Code of finding 1

[**Finding 1**]. The proposed algorithm is effective, which performs better than the selected benchmarks such as community division method, random division method as well as a random guess.

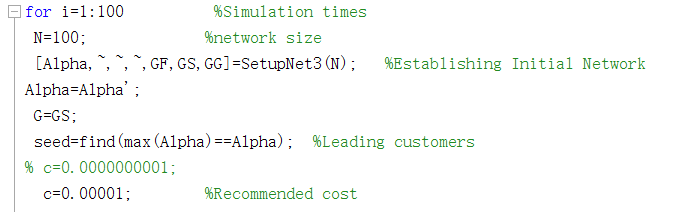
**Note: This code compares our algorithm with community division method and random division method.**

一、Main Instructions for Use：

**main.m** is the main program of simulation, which can be started by running directly. After **main.m**, you can get the result of our algorithm. Then keep the parameters and run **mainSubnets3.m** directly. You can get the results of community division method and random division method

**1.1** **Parameter definition**

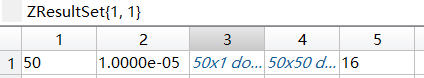
In the **main.m**, you can set initial values for the **times of simulations**, **cost of recommendation**, **leading customers**, and **market size**.



**1.2** **Result review：**

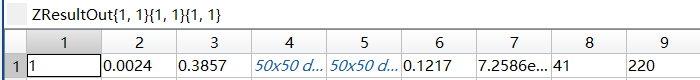
Note that all results are stored in **cell** form for easy reading.

**ZResultSet** stores parameter settings for each simulation



**ZResultSet{1,** simulation times**}** from left to right are **market size**, **cost of recommendation**, **the preferences of each customer, initial network** and **leading customers.**

. **ZResultOut** stores the result for each simulation。Each simulation has nine results, R1, R2, R3 are three different awarding strategies, and each recommendation has three different accepting strategies A1, A2, A3.



**ZResultOut{1,** simulation times **}{1,** awarding strategies **}{1,** accepting strategies **}**

From left to right, the order is **simulation number,** **optimal reward**, **optimal price**, **recommendation network** (double matrix format), **action network** (double matrix format), **total net profit**, **total rewarding to consumers**, **number of participants**, **total number of recommended actions**.

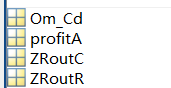
**1.3 Comparison with the community division method and random division method**

**mainSubnets3.m** is designed to compare the algorithm in the paper, community division method and random division method.

**mainSubnets3.m** can only be executed after running **main.m**

After executing **mainSubnets3.m**, each simulation profit calculated by the algorithm in this paper will be stored in the variable **profitA**, the profit of community division method will be stored in **ZRoutC**, and the profit of random division method will be stored in **ZRoutR**.

In particular, **Om\_Cd** stores the results of 



二、Procedure contents：

|  |  |  |
| --- | --- | --- |
| **Main script** | | |
| **main.m** | | The main program, starts the simulation when it runs. |
| **mainSubnets3.m** | | compare the algorithm in the paper, community division method and random division method. |
| **Function** | | |
| **Basic function** | **Sub function** | **contents** |
| **getsubGC.m** |  | Code of community division method |
| **getsubGR.m** |  | Code of random division method |
| **SetupNet3.m** |  | Generate Initial Networks |
|  | **FreeScaleBA.m** | Generate scale-free networks |
|  | **randnet.m** | Generate stochastic networks |
| **functionrs1.m** |  | Calculate the initial price and reward in case of R1 |
|  | **GiveSs.m** | Generate action networks based on a given initial network |
|  | **Givers.m** | Generate recommendation networks based on a given action networks and an accepting strategy |
|  | **Derelation.m** | Recursively delete the recommendation action of a node and the recommendation behavior derived from that recommendation action. |
|  | **dedges.m** | Delete an edge from the network and its recommendation order. |
|  | **findpath.m** | Determine whether the recommendation network is formed from the leading customer. |
| **functionrs12.m** |  | Iteratively solves the optimal initial price and optimal reward for R1 with a given initial price and reward |
|  | **functionrs12\_find.m** | Recursive function based on functionrs12.m |
|  | **TheStepOners1.m** | Solve R1 initial price and reward |
|  | **TheStepTwors1.m** | Update R1's action network and recommendation network，considering different accepting strategies. |
|  | **maxprofit1.m** | Calculate the optimal profit of R1 |
| **functionrs2.m** |  | Iteratively solves the optimal initial price and optimal reward for R2 with a given initial price and reward |
|  | **functionrs2\_find.m** | Recursive function based on functionrs2.m |
|  | **TheStepTwors2.m** | Update R2's action network and recommendation network，considering different accepting strategies. |
|  | **maxprofit2.m** | Calculate the optimal profit of R2 |
| **functionrs3.m** |  | Iteratively solves the optimal initial price and optimal reward for R3 with a given initial price and reward. |
|  | **functionrs3\_find.m** | Recursive function based on functionrs3.m |
|  | **TheStepTwors3.m** | Update R3's action network and recommendation network，considering different accepting strategies. |
|  | **maxprofit3.m** | Calculate the optimal profit of R3 |
|  | **nextrs.m** | Calculate the benefits of an edge in case of R3 |
|  | **nextrs2.m** | Calculate the benefits of a node in R3 |

Note: The basic function is called directly by the main script, and the sub function is called in the basic function or by itself.