge in the graph (see screenshot below). In the for loop, the user can alter the number of iterations, which we set to 100. This method generates a randomized graph and runs multiple rounds of the games, eventually writing it to a CSV, which we used as our data for the analysis part of our experiment.
   c.

```java
public class Main {

    // Round 1 payoff matrix
    static double[] INITIAL_PAYOFF_MATRIX = new double[] {100, 100, 2, 2};

    /**
    * Runnable method to run the simulation.
    */
    public static void main(String[] args) {

        // nodes in graph
        int n = 1000;

        // probability of each edge existing
        double pVal = 0.5;

        // randomized graph creation
        Graph g = new Graph(n, pVal);
```

2. `Graph.java`: The graph class generates a random graph and has code to run multiple iterations of the games on each of the edges in the network. It updates the agreement sentiments and payoff matrices for each node and edge respectively based on the previous round.
   a. This file first randomizes a graph, based on the randomized Erdős–Rényi model, using a constructor that takes in two inputs: the number of nodes in the graph, and a probability value (between 0 and 1) that gives the probability that any potential edge between nodes in the graph exists. It stores an adjacency matrix to represent the graph, which stores a 1 at index [i][j] if that edge exists and 0 if it does not. If the edge does exist, a game object is created for that edge that represents an "edge" or game between the two "nodes" or people in the network.

i.



```java
public Graph(int n, double p) {
    // update instance variables
    this.n = n;
    adjacencyMatrix = new int[n][n];
    game = new Game[n][n];
    nodes = new Node[n];

    // create a new node object for each node
    for (int i = 0; i < n; i++) {
        nodes[i] = new Node();
    }

    // update the adjacency matrix with a 1 for the existence of an edge
    // and a new game for each edge created
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (Math.random() < p) {
                adjacencyMatrix[i][j] = 1;
                adjacencyMatrix[j][i] = 1;
                game[i][j] = new Game();
                game[j][i] = new Game();
            }
        }
    }
}
```

b. The iteration method shows what happens at each round. For each node we do the following: We look at the games played on all of its incident edges and we run our mixed strategy NE calculator on each game to calculate the "p" value, or the probability that the player agrees with the proposition. We take the average across all incident games to determine what the node will decide. Based on this average p value, we do a weighted randomization to determine if the node will agree/disagree in the next round. We also update the payoff matrix to reflect that the payoff is proportionally higher when the node is in alignment with the sentiments of its neighbors. This calculation can be seen in the code, explained in the comments.

c. The toString method helps the user see the agreement ratio values produced at each iteration to see how much of the crowd agrees with the proposition and how this changes over time.

d. The node class simply stores whether or not the person agrees with the given proposition.

3. `Game.java`: Game class to represent a game played on an edge between two nodes, storing the payoff matrix and the p and q values of the mixed strategy NE.
    a. The constructor pulls the initial payoff matrix from the main class to use for the first round of the game. The payoff matrices at each edge eventually evolve and reach a sort of equilibrium as more rounds are played.
    b. Calculates the mixed equilibrium NE based on the formulae given in class to find p and q. This assumes that the top right and bottom left both have unchanging payoffs of -1, -1. This method accesses the payoff matrix for the game at the time to find the a, b, c, and d values of the payoff matrix.

c.

```
void mixedEquilibria() {
    this.q = (this.payoffMatrix[2] + 1) / (this.payoffMatrix[0] + this.payoffMatrix[2] + 2);
    this.p = (this.payoffMatrix[3] + 1) / (this.payoffMatrix[1] + this.payoffMatrix[3] + 2);
}
```

4. Summary.txt: A summary of our project, which can be found at the top of this document as well.

CSV Files:

1) CSVs titled experimentX can be found outside of the SRC folder. Each one contains columns that specify the inputs for the experiment, including the number of nodes, the edge existence probability, and the initial payoff matrix. These have been included in our data analysis section for our writeup.