Introduction of the project :-

As we know on various social media networks the main aim of a user is to form connections which are beneficial to his/her own interests or to form as many connections as possible. We can represent the process of network formation on various social media platforms as outcomes of a game and can apply the concepts of game theory to analyze a particular situation to find the best possible outcome for a particular
user.
Through our work ,we have tried to find out the patterns of network formation and convergence of the network i,e. after a particular point there is no change in payoffs of the agent using the Game theory approach. Game theory provides mathematical models of strategic interactions among rational agents. We have used concepts of Bayes Nash Equilibrium and herd behavior assuming sequential entry of each agent to study the network formation.

Designing a Problem :-

The design problem is to represent the network formation and analyze its various situations/possibilities, which include cases of sequential entry, various equilibrium situations, herd behavior etc.

Theoretical Aspects That Were Covered :-

Payoff :-

Payoff is the term used to quantify the gain of a player. In our work we have calculated payoff by formulating utility functions.

Rational Behavior

Rational behavior is a decision making process in which an individual makes a decision to get optimum benefits.

Nash Equilibrium

Nash equilibrium is a set of strategies which no player wants to change knowing their payoffs ,as they will not gain anything by changing their strategy.

Bayes Nash Equilibrium

Bayes Nash equilibrium (BNE) is similar to Nash equilibrium but here players have incomplete information about other players i.e , they don’t know the exact payoff function of other players but will have some belief about the payoff function of other players.

Herd Behavior

Herd behavior is seen when everyone is doing what everyone else is doing even if their private information suggests them to do something else.

Sequential Entry and Perfect Equilibrium

Sequential Entry and Perfect Equilibrium :- We saw some examples of networks like star network, which corresponds to herd behavior, a network with linear segment and star. We learnt that network formation is an outcome of a sequential entry game in which the player enters in an exogenously given sequence , each agent has a type $\theta_i \in [0, 1]$, each agent may choose only one existing agent to connect and agents have the aim to attract connections. This can be considered in two settings: complete information and asymmetric information. In case of complete information we studied network formation by truncating the network and analyzing the utility function for entry of each agent.

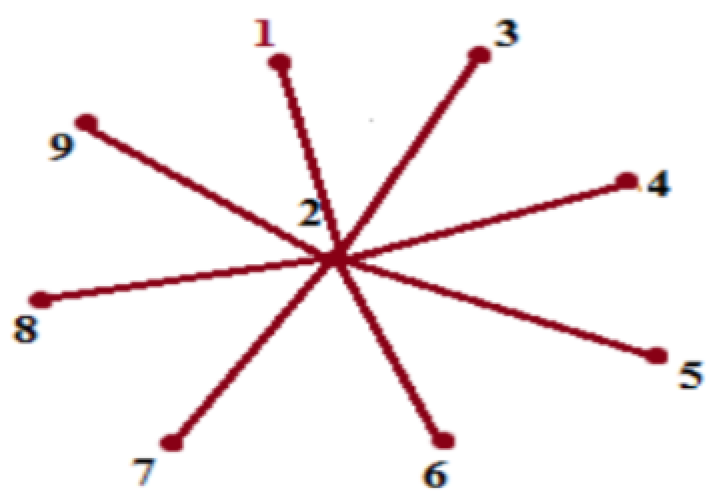
Game Theory

The branch of mathematics concerned with the analysis of strategies for dealing with competitive situations where the outcome of a participant's choice of action depends critically on the actions of other participants.

It is a Technique for modeling and analyzing strategic interactions between two or more agents/players.

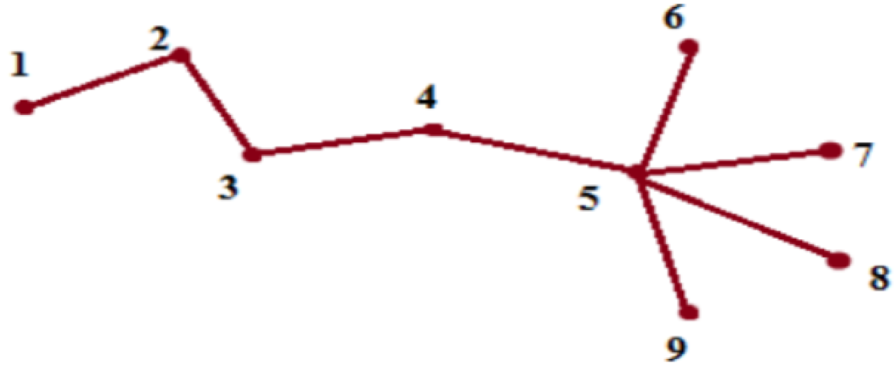
Different Types Of Networks :-

- **Star Network :-**



In the above network we can see herding behavior

- **Network with a Linear Segment and a Star :-**



Explanation of the Model Implemented :-

A] Calculating payoff for each player in all possible networks

A network is represented by an array of integers A. where i th agent is connecting with agent A[i].Agents enter the game in an exogenously given sequence(i.e. A[i] < i) Each agent has a type θi ∈ [0, 1]. α ∈ (0, 1) is an impatentient factor and δ ∈ (0, 1) is the utility received by i ∈ N from a direct link with an existing agent l < i with the highest type.

Utility function is given as :

$$U_i(x) = \Sigma_{j=1,2,3\ldots(i-1).}\delta^{|pij|}\ \theta_j + \Sigma_{k=(i+1),(i+2),\ldots}\alpha^{(k-1)-i}\ \delta^{|pij|}\theta_k$$

where i, j, k ∈ N; δ ∈ (0, 1) and pij is the shortest path connecting i and j.
Ui is the utility function to decide the payoff for agent "i" entering the network.
δ ∈ (0, 1) is the utility received by i ∈ N from a direct link with an existing agent j < i.
Pij is the shortest path between i and j.
Each agent has a type **θi** ∈ [0, 1] (Higher the value of θ more are the odds of the player to have many connections.)
α is the impatient factor. Which basically signifies that payoff from future players coming after a large time will not matter to the utility of the current agent.

- [ ] **Finding a distance matrix for given network :**

```
vector<vector<int>> distance_Matrix(vector<int>net){
```

```
    int n = net.size();
    vector<vector<int>>dm(n+1,vector<int>(n+1,0));
    for(int i=1;i<n+1;i++){
        for(int j=i+1;j<n+1;j++){
            dm[i][j] = dm[i][net[j-1]]+1;
            dm[j][i] = dm[i][j];
        }
    }
    return dm;

}
```

The above function finds a distance matrix for a given network. It takes the vector<int> as an input which is a network for which we are going to find the shortest path between every pair of agents.the distance matrix(dm) is a 2D vector where dm[i][j] represents the shortest distance between the agent i and j it will be same as dm[j][i] so basically this distance matrix will be symmetric matrix.

- ☐ **Calculating payoff for each agent in the network:**

```
vector<ld> calculate_Payoff(vector<vector<int>>dm,ld
delta,ld alpha,vector<ld>theta){
    int n = dm.size();
    vector<ld>payoff(n,0);
    for(int i=1;i<n;i++){
        for(int j=1;j<i;j++){
            payoff[i]+=(pow(delta,dm[i][j])*theta[j]);
        }
        for(int k=i+1;k<n;k++){

payoff[i]+=((pow(delta,dm[i][k]))*(theta[k])*(pow(alpha,k-i-
1)));
        }
    }
    return payoff;
```

```
}
```

Calculate_payoff function calculates the payoff for each agent in the network. It takes the distance matrix(dm) ,alpha,delta and theta as input and returns the long doble vector payoff where payoff[i] denotes the payoff of the agent i.

- ☐ **Generating all possible networks with given number of agents:**

```
void Networks(vector<int>&net,int n,ld delta,ld
alpha,vector<ld>theta){
    if(n==1){
        vector<vector<int>>dm;
        dm = distance_Matrix(net);
        vector<ld>payoff;
        payoff = calculate_Payoff(dm,delta,alpha,theta);

        for(auto x:net){
            cout<<x<<" ";
        }
        cout<<endl;
        for(int i=1;i<payoff.size();i++){
            cout<<setprecision(9)<<payoff[i]<<" ";
        }
        cout<<endl;
        cout<<endl;
        cout<<endl;

    }
    for(int i=1;i<n;i++){
        net[n-1] = i;
        Networks(net,n-1,delta,alpha,theta);
    }
```

```
}
```

For a given number of agents n this function will generate all possible networks.as we know the ith agent in the network can only connect to any j where j<i. So for n numbers for agents there are (n-1)! Possible networks. So this function will generate all these possible networks.a network is nothing but an array of integers A. where i th agent is connecting with agent A[i].

- **Finding a perfect equilibrium**

```
    ld mx_payoff = 0;
    int ind = 0;
    for(int i=2;i<net.size();i+=1){

        int z = net[i];
        for(int j=1;j<=i;j++){
            net[i] = j;
            vector<vector<int>>dm;
            dm = distance_Matrix(net);
            vector<ld>payoff;
            payoff = calculate_Payoff(dm,delta,alpha,theta);

            if(payoff[i+1]>mx_payoff){
                mx_payoff = payoff[i+1];
                ind = j;

            }



            for(auto x:net){
                cout<<x<<" ";
            }
            cout<<endl;
```

```
            for(int i=1;i<payoff.size();i++){
                cout<<payoff[i]<<" ";
            }
            cout<<endl;
            cout<<endl;

        }
        net[i] = ind;

        mx_payoff = 0;
        ind = 0;
    }
```

For a given network we will first calculate the payoff for each agent and then keeping all other players strategies constant we will try all possibilities for a specific agent and again will calculate the payoff for that agent and will choose to connect to the agent giving the max payout for that agent.and we will repeat this for every agent in network and atlast the modified network will be an equilibrium in which each agent can achieve the desired payout by not deviating from their initial strategy.

- **Checking convergence in sequential entry game :**

```
    vector<int>net = {0,1,2,3,4};
    for(int k=0;k<6;k++){
        ld mx_payoff = 0;
        int ind = 0;
        for(int i=2;i<net.size();i+=1){


            for(int j=1;j<=i;j++){
                net[i] = j;
                vector<vector<int>>dm;
                dm = distance_Matrix(net);
```

```
                vector<ld>payoff;
                payoff =
calculate_Payoff(dm,delta,alpha,theta);

                if(payoff[i+1]>mx_payoff){
                    mx_payoff = payoff[i+1];
                    ind = j;

                }

            }
            net[i] = ind;

            mx_payoff = 0;
            ind = 0;
        }

        cout<<"for N = "<<net.size()<<" equilibrium will be :
"<<"{";

        for(int l=0;l<net.size()-1;l++){
            cout<<net[l]<<",";
        }
        cout<<net[net.size()-1];
        cout<<"}";
        cout<<endl;
        net.push_back(1);

    }
```

Here we will be checking if any agent's optimal strategy is converging or not.initially we will start with a perfect equilibrium network and then a new agent

will enter in the sequential entry game and then we will find the new equilibrium which will give the optimal strategies for each agent.and we will observe equilibrium for n=4,5,6…..
And will check if any convergence is appearing in terms of the actions of agents as n increases.

How did we obtain the equilibrium?

We used the **One-Shot Deviation Principle**.
The one-shot deviation principle is the principle of optimality of dynamic programming applied to game theory. It says that a strategy profile of a finite extensive-form game is a subgame perfect equilibrium if and only if there exist no profitable one-shot deviations for each subgame and every player.
So we initially start with a set of strategies for the players and then check if the players are getting benefited from deviating to that strategy or not.

So, initially we start with the strategy {ϕ,1,1,1,1,1}.
As we know that the players enter sequentially, so player 1 and 2 have no choice but to connect among themselves.
Player 3 entered the game and has 2 choices, i.e., to connect with either player 1 or player 2.
Following are the payoff when player 3 connects with player 1 and with player 2 respectively.
Now after completing the model we simulated some cases for different values of α, δ and Θ, and compiled the results.

Simulation Results(for n = 6)

Case :- 1
α = 0.9 and δ = 0.9
A] The value of Θ for first player is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}

```
0 1 1 1 1 1
0 3.31703 2.59703 2.79459 2.9331 3.006 3.006


0 1 2 1 1 1
0 3.24413 2.67803 2.66903 2.8602 2.9331 2.9331
```

As we can clearly see that player 3 has higher payoff when it connects to player 1, therefore our initial network stands as it is.

Now we observe the payoff for player 4 from its connection with player 1,2 or 3.
0 1 1 1 1 1
0 3.31703 2.59703 2.79459 2.9331 3.006 3.006
0 1 1 2 1 1
0 3.25142 2.66993 2.72169 2.79369 2.9331 2.9331
0 1 1 3 1 1
0 3.25142 2.53142 2.87559 2.79369 2.9331 2.9331

As we can clearly see that player 4 has higher payoff when it connects to player 1, therefore our initial network stands as it is.
Similarly we performed the same analysis for player 5

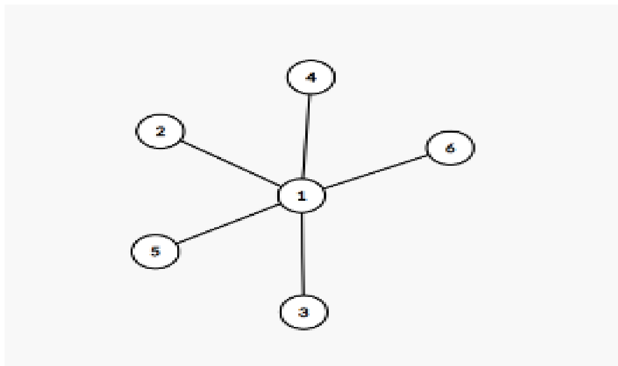
0 1 1 1 1 1
0 3.31703 2.59703 2.79459 2.9331 3.006 3.006
0 1 1 1 2 1
0 3.25798 2.66264 2.72898 2.8602 2.8593 2.9331
0 1 1 1 3 1
0 3.25798 2.53798 2.86749 2.8602 2.8593 2.9331
0 1 1 1 4 1
0 3.25798 2.53798 2.72898 3.0141 2.8593 2.9331

and player 6

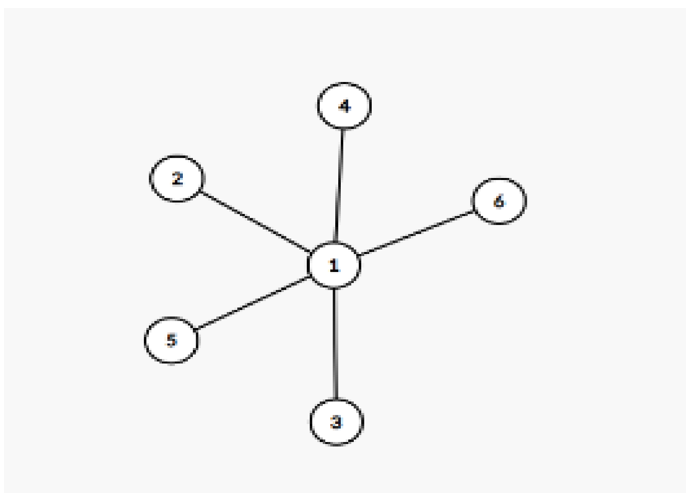
0 1 1 1 1 1
0 3.31703 2.59703 2.79459 2.9331 3.006 3.006
0 1 1 1 1 2
0 3.26389 2.65608 2.73554 2.86749 2.9331 2.8593
0 1 1 1 1 3
0 3.26389 2.54389 2.8602 2.86749 2.9331 2.8593

0 1 1 1 1 4
0 3.26389 2.54389 2.73554 3.006 2.9331 2.8593
0 1 1 1 1 5
0 3.26389 2.54389 2.73554 2.86749 3.087 2.8593

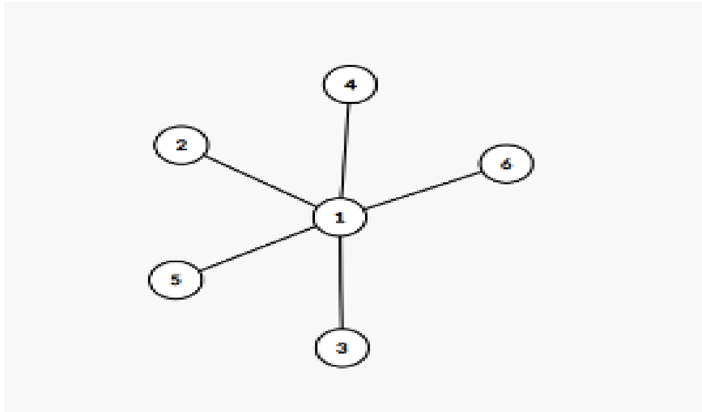
and obtained the equilibrium.



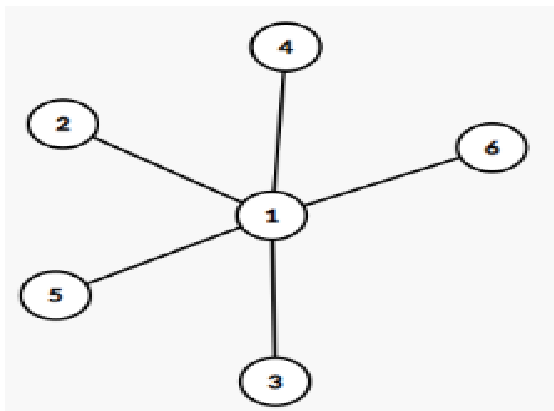
B] The value of Θ for first two players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ɸ,1,1,1,1,1}



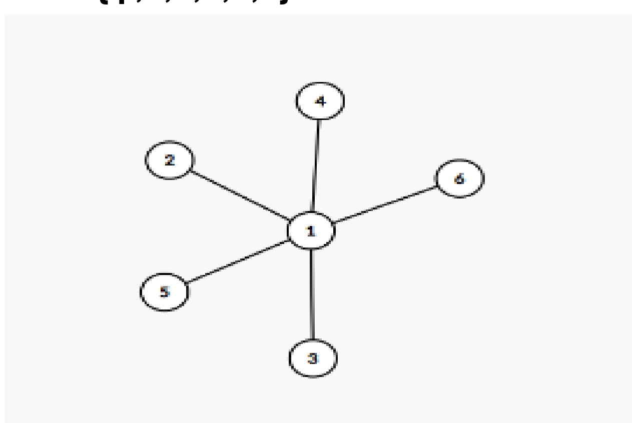
C] The value of Θ for first three players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ɸ,1,1,1,1,1}



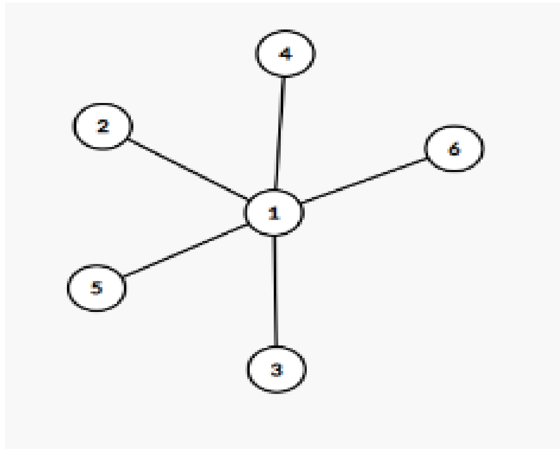
D] The value of Θ for first four players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}



E] The value of Θ for first five players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}



F] The value of Θ for first six players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}

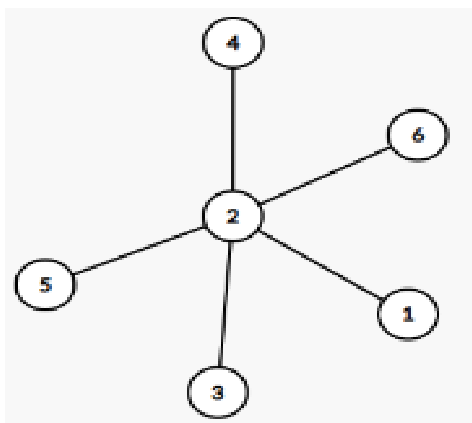


Case :- 2

α = 0.9 and δ = 0.001

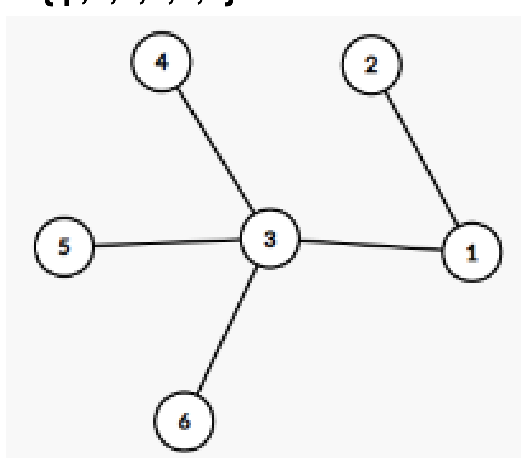
A] The value of ϴ for first player is 0.1 and 0.9 for rest of the players :-

Equilibrium obtained: - {ϕ,1,2,2,2,2}



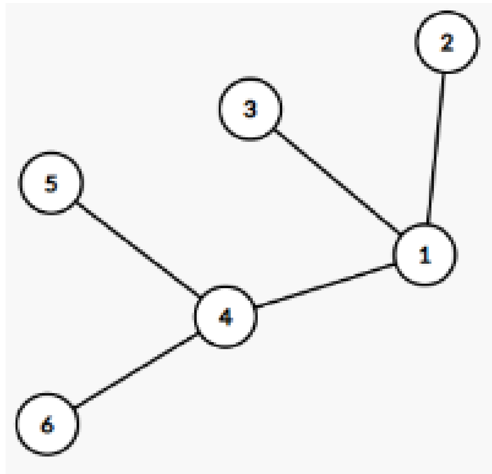
B] The value of ϴ for first two players is 0.1 and 0.9 for rest of the players :-

Equilibrium obtained: - {ϕ,1,1,3,3,3}



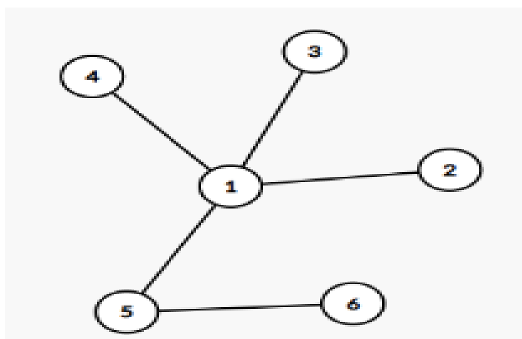
C] The value of ϴ for first three players is 0.1 and 0.9 for rest of the players :-

Equilibrium obtained: - {ϕ,1,1,1,4,4}



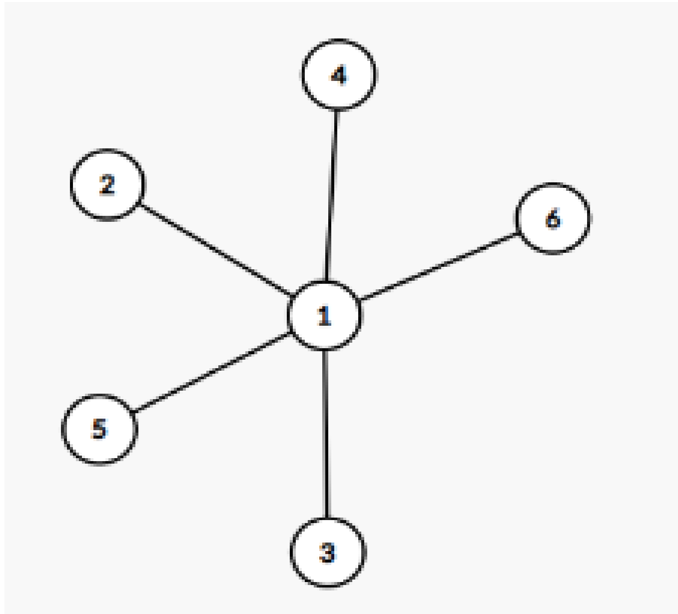
D] The value of ϴ for first four players is 0.1 and 0.9 for rest of the players :-

Equilibrium obtained: - {ϕ,1,1,1,1,5}

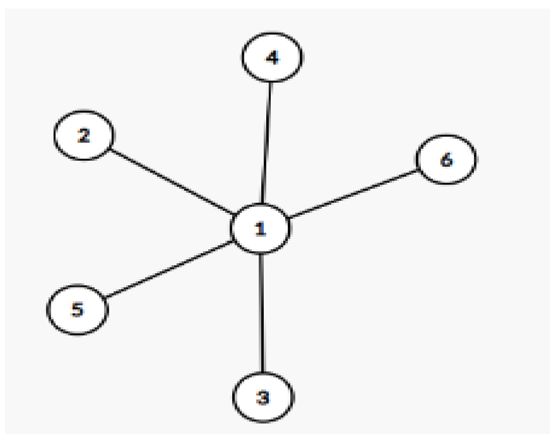


E] The value of ϴ for first five players is 0.1 and 0.9 for rest of the players :-

Equilibrium obtained: - {ϕ,1,1,1,1,1}



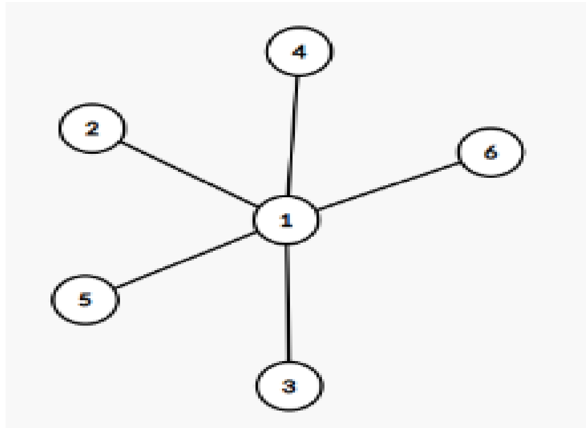
F] The value of ϴ for all six players is 0.1
Equilibrium obtained: - {ϕ,1,1,1,1,1}



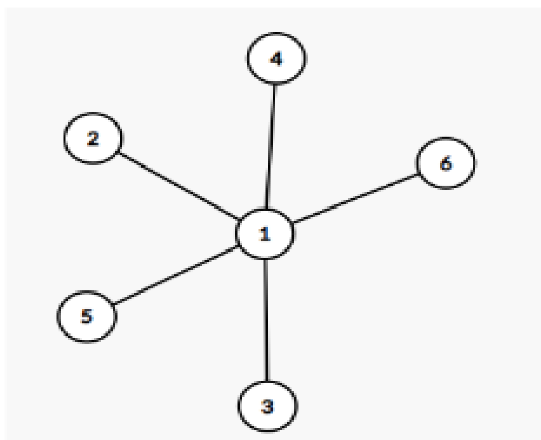
Case :- 3

α = 0.001 and δ = 0.9

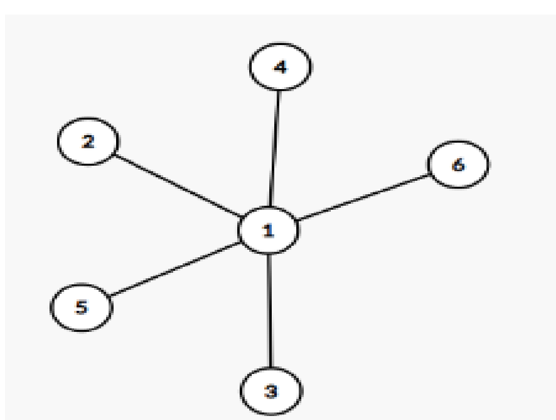
A] The value of ϴ for first player is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}



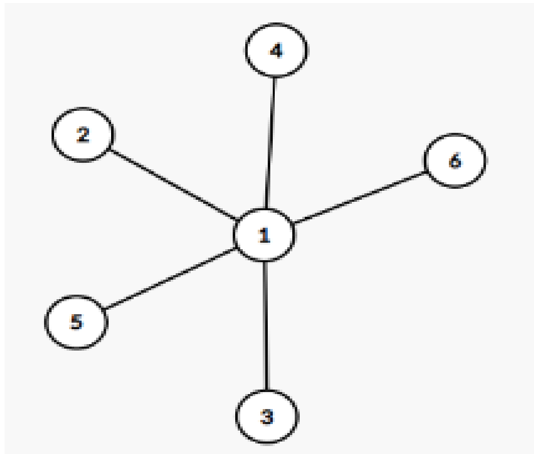
B] The value of Θ for first two players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}



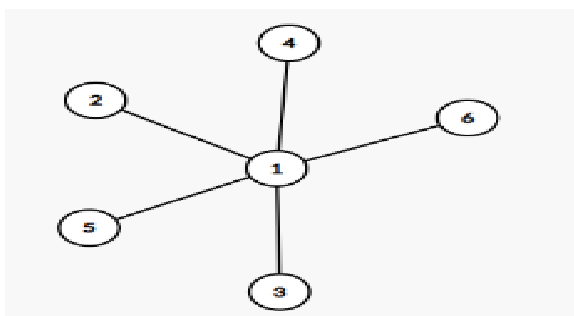
C] The value of Θ for first three players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}



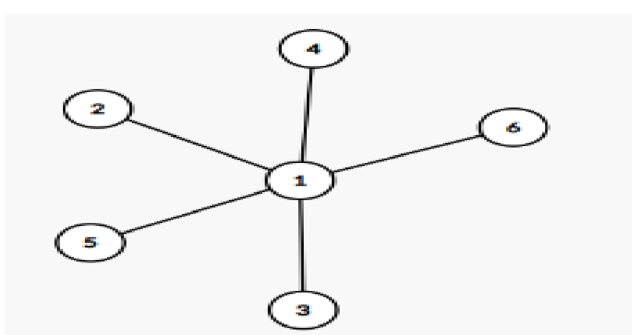
D] The value of Θ for first four players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}



E] The value of Θ for first five players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}



F] The value of Θ for all six players is 0.1
Equilibrium obtained: - {ϕ,1,1,1,1,1}

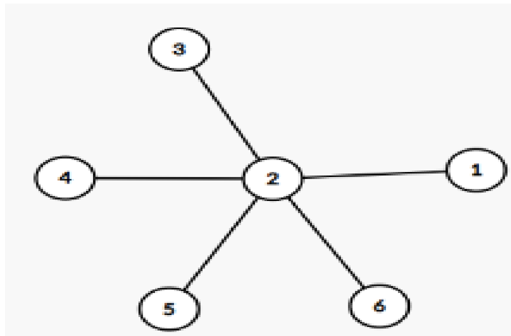


Case :- 4

α = 0.001 and δ = 0.001

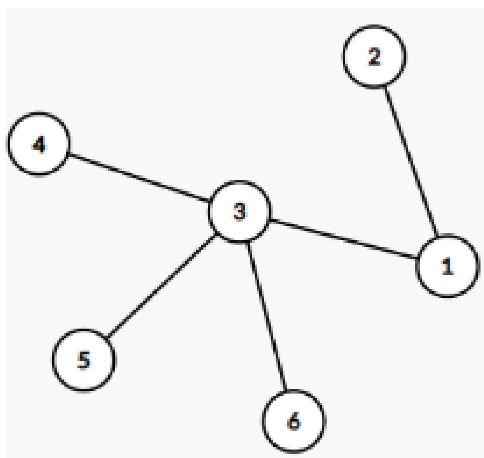
A] The value of Θ for first player is 0.1 and 0.9 for rest of the players :-

Equilibrium obtained: - {ϕ,1,2,2,2,2}



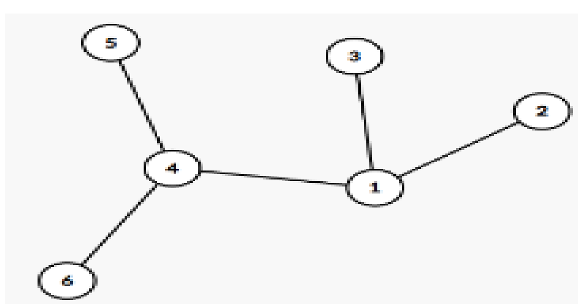
B] The value of Θ for first two players is 0.1 and 0.9 for rest of the players :-

Equilibrium obtained: - {ϕ,1,1,3,3,3}

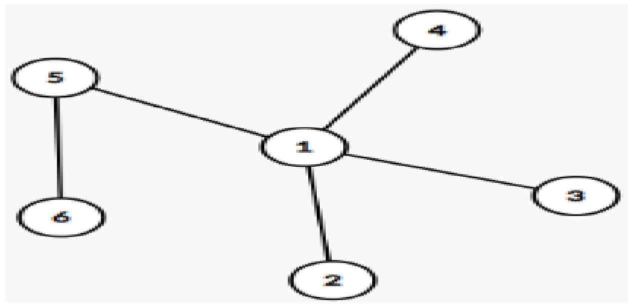


C] The value of Θ for first three players is 0.1 and 0.9 for rest of the players

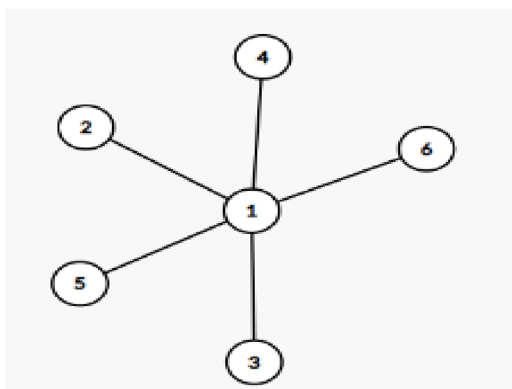
Equilibrium obtained: - {ϕ,1,1,1,4,4}



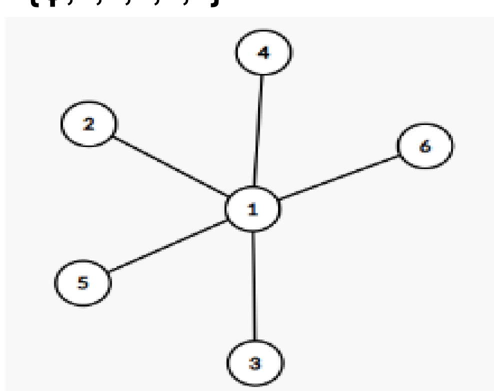
D] The value of Θ for first four players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,5}



E] The value of Θ for first five players is 0.1 and 0.9 for rest of the players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}



F] The value of Θ for all six players :-
Equilibrium obtained: - {ϕ,1,1,1,1,1}

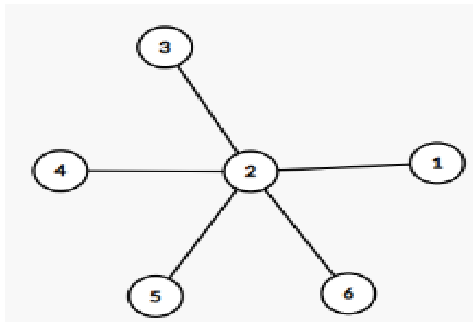


Case :- 5

α = 0.9 and δ = 0.9

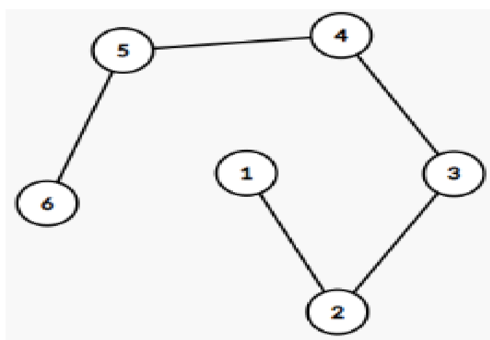
A] The value of Θ for player 2 is 0.9 and 0.1 for rest of the players.

Equilibrium obtained: - {ϕ,1,2,2,2,2}



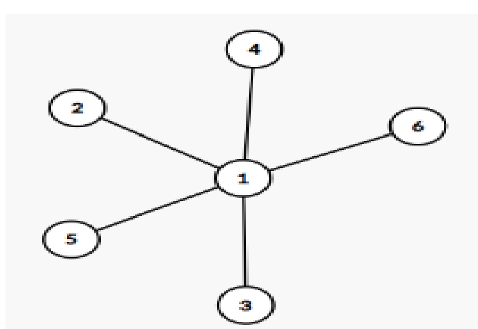
B] The value of Θ is as follows 0.1,0.3,0.5,0.7,0.9,0.9 for player 1,2,3,4,5,6 respectively

Equilibrium obtained: - {ϕ,1,2,3,4,5}



C] The value of Θ for all six players is 0.9

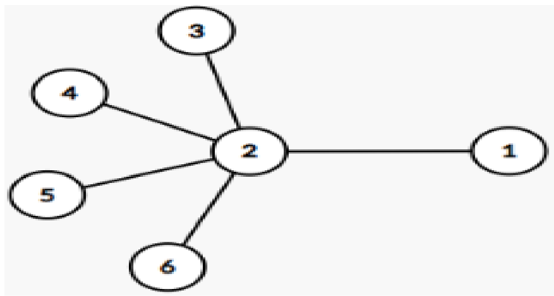
a] Equilibrium obtained: - {ϕ,1,1,1,1,1}



Player 2 always connects with 1, when 3 connects with 1 the best choice for player 4 would be connecting to 1 and same is for player 5 and 6.

b] When player 3 makes a mistake and connects with 2 :-

Equilibrium obtained: - {ϕ,1,2,2,2,2}



Player 2 always connects with 1, let say player 3 makes a mistake and decides to connect with player 1,now the best choice for player 4 would be to connect to player 2 and so on for other players.

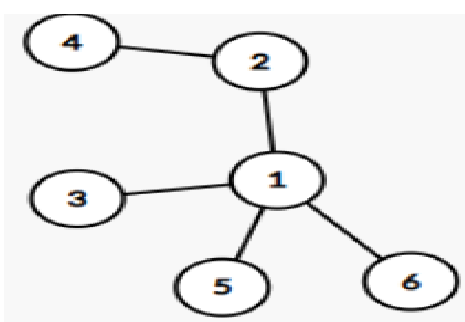
c] Player 4 commits a mistake.

When player 3 connects with player 1 while player 4 commits a mistake and connects to 2. In these cases 5 and 6 can either go to 1 or 2 it wont make a difference for any of the players as the payoff will be the same in either case.

1] Player 5 connects with player 1,

Equilibrium obtained: - {ϕ,1,1,2,1,1}

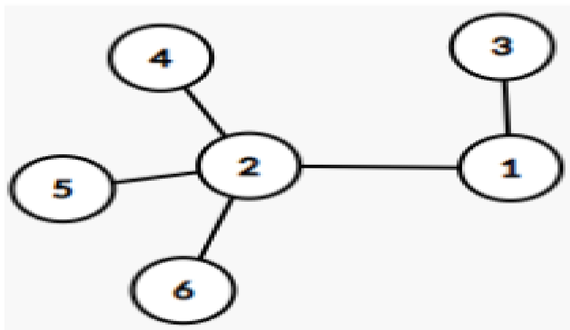
In this case player 6 connecting to 1 will be the equilibrium. And the network looks like



2] Player 5 connects with player 2,

Equilibrium obtained: - {ϕ,1,1,2,2,2}

In this case player 6 connecting to 2 will be the equilibrium. And the network looks like



Simulation result for analyzing convergence: -
$\alpha = 0.9$; $\delta = 0.9$;
θ_i = {0,0.1,0.1,0.9,0.9,0.9,0.1,0.9,0.1,0.9,0.1,0.1,0.1}

- **When n = 4**

```
0 1 1 3
1.40949 1.40949 0.981 0.9639

0 1 2 3
1.27754 1.5561 0.981 0.9639

0 1 1 1
1.4751 1.4751 0.9 0.9

0 1 1 2
1.40949 1.548 0.8271 0.8271

0 1 1 3
1.40949 1.40949 0.981 0.9639
```

As from the above results we can see that there are multiple equilibriums in it so let's consider any of them .
We are considering the {ϕ,1,1,3}

- **When n=5**

```
0 1 1 3 1
1.99998 1.99998 1.6371 1.62 1.5561

0 1 2 3 1
1.86803 2.14659 1.57149 1.55439 1.41759
0 1 1 1 1
2.06559 2.06559 1.5561 1.629 1.629

0 1 1 2 1
1.99998 2.13849 1.4832 1.4832 1.5561

0 1 1 3 1
1.99998 1.99998 1.6371 1.62 1.5561

0 1 1 1 1
2.06559 2.06559 1.5561 1.629 1.629

0 1 1 1 2
2.00654 2.1312 1.49049 1.5561 1.4832

0 1 1 1 3
2.00654 2.00654 1.629 1.5561 1.62

0 1 1 1 4
2.00654 2.00654 1.49049 1.71 1.62
```

In this case equilibrium will be {ϕ,1,1,1,1}

- **When n=6**

0 1 1 1 1 1
2.12464 2.12464 1.62171 1.7019 1.71 2.358

0 1 2 1 1 1
2.05174 2.20564 1.47664 1.629 1.6371 2.2851

0 1 1 1 1 1

2.12464 2.12464 1.62171 1.7019 1.71 2.358

0 1 1 2 1 1
2.05903 2.19754 1.54881 1.54881 1.6371 2.2851

0 1 1 3 1 1
2.05903 2.05903 1.70271 1.68561 1.6371 2.2851

0 1 1 1 1 1
2.12464 2.12464 1.62171 1.7019 1.71 2.358

0 1 1 1 2 1
2.06559 2.19025 1.5561 1.629 1.5561 2.2851

0 1 1 1 3 1
2.06559 2.06559 1.69461 1.629 1.6929 2.2851

0 1 1 1 4 1
2.06559 2.06559 1.5561 1.7829 1.6929 2.2851

0 1 1 1 1 1
2.12464 2.12464 1.62171 1.7019 1.71 2.358

0 1 1 1 1 2
2.11873 2.1312 1.61515 1.69461 1.7019 2.1393

0 1 1 1 1 3
2.11873 2.11873 1.629 1.69461 1.7019 2.2761

0 1 1 1 1 4
2.11873 2.11873 1.61515 1.71 1.7019 2.2761

0 1 1 1 1 5
2.11873 2.11873 1.61515 1.69461 1.719 2.2761

From the above observations the equilibrium will be at {ϕ,1,1,1,1,1}

Likewise we did for n=7,8,9….. And we have written code to automate this in which it will return the equilibrium for a given number of players the outputs is as follows:

```
for N = 4 equilibrium will be : {0,1,1,3}
for N = 5 equilibrium will be : {0,1,1,1,1}
for N = 6 equilibrium will be : {0,1,1,1,1,1}
for N = 7 equilibrium will be : {0,1,1,1,1,1,1}
for N = 8 equilibrium will be : {0,1,1,1,1,1,1,1}
for N = 9 equilibrium will be : {0,1,1,1,1,1,1,1,1}
```

As we can see from the output when N=4 player 4 wants to connect with player 3 but as the number of players in the game increases the player 4 changes his/her strategy by connecting to player 1 hence, we can say that convergence is happening in terms of players actions as the N increases .

Observations :-

- For case 1, **α** and **δ** are kept 0.9, we saw even after varying the values of theta we were getting the same network i.e; everyone connecting to agent 1. This is the herd behavior.
- In case 2 ,where **α >> δ** , we saw that agents were connecting to the agents having same theta value, like in case 2 , where theta for 1 and 2 were same and theta for 3,4,5,6 were same we saw , 2 connected with 1 but 4,5,6 connected with 3. Similar patterns were seen for other cases as well.
- In case 3 ,where **α << δ** , we saw similar things happening as in case 1.
- In case 4, where **α** and **δ** are small (0.0001), we saw similar things happening as in case 2.
- In case 5, **α** and **δ** are kept 0.9, now we see the coming agent connects with the agent having highest theta value, as a result we saw a linear network when theta values were increasing.

Findings

- Payoffs are majorly influenced by **δ** values, it is seen by the fact that **δ** values were the same for the cases 1 and 3, and 2 and 4, and so are the observations.
- After **δ,** theta values are the important factor to define the structure of the network.

1. **Concluding Remarks:** Many major concepts like game theory, Nash equilibrium, Bayes Nash equilibrium, herd behavior etc. were much more cleared while working on this project.
   Some concepts of c++ and data structures were also covered during this project. Hence working on this project helped in understanding many more new concepts and their daily life applications.

2. **References:**

1. “A Course in Game Theory” by Martin J. Osborne and Ariel Rubinstein (ISBN 0-262-65040-1);
2. Game theory / Michael Maschler
3. Modeling bounded rationality / Ariel Rubinstein.
4. A Simple Model of Herd Behavior Author(s): Abhijit V. Banerjee Source: The Quarterly Journal of Economics, Vol. 107, No. 3, (Aug., 1992), pp. 797-817
5. Graph Editor: https://csacademy.com/app/graph_editor/