IMP Project

Kai Lin 11610205
School of Computer Science and Engineering
Southern University of Science and Technology
Email: 11610205@mail.sustc.edu.cn

1. Preliminaries

Influence maximization(IMP) is the problem of finding a small subset of nodes(seed nodes) in a social network that could maximize the spread of influence.Te influence spread is the expected number of nodes that are influenced by the nodes in the seed set in a cascade manner. And I will deeply study this problem into two parts. Te first part is to find a influence value on specific seeds into a network graph, the second part is to find the give number seed set that best spread seed in given network graph.

1.1. Software

This project is implemented in Python using IDE PyCharm. The libraries being used includes Numpy , multiprocessing , time , sys , getopt and Random.

1.2. Algorithm

The Algorithm being used in this project including climbing the IC model ,LT model and the ISE for calculate the influence value in part 1. Then the part 2 is the Greedy Algorithm that CELF for finding small seed number and get a heuristic algorithm for find big seed number (larger then 10). [1]

2. Methodology

For ISE part is write for IC and LT model that do the calculate the influence value on specific seeds into a network graph . Then is the second part to find a influence value on specific seeds into a network graph, the second part is to find the give number seed set that best spread seed in given network graph . IMP problem can find best result by greedy algorithm , but its time is to long for large graph.For CELF it can do for fast enough for a limit number seeds(less then 10) but if seed number larger the data need refresh is more and more[1].So for seeds number larger then 10 use another heuristic algorithm to do calculate for a beeter cost performance .

2.1. Representation

Firstly , global variables defined ahead of the file

- **vertices**: the number of vertices.
- **edges**: the number of edges.
- **seeds**: the seeds given in ISE.
- **seeds_num**: the number of seeds in IMP.

Then the class Graph

- **Graph**: the class for main data need use .
  - node: a set for all node attend.
  - edge: a list to remember all edge for the graph .
  - in_edge: a list to remember all in edge for the graph .
  - weight: a dict to remember all edge distance for the graph .

Other use data in the algorithm

- **file_name**:the file name get from command line .
- **time_limit**:the time limit get from command line .
- **model**:the model need to use in this calculate .
- **seed_path**:the seeds path name in ISE .
- **seed_num**:the seeds number in IMP .

2.2. Architecture

Here are the important functions in Python file ISE with pesudocode format.

- **IC_model**: the calculate modle by the weight for edges only .
- **LT_model**: the calculate modle by the weight for edges and the weight that has active points .
- **spread_check**: use the prescribed model to calculate the end result.

Then is other important functions in Python file IMP with pesudocode format.

- **CELF**: the Greedy algorithm to find good seeds in small number .
- **fast_find**: the Heuristic algorithm to find good seeds in big number .

2.3. Detail of Algorithm

Here show the psudocode main use that IC_model , LT_model , spread_check ,CELF and fast_find .

Algorithm 1 IC_model:the calculate modle by the weight for edges only .

```
Input: Graph graph ,set seed
Output: int result
function IC_MODEL(graph, seed)
    work ← seed
    all ← seed
    result ← len(seed)
    while len(work) > 0 do
        new_work ← set()
        for node in work do
            for near_node in graph.edge[node] do
                if random() larger then edge_weight then
                    if near_node not in all then
                        all.add(near_node)
                        new_work.add(near_node)
                    end if
                end if
                result+ = len(new_work)
                work = new_work
            end for
        end for
    end while
    return result
end function
```

Algorithm 2 spread_check:use the prescribed model to calculate the end result.

```
Input: Graph graph ,set seed,Stringmodel
Output: float result
function SPREAD_CHECK(graph, seed, model)
    result ← 0.0
    if model == "IC" then
        for  i in range(10000) do
            sum+=IC_model(graph,seed)
        end for
    end if
    if model == "LT" then
        for  i in range(10000) do
            sum+=LT_model(graph,seed)
        end for
    end if
    return result/10000
end function
```

Algorithm 3 LT_model:the calculate modle by the weight for edges and the weight that has active points .

```
Input: Graph graph ,set seed
Output: int result
function LT_MODEL(graph, seed)
    work ← seed
    all ← seed
    result ← len(seed)
    ran ← dict()
    while len(work) > 0 do
        new_work ← set()
        for node in work do
            for near_node in graph.edge[node] do
                if near_node not in all then
                    tol_weight take all connect edge has
added weight
                    if near_node not in ran then
                        ran[near_node] ← random()
                    end if
                    if tol_weight > ran[near_node] then
                        all.add(near_node)
                        new_work.add(near_node)
                    end if
                end if
            end for
            result+ = len(new_work)
            work = new_work

            return result

```

Algorithm 4 CELF:the Greedy algorithm to find good seeds in small number .

```
Input: Graph graph ,int seed_num,Stringmodel
Output: set ans
function CELF(graph, seed_num, model)
    ans ← set()
    ans_list ← dict()
    for node in graph.node do
        ans.add(node)
        ans_list[node]=spread_check(graph,ans,model)
        ans.remove(node)
    end for
    new_add=max(ans_list,key=ans_list.get)
    ans.add(new_add)
    while len(ans)<seed_num do
        new_add=max(ans_list,key=ans_list.get)
        ans_list[new_add]=spread_check(graph,new_add—ans,model
        new_add_r=max(ans_list,key=ans_list.get)
        if new_add == new_add_r then
            ans.add(new_add)
            ans_list.pop(new_add)
        end if
    end while
    return ans
end function=0
```

Algorithm 5 fast_find:the Heuristic algorithm to find good seeds in big number .

Input: $Graph$ graph ,int seed_num,$String$model
Output: set ans

```
function FAST_FIND(graph, seed_num, model)
    point ← 0
    ans ← set()
    ans_list ← dict()
    has_add ← set()
    for node in graph.node do
        ans.add(node)
        ans_list[node]=spread_check(graph,ans,model)
        ans.remove(node)
    end for
    ans_list=sorted(ans_list)
    while len(ans)<seed_num do
        new_add= ans_list[point]
        if new_add not in has_add then
            ans.add(new_add)
            has_add.add(new_add)

            do the model IC or LT use ans
            when make one node active
            has_add.add(node)

            point+=1
        end if
        if new_add in has_add then
            point+=1
        end if
    end while
    return ans
end function
```

For the ISE just use the one special model and the spread_check is enough

For the IMP use the one special model , the spread_chec and choice method by the need seed_num if seed_num smaller then 10 use CELF is enough. If larger then 10 use the fast_find method that say in this article.

3. Empirical Verification

In Empirical Verification,I used two data sets to test my ISE and IMP algorithms separately. In IMP, I will use the basic CELF and the fast algorithm (design by myself) to compare the results and time, and discuss the use domain by the result and time use.

3.1. Design

For ISE is use the special model to calculate for enough times .

The IMP problem is a NP hard problem and under the IC model and LT model its computation is P hard. Because we are only given a little network data so the design of the testing plan become much more harder. We should consider the difference of model we use and the key factor should be the model, strategy, size of data.

3.2. Data and data structure

In this project, the dataset I use is network.txt and NetHEPT.txt. These two datasets are one small dataset with 62 vertices and 159 edges, and the other is a large dataset with 15233 Vertices and 58891 edges.

3.3. Performance and result

ISE

data set	run time	result
Network-seeds-IC	1.166s	5.02
Network-seeds-LT	1.355s	5.0226
Network-seeds2-IC	2.110s	30.4695
Network-seeds2-LT	3.480s	36.9932
NetHEPT-50seeds-IC	25.654s	1002.5355
NetHEPT-50seeds-LT	57.781s	1268.179
NetHEPT-5seeds-IC	8.569s	276.2911
NetHEPT-5seeds-LT	35.646s	337.4792

For ISE the time relate the graph size and seed number the result has shown in the table

IMP

data set	run time	result
Network-5-IC	1.306s	30.72
Network-5-LT	1.289s	37.54
NetHEPT-5-IC	9.036s	322.89
NetHEPT-5-LT	18.316s	392.975
NetHEPT-50-IC-bonus	55.618s	1161.6776
NetHEPT-50-LT-bonus	56.879s	1522.8746

Use CELF for small seed number can get a good result in limit time but if do for 50 seeds in NetHEPT will take for about 400s but for fast_find one time just about 3 s and the result has shown in the table.

3.4. Hype-parameters

In ISE, my Hype-parameter is the average calculation parameter, and I set it to 10000 in order to get a stable solution.

In IMP, I set the average calculation parameter K=1000 in the basic greedy algorithm and CELF. However, when the value of seed_size is greater than 10 or the fixed point and number of edges of the graph are too large, I set my average calculation parameter time = 100 in fast_find. The purpose is to be able to draw results within a limited time .

3.5. Analysis

For the Influence maximization(IMP) project is like the carp problem and not like carp. The IMP is NP hard problem is large then CARP that both can not check all result in a limited time . But for CARP use get initial solver and do for better result.But IMP is for get result in add result add node one by one . So for fast_find result the problem is to take the right heuristic .For IMP has many good algorithm like Greedy,CELF_Greedy, Lv_CELF _Greedy,CELF++_Greedy and Lv_CELF++_Greedy [2].For many codes try and understand then write by self thought is the best project do in my understant .

Acknowledgments

I would like to thank my friends Peijun Ruan who together discuss the IMP's frame . Read and discuss the parper which the hard to understand part .Last I would like to thank SA who will check my codes and reports.

References

[1] Li, J. Fan, Y. Wang and K. Tan, "Influence Maximization on Social Graphs: A Survey", IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 10, pp. 1852-1872, 2018.

[2] Lv, J. Guo, Z. Yang, W. Zhang and A. Jocshi, "Improved Algorithms OF CELF and CELF++ for Influence Maximization", Journal of Engineering Science and Technology Review, vol. 7, no. 3, pp. 32-38, 2014.