

Simulating network formation using Game theory approach

Objective

In this project ,we have tried to represent the network formation and analyze its various situations/possibilities , which include cases of sequential entry, various equilibrium situations and herd behavior.

Theoretical Aspects That Were Covered :-

Sequential Entry

A sequential entry game is 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.

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.

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.

Model

- 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 $\theta_i \in [0, 1]$. $\alpha \in (0, 1)$ is an impatient factor and $\delta \in (0, 1)$ is the utility received by $i \in N$ from a direct link with an existing agent $l < i$ with the highest type.

Utility Function is given by,

.

$$U_i(x) = \sum_{j=1,2,3 \dots (i-1)} \delta^{|p_{ij}|} \theta_j + \sum_{k=(i+1),(i+2), \dots} \alpha^{(k-1)-i} \delta^{|p_{ik}|} \theta_k$$

where $i, j, k \in N$; $\delta \in (0, 1)$ and p_{ij} is the shortest path connecting i and j .

Model

- 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;
}
```

Model

- 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;
}
```

Model

- 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);
    }
}
```

Model

- 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;

}

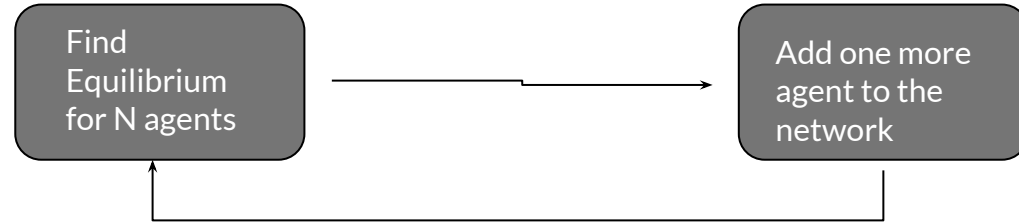
net[i] = ind;

mx_payoff = 0;
ind = 0;

}
```

Model

- Checking convergence in sequential entry game :

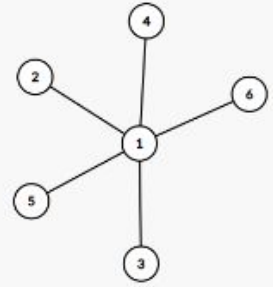


Observations

When α and δ are kept 0.9

- Number of agents taken is 6
- θ Values are take as 0.1 for 1 and 0.9 for others .
- Then 0.1 for 1 and 2 and 0.9 for rest .
- This was done till all θ s become 0.1.
- In all these cases Network formed was

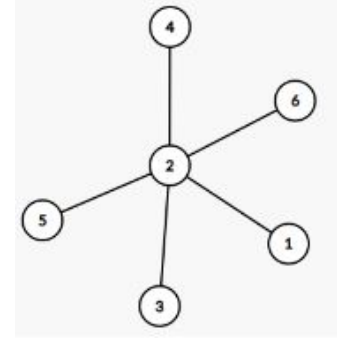
$\{\phi, 1, 1, 1, 1, 1\}$



When α is 0.9 and δ is 0.001

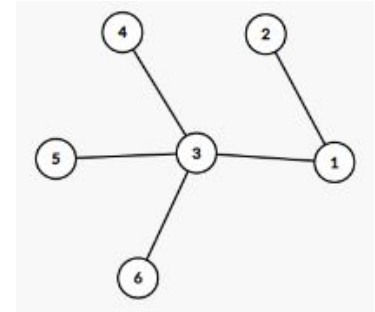
- θ Values are take as 0.1 for 1 and 0.9 for others .

Equilibrium obtained: - $\{\phi, 1, 2, 2, 2, 2\}$



- The value of θ for first two players is 0.1 and 0.9 for rest of the players

Equilibrium obtained: - $\{\phi, 1, 1, 3, 3, 3\}$

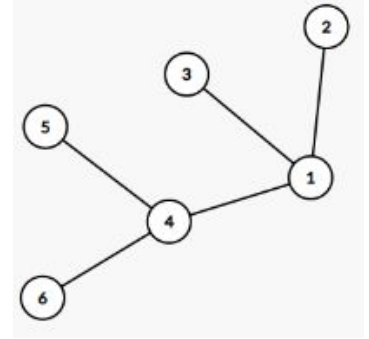


Observations

Previous case continued.....

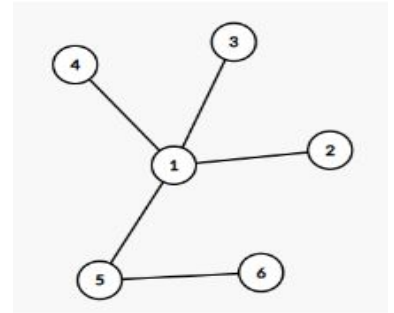
- The value of Θ for first three players is 0.1 and 0.9 for rest of the players

Equilibrium obtained: - $\{\phi, 1, 1, 1, 4, 4\}$



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

Equilibrium obtained: - $\{\phi, 1, 1, 1, 1, 5\}$

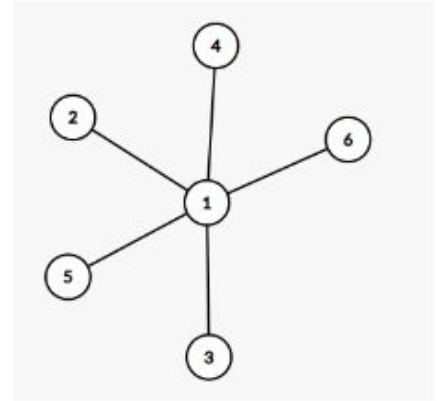


Observations

Previous case continued.....

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

Equilibrium obtained: - $\{\phi, 1, 1, 1, 1, 1\}$



- The value of Θ for all six players is 0.1

Equilibrium obtained: - $\{\phi, 1, 1, 1, 1, 1\}$

- We saw that agents were connecting to the agents having same Θ value, like in case 2, where Θ for 1 and 2 were same and Θ 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.

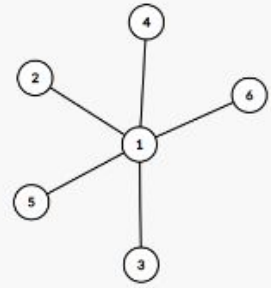
Observations

Observations

When α is 0.001 and δ is 0.9

- θ Values are take as 0.1 for 1 and 0.9 for others .
- Then 0.1 for 1 and 2 and 0.9 for rest .
- This was done till all θ s become 0.1.
- In all these cases Network formed was

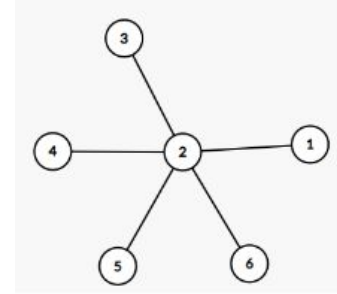
$\{\phi, 1, 1, 1, 1, 1\}$



When α and δ are kept 0.9

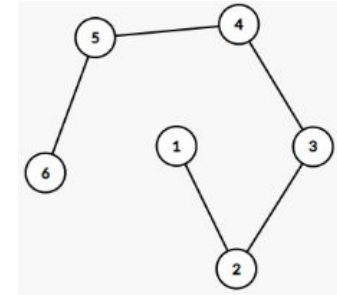
- The value of Θ for player 2 is 0.9 and 0.1 for rest of the players.

Equilibrium obtained: - $\{\phi, 1, 2, 2, 2, 2\}$



- 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: - $\{\phi, 1, 2, 3, 4, 5\}$



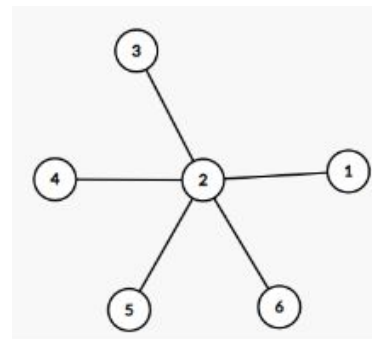
Observations

Observations

When α and δ are kept 0.9 also all Θ s are 0.9 but agents change their strategy

- When player 3 makes a mistake and connects with 2

Equilibrium obtained: - $\{\phi, 1, 2, 2, 2, 2\}$



- 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.

Previous case continued.....

Player 4 commits a mistake.

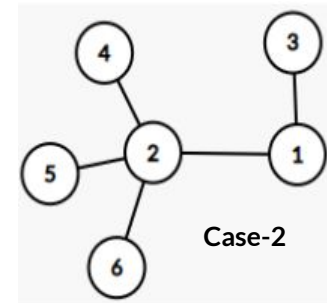
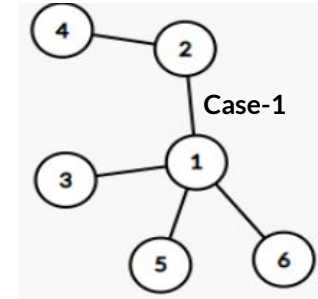
- When player 3 connects with player 1 while player 4 commits a mistake and connects to 2. In these cases 5 and 6 can either go to 1 or 2 it won't make a difference for any of the players as the payoff will be the same in either case. So we will have equilibrium networks.

1. **Player 5 connects with player 1,**

Equilibrium obtained: - $\{\phi, 1, 1, 2, 1, 1\}$

2. **Player 5 connects with player 2,**

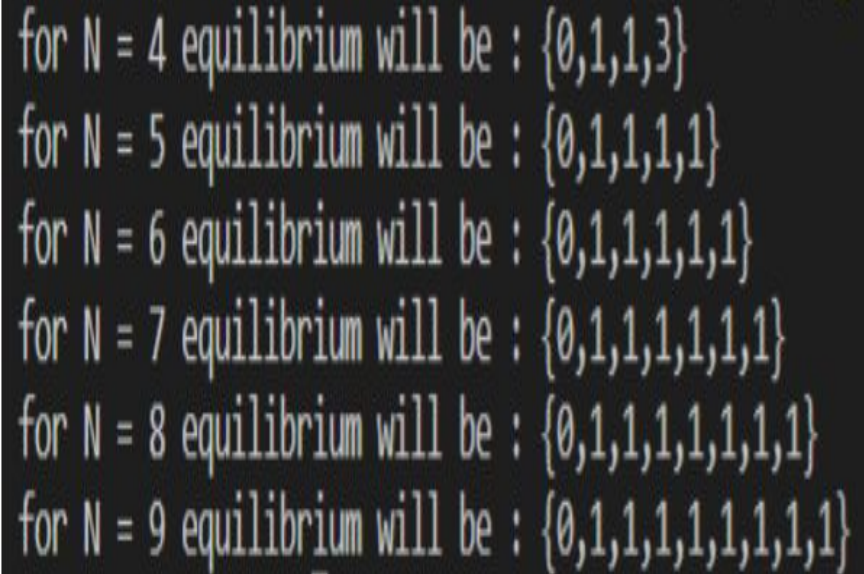
Equilibrium obtained: - $\{\phi, 1, 1, 2, 2, 2\}$



Observations

Simulation result for analyzing convergence for 4th player: -

- $\alpha = 0.9$; $\delta = 0.9$;
- For $n=4$ considering one of the equilibrium $\{\phi, 1, 1, 3\}$.
- On increasing n the equilibrium for the above network shifts to $\{\phi, 1, 1, 1, 1\}$
- On further increasing n the equilibrium for the 4th player converges to the player 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 .



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 : $\{0, 1, 1, 1, 1, 1, 1, 1, 1\}$

Conclusion



- 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
-