

## Report for Project 3:

### ISE and IMP problem

#### **Preliminaries**

Describe any algorithm, software or codes that is used in your project.

#### 1.1 Problem Description

##### 1.1.1: ISE problem Description

The ISE issue is an important part of the IMP problem, and it has been taken out as a part to practice, so it is necessary to describe it separately.

The basic model of the ISE problem is that given a propagation rule, or model, is initialized into a directed graph with a seed set, trying to get the final number of active nodes. The whole problem is simulating the effect of the seed set on the directed graph. Estimation of the range of force spread.

The ISE problem is based on two basic propagation models, a linear threshold model (IC) and an independent cascade model (LT). The basis of the model is to abstract the social network into a directed graph, with nodes representing people, edges representing influencing relationships, and nodes consisting of two State. An active node can change other nodes to become active nodes by affecting others. The reaction is implemented by round rather than individual activation. The input to the problem is a directed graph, a seed set, and the output is the activated node. Quantity.

In the case of the linear threshold model (IC), the impact threshold of each node is initially random. After the seed set is given, whether the weight is greater than the node that has not been activated to determine whether to activate or not, the weight of several nodes is determined. Will be superimposed to affect a node until no nodes can be activated.

In the independent cascading model, each node will have an activation success rate. The seed set arguments attempt to activate each neighbor node. The activation node of the previous round in each round will not be activated, only the new active node will go. Activate other nodes, but each time the activation is unsuccessful, the weight of the node will be superimposed. Until there is no new node.

There are many similarities between the linear threshold model and the independent cascade model. The input and output are the same. The only difference is that they are handled differently. In the linear threshold model, nodes can always activate other nodes, but the node pairs in the independent cascade model. A

neighbor node has only one chance to activate (the weights will be superimposed).

#### 1.1.2: IMP problem description

A very important point in the IMP problem is the ISE problem, which is to estimate the scope of influence transmission. The IMP problem itself is closely related to it, so it is necessary to separate the two transactions.

The IMP problem is to select an influence propagation model (as in ISE, there are only two - independent cascade models (LT) or linear threshold models (IC)), select a predetermined scale (such as 10), give a Directed graphs, the result of trying to get is a set, the set as a seed set can make the impact spread the most.

This problem is so closely related to the ISE problem that it is obvious that the evaluation function of this problem is the modeling result of the ISE problem, which means that completing the ISE problem is the prerequisite for completing the IMP problem.

#### 1.1.3 Problem Description Conclusion

In summary, the problem is based on a directed graph. There are two propagation models. The purpose of the IMP problem is to obtain a limited set (within a limited time) so that the spread range is maximized on the directed graph. In solving the IMP problem, it is necessary to solve the ISE problem, that is, to estimate the range of propagation on the directed graph. The IMP problem is an NP-hard problem.

### 1.2 Problem Applications

#### 1.2.1: ISE problem's Application

First of all, the ISE problem is an important part of the IMP problem. The IMP problem must be used to solve the ISE problem before the evaluation function can be carried out, but in addition he has his own independent use.

Obviously, if we have selected some basic nodes, the study of ISE problems can help us estimate the possible diffusion results. Such an algorithm can obviously help marketers to estimate the results of marketing, after the release of the advertisement. The conversion rate, whether public opinion reaches the threshold, whether the individual's harmful speech will spread on a large scale, etc., can play an important role in estimating human behavior, especially in today's large-scale application of social networks. It is necessary to pay attention to the algorithm

#### 1.2.2 IMP problem's Application

The IMP question focuses on how to extract the top N individuals from the directed graph, so that the number of nodes ultimately affected is the highest after the interaction is completed. If the ISE problem focuses on the "estimate" result, then solving the IMP problem can help us positively. Going to "manufacturing" results.

Obviously, marketing, advertising, social mainstream media propaganda, secretly leading the direction of public opinion, etc., have their own limitations, marketing and advertising can only have limited funds to choose the propagator. Media propaganda and secret Public opinion must control the initial source to prevent negative effects. At this time, we need to use the results of the IMP problem to help us pick the most influential nodes to achieve results. Especially in the large-scale application of social networks today. Acquisition becomes a breeze, which makes it even more necessary to estimate the results in order to achieve the goal of "how fast, better, and strive for the upper reaches"

### 1.2.3 Problem Description Conclusion

In summary, the resolution of the ISE problem helps us to estimate the situation in the social network, and thus better allocate resources, more inclined to estimate the existing situation; the resolution of the IMP problem helps us to allocate more rationally. Resources to achieve the purpose of maximizing impact communication, more inclined to take the initiative to solve the problem.

## 2. **Methodology**

Describe the details of your representation/algorithm/architecture, etc.

### 2.1: ISE part

#### 2.1.1 Notation

Graph: the directed graph used

Seed: the seed collection used

Node: the node in the directed graph

Conn (Node): A collection of neighbors of a node in a directed graph

Random (): random number, generated by NumPy

#### 2.1.2: Data structure

##### 2.1.2.1 Linear Threshold Model

Had\_Acti: set (), a collection with no repeating elements inside.

Next\_seed: set (), a collection used internally by a loop with no repeating elements

inside.

Node: Node, which has its own neighbors and its own value.

Graph: A directed graph composed of nodes.

Seed: a list, since there must be no repeating elements inside, it can be regarded as set ()

Conn (Node), the neighboring node of Node, is a collection, and the elements are nodes.

#### 2.1.2.2 Independent cascading model

Had\_Acti: set (), a collection with no repeating elements inside.

Addition: set (), collection, no repeating elements inside

Node: Node, which has its own neighbors and its own value.

Graph: A directed graph composed of nodes.

Seed: a list, since there must be no repeating elements inside, it can be regarded as set ()

Conn (Node), the neighboring node of Node, is a collection, and the elements are nodes.

Value\_Array: a random array whose length is determined at initialization time

Thresh\_array: an array of all 0s, the length is determined at initialization time.

Next\_add: set (), collection, no repeating elements.

#### 2.1.3 Model design.

For the ISE problem, the input and output of the problem have been defined. First, the input and output formats are normalized, and the data structure is created for the format. Then the problem is solved. Since the model has been established, the main difficulty in solving the problem lies in Write efficient code instead of implementing it.

#### 2.1.4

This problem will involve two algorithms, one is the simulation of the linear threshold model, and the other is the simulation of the independent cascade model. The simulation of the two models, the input is Graph and seed, the output is an integer. The function is the final size of the seed spread in the graph under the model.

2.1.4.1 For the simulation of the linear threshold model, the pseudo code of the algorithm is as follows:

Graph, seed is a known condition

Initialize Had\_Acti to set ()

While seed not empty:

    Initialize next\_seed to set ()

    Add all nodes in the seed to Had\_Acti

    For all nodes in the seed I:

        For all nodes in Conn(i):

            If Random () < node weight and node is not in Had\_Acti

                This node joins Next\_seed

            End

        End

    End

End

Seed = next\_seed

End

Add all nodes in the seed to Had\_Acti

Returns the length of Had\_Acti

2.1.4.2 For the independent cascade model:

Graph, seed is a known condition

Initialize Had\_Acti

Initialize Addition

Add all nodes in the seed to Had\_Acti and Addition.

Randomly generated random seed

Initialize value\_array with the number of nodes in the Graph as the length

Initialize Thresh\_Array with the number of nodes in the Graph as the length

    For each node in the Graph, if its corresponding Value\_array value is zero:

        Add it to Had\_Acti and Addition.

While addition is not empty:

    For each node in addition i:

        For each j in Conn (i):

            The j position in Thresh\_array plus the weight equal to j

        End

    End

Initialize next\_add

    For each node in addition i:

```

For each j in Conn(i):
    If the value corresponding to j in value_array is smaller than the value
        corresponding to j in the thresh_array
        Next_add joins j
    End
End
End
End
For all nodes in next_add:
    If it is not in had_acti, add it to the addition
    Add it to had_acti
Returns the length of had_acti.

```

## 2.2 IMP part.

### 2.2.1 Notation

ISE (): reads in graph, seed, and model type, returns a value

Node: Node, which has its own neighbors and its own value.

Graph: A directed graph composed of nodes.

Seed: a list, since there must be no repeating elements inside, it can be regarded as set ()

Conn (Node), the neighboring node of Node, is a collection, and the elements are nodes.

### 2.2.2 Data Structure

Node: Node, which has its own neighbors and its own value.

Graph: A directed graph composed of nodes.

Seed: a list, since there must be no repeating elements inside, it can be regarded as set ()

Conn (Node), the neighboring node of Node, is a collection, and the elements are nodes.

List: Priority queue. If the element is a tuple, it is sorted from big to small according to its first element.

New\_list: Priority queue. If the element is a tuple, it is sorted from big to small according to its first element.

### 2.2.3: Model design

The problem does not need to consume too much analysis, because the framework

of the problem has been given, the input and output have been defined. Solving the problem is very energy-consuming, the initial idea is to use dynamic programming or greed to solve the problem, but the problem is the NP-hard problem, there is no general solution within the polynomial time. All of them know how to use greedy algorithms or some heuristic algorithms by reading the paper. Considering the time factor, the greedy algorithm is used.

#### 2.2.4 Detail of algorithms

The core idea is to add a node each time, so that the amount of influence propagation is the largest each time.

However, the content added directly if the increment is greater than the last largest increment is added to the process. This is a sub-model based on influence magnification.

Initialize seedset ()

Initialize list

Initialize before\_seed\_count = 0

In each node in the Graph, pick the node with the largest increment as the initial node and join seedset ().

Loop until the size of the seedset meets the requirements:

Initialize before\_seed\_count= ISE (graph, seedset, model)

List pop out two nodes, node1, node2

Initialize after\_seed\_count = ISE difference before and after joining node1

If the difference is greater than the difference of node2 in the previous loop, join node1 directly.

Otherwise, the violence traverses all nodes to find the optimal solution point.

Return to seedset

**It should be pointed out that all the occurrences here are pseudo code!!!**

### 3. Empirical Verification

Describe the experiments that you conducted to test/verify the quality of your program.

#### 3.1 Data set:

First, the dataset provided by the platform was used to verify the algorithm of ISE.

Secondly, using a random way, I generated some maps of moderate size (about 5000 nodes) and used the violent traversal method to get the corresponding

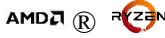
IMP\_seedset (about 30 nodes, in order). Due to the length problem, the data could not be Put it up.

### 3.2 Performance Measure

First of all, regardless of the problem, time is the most important performance parameter, first check whether timeout. Once the timeout needs to be improved.

For the ISE problem, due to the determination of the model, the accuracy is largely limited by the time of each run. The less time each run, the more times, the higher the accuracy, the accuracy and time are linked together.

For the IMP problem, time is the most important referee. Because the complexity is at least one complexity above the linear relationship with the scale of the graph, time is the most important referee.

This project use *Python* (🐍) version-3.7.3 with editor *PyCharm-2019.1.3*(👤) and the platform is *Windows 10 Professional Edition* (version 1903) with  **AMD**® **RYZEN** R5-2600X @3.90~4.20GHz 6 Cores and 12 Threads.

### 3.3 Hyperparameters

If there is a hyperparameter, then the hyperparameter must be the seed of the random number. In view of the fact that the seeds of the random number are too many to be resolved one by one, and the seed of the random number is statistically meaning, it will not affect the performance., so I don't think it needs to be discussed. Other parameters I don't think are hyper-parameters, including the number of multi-processes used.

### 3.4 Experimental results:

Please see them in the excel form.

ISE:

Dataset	RunTime	Result
NetHEPT-seeds50-LT	102.62	1454.6630859375
NetHEPT-seeds50-IC	102.73	1127.396484375
network-seeds5-LT	51.56	37.1015625
network-seeds5-IC	51.72	30.6171875

Those are the public dataset's result their average result ( $\lim_{n \rightarrow \infty} avg$ ) should be



Network-seeds5-IC	30.47
Network-seeds5-LT	37.02
NetHEPT-seeds50-IC	1127.44
NetHEPT-seeds50-LT	1456.98

Each one of the diffs is smaller than 1%.

There form come from the web:

NetHEPT-seeds50-LT	102.41	1454.7
NetHEPT-seeds50-IC	102.4744	1127.396
NetHEPT-seeds50-IC	52.4982	1127.396
random-graph50-50-seeds-5-LT	51.77532	41.13672
random-graph5000-5000-seeds-50-LT	52.10597	1040.155
NetHEPT-seeds50-IC	102.8802	1127.396
random-graph50-50-seeds-5-LT	51.68772	41.13672
random-graph5000-5000-seeds-50-LT	51.84519	1040.155
random-graph1000-1000-seeds-10-LT	51.7911	190.6035
random-graph1000-1000-seeds-10-IC	51.85243	176.5029
random-graph500-500-seeds-10-LT	51.90976	209.6729
random-graph5000-5000-seeds-50-IC	52.10155	943.7539
random-graph1000-1000-seeds-10-IC	52.161	176.5
random-graph500-500-seeds-10-LT	51.88053	209.6729
random-graph1000-1000-seeds-10-IC	51.95525	176.5029

random-graph5000-5000-seeds-50-LT	51.86203	1040.155
random-graph1000-1000-seeds-10-LT	51.84283	190.6035
random-graph50-50-seeds-5-IC	52.06382	30.70215
random-graph500-500-seeds-10-IC	52.99558	167.6846
random-graph5000-5000-seeds-50-LT	52.06358	1040.155
random-graph500-500-seeds-10-IC	52.39773	167.6846
random-graph5000-5000-seeds-50-IC	52.12047	943.7539
random-graph50-50-seeds-5-LT	51.64786	41.13672
random-graph50-50-seeds-5-IC	51.78931	30.70215
random-graph50-50-seeds-5-LT	51.64949	41.13672
random-graph50-50-seeds-5-LT	51.76895	41.13672

Maybe something exists more than once, it shows it is stable.

IMP:

random-graph5000-50-IC	76.61369	297.858
random-graph15000-50-IC	101.8415	219.3255
random-graph500-10-LT	4.702714	112.8801
random-graph50-5-LT	1.454002	31.9282
random-graph500-10-LT	4.016643	122.3548
random-graph15000-50-IC	101.7987	432.7348
random-graph1000-10-IC	3.29643	95.5042
random-graph50000-100-IC	102.4791	457.8284

random-graph500-10-IC	3.286068	101.3713
random-graph15000-50-LT	101.7662	257.8736
random-graph50-5-LT	1.739271	38.025
random-graph1000-10-LT	7.421616	110.9524
random-graph100000-100-IC	103.6838	614.5768
random-graph100000-100-IC	103.8274	264.5746
random-graph500-10-IC	2.50118	100.5331
random-graph50-5-IC	1.639718	27.9654
random-graph50-5-IC	1.609352	24.0143
random-graph50000-100-LT	102.6107	334.1854
random-graph50-5-LT	1.63189	37.1593
random-graph500-10-IC	2.673277	93.7741
random-graph500-10-LT	4.865753	103.6944
random-graph15000-50-LT	101.7029	371.8177
random-graph50000-100-LT	102.4485	338.2596
random-graph1000-10-IC	2.86	85.2138
random-graph1000-10-LT	8.138304	88.8481
random-graph100000-100-IC	103.5341	445.094
random-graph1000-10-LT	8.407256	128.671
random-graph1000-10-IC	2.904001	88.1909
random-graph50000-100-IC	102.4437	302.6005
random-graph5000-50-LT	77.37488	261.091
random-graph50-5-IC	1.639718	27.9654
random-graph15000-50-IC	101.8795	301.7423
random-graph50000-100-IC	102.6441	374.3926
random-graph50000-100-IC	102.8641	411.6554
random-graph5000-50-IC	76.59387	327.8459
random-graph500-10-LT	4.302436	107.8817
random-graph5000-50-LT	76.58068	294.2318

random-graph50000-100-LT	102.4794	344.003
random-graph15000-50-LT	101.8663	267.1956
random-graph50000-100-LT	102.7066	302.6817
random-graph100000-100-LT	104.2123	284.086
random-graph5000-50-LT	76.6273	226.2384
random-graph50-5-IC	1.392601	23.0684
random-graph1000-10-IC	3.153191	80.6994
random-graph15000-50-IC	101.6888	318.5306
random-graph50000-100-LT	103.4777	308.6154
random-graph15000-50-IC	101.8795	301.7423
random-graph100000-100-LT	103.6048	353.4779
random-graph1000-10-LT	7.761585	131.4486
random-graph50-5-IC	1.587664	26.5563
random-graph15000-50-LT	101.7231	328.9374
random-graph50000-100-IC	102.5166	280.8601
random-graph100000-100-IC	103.9139	202.6546
random-graph100000-100-LT	103.598	330.1379
random-graph5000-50-LT	76.57325	802.7251
random-graph15000-50-LT	101.7207	361.4183
random-graph500-10-LT	4.287291	82.0323
random-graph500-10-IC	2.188308	64.002
random-graph50-5-LT	1.526827	34.5855
random-graph50-5-LT	1.745689	35.2418
random-graph5000-50-LT	76.58049	238.8915
random-graph100000-100-LT	103.4554	279.1619
random-graph15000-50-IC	101.8592	233.2061
random-graph100000-100-	103.7085	377.7722

LT		
random-graph1000-10-IC	3.150323	94.6206
random-graph500-10-IC	2.513502	72.0424
random-graph1000-10-LT	8.667221	99.5432
random-graph100000-100-IC	103.7075	261.7677
random-graph50-5-IC	1.671899	27.1183
random-graph5000-50-IC	76.99162	309.8272
random-graph5000-50-IC	76.50544	300.9854
random-graph5000-50-IC	76.5939	343.3427

All of those data come from the web 10.20.107.171:8080 's packets use F12.

Because the web doesn't show IMP's correct answer, so it's hard to analysis. The only thing sure is that imp's TLE is less than the ISE's.

### 3.5 Conclusion

Analysis of the experimental results shows that for a limited time, the program has at least some output, which is in line with my expectations.

For the ISE problem, since the model has been built, the problem is only to convert it efficiently into code, so both performance and time are in line with my expectations of the program. No matter accuracy or time, there is basically no blame. local.

For the IMP problem, since this is an NP-hard problem, as long as the algorithm has output, it has met my expectations. The part that does not meet the expectations is his time and precision. In terms of accuracy, in order to meet the time-consuming, the problem, I added a lot of random simulation code. I think the advantage of the algorithm is that there is output, which can satisfy the demand. The disadvantage is that the code written in Python language is too inefficient. If you use Python with C++, the effect will be better.

## 4. Reference

List the references, please follow the IEEE format to prepare your references. The IEEE format can be found at:

<http://ieeauthorcenter.ieee.org/create-your-ieee-article/use-authoring-tools-and-ieee-article-templates/ieee-article-templates/templates-for-transactions/>

- [1] Kempe, D., Kleinberg, J., & Tardos, É. (2003, August). Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 137-146). ACM.
- [2] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., Faloutsos, C., VanBriesen, J., & Glance, N. (2007, August). Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 420-429). ACM.
- [3] Goyal, A., Lu, W., & Lakshmanan, L. V. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks (Technical Report).
- [4] Goyal, A., Lu, W., & Lakshmanan, L. V. (2011, March). Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web* (pp. 47-48). ACM.
- [5] Tang, Y., Shi, Y., & Xiao, X. (2015, May). Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 1539-1554). ACM.
- [6] 陈卫. (2015). 社交网络影响力传播研究. *大数据*, 1(3), 2015031.
- [7] 吴凯, 季新生, 郭进时, & 刘彩霞. (2013). 基于微博网络的影响力最大化算法. *计算机应用*, 33(08), 2091-2094.
- [8] 宋甲秀, 杨晓翠, & 张曦煌. (2018). 复杂网络中 Top-k 影响力节点的识别算法. *计算机科学与探索*, 12(6), 928-939.
- [9] 曹玖新, 闵绘宇, 徐顺, & 刘波. (2016). 基于启发式和贪心策略的社交网络影响最大化算法. *东南大学学报: 自然科学版*, 46(5), 950-956.
- [10] 田家堂, 王轶彤, & 冯小军. (2011). 一种新型的社会网络影响最大化算法 (Doctoral dissertation).