Project 1 – Information Exposure Maximization

1. Overview

Information Exposure Maximization (IEM), which selects two sets of users (called campaigns) from a social network to maximize the expected number of vertices that are either reached by both campaigns or remain oblivious to both campaigns, is an important algorithmic problem in social influence analysis. The IEM problem is theoretically complex and it has been proven that obtaining an optimal solution of IEM is NP-hard. In this project, you need to design two search algorithms, one heuristic algorithm and one evolutionary algorithm (or simulated annealing algorithm), to solve the IEM problem. The score you get in this project will be given according to the performance of your algorithms in our test.

2. Problem Description

Information Exposure Maximization is modeled as an algorithmic problem in [1] to reduce the echo chamber and filter-bubble effect: users get less exposure to conflicting viewpoints and are isolated in their own informational bubble. This problem studies a social network represented as a graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, where V is the set of nodes in G (i.e., users) and E is the set of edges in G (i.e., social links between users). The goal of the IEM problem is to find two sets of users with the maximum balanced information exposure. In the following, several preliminary definitions are provided first, and then a formal definition and an example of the IEM problem are presented.

1) Preliminary Definitions

Social Network: $\boldsymbol{G = (V, E)}$, where $V = \{v_1, \cdots, v_n\}$ represents the node set, and $E = V \times V$ represents the edges between nodes.

Campaigns: $\boldsymbol{C = \{c_1, c_2\}}$ represents two campaigns; each campaign holds a viewpoint.

Initial Seed Set: $\boldsymbol{I_i \subseteq V, i \in \{1, 2\}}$ represents the initial seed set for campaigns c_i.

Balanced Seed Set: $\boldsymbol{S_i \subseteq V, i \in \{1, 2\}}$ represents the target seed set that

you need to find for each campaign c_i.

Budget: $\boldsymbol{k}$ represents the size of the target seed set; $|S_1| + |S_2| \leq k$.

Diffusion Probability: $\boldsymbol{P_i = \{p^i_{(u,v)} | (u,v) \in E\}, i \in \{1, 2\}}$ represents the edge weight associated with campaign c_i, where $p^i_{(u,v)}$ represents the probability of node u activating node v under each campaign c_i.

Diffusion Model: $\boldsymbol{M}$ captures the stochastic process for seed set $U_i = I_i \cup S_i$ spreading information on G. We assume that information on the two campaigns propagates in the network following the independent cascade (IC) model. The two campaigns' messages propagate independently of each other (such propagation is often called *heterogeneous propagation*). The diffusion process of the first campaign (the process for the second campaign is analogous) unfolds in the following discrete steps:

i. In step $t = 0$, the nodes in seed set U_1 are activated, while the other nodes stay inactive;

ii. Each active user u for campaign c_1 in step t will activate each of its outgoing neighbor v that is inactive for campaign c_1 in step $t - 1$ with probability $p^1_{(u,v)}$;

    The activation process can be considered as flipping a coin with head probability $p^1_{(u,v)}$: if the result is head, then v is activated; otherwise, v stays inactive;

    Note that u has only one chance to activate its outgoing neighbors for campaign c_1. After that, u stays active and stops the activation for campaign c_1;

iii. The diffusion instance terminates when no more nodes can be activated.

Exposed Nodes: Given a seed set U, $r_i(U)$ is the vertices that are reached from U using the aforementioned cascade process for campaign c_i. Note that in one propagation, in addition to nodes that were successfully activated by U, nodes that were once attempted to be activated but were not successfully activated by U are also considered to be reached by U. Since the diffusion process is random, $r_i(U)$ is a random variable.

2) Problem Formulation

Given a social network $G = (V, E)$, two sets I_1 and I_2 of initial seeds of the two campaigns, and a budget k. The IEM is to find two sets S_1 and S_2, where $|S_1| + |S_2| \leq k$, and maximize the expected number of vertices that are either reached by both campaigns or remain oblivious to both campaigns, i.e.,

$$\max \Phi(S_1, S_2) = \max \mathbb{E}\left[\left|V \setminus \left(r_1(I_1 \cup S_1) \vartriangle r_2(I_2 \cup S_2)\right)\right|\right]$$

$$\text{s.t.} \quad |S_1| + |S_2| \leq k$$

$$S_1, S_2 \subseteq V,$$

where $\vartriangle$ means symmetric difference, $A \vartriangle B = (A \setminus B) \cup (B \setminus A)$.

3) An example of the IEM problem

An example of the IEM problem is shown in **Fig. 1**. Given the social network $G = (V, E)$ with two campaigns c_1 and c_2. The diffusion probabilities of the two campaigns are shown on the edges, where the red one is the edge weight associated with campaign c_1 and the blue one is the edge weight associated with campaign c_2. Assume that the initial seed sets of the two campaigns are $I_1 = \{1\}$, $I_2 = \{9\}$, respectively, and the budget $k = 2$, the IEM aims at discovering two sets S_1 and S_2 with $|S_1| + |S_2| \leq k$ and with maximum $\Phi(S_1, S_2)$.

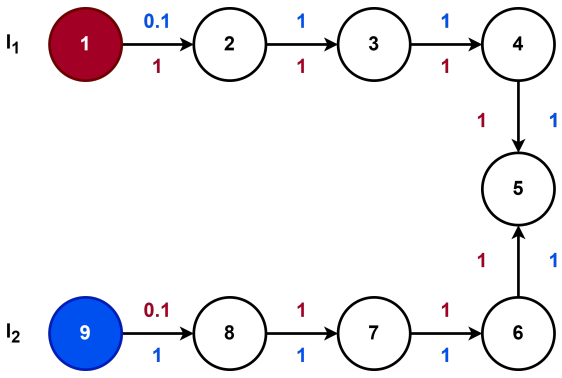

Fig. 1. An example of IEM problem

The optimal solution of this IEM problem instance is $S_1 = \{8\}, S_2 = \{2\}$ and $\Phi(S_1, S_2) = 7$.

3. Experimental Platform

1) Programming Language

Python version: 3.10

2) Input and Output Format of Algorithm

The executable evaluator must be named **Evaluator.py**, and the executable solver of two kinds of algorithms must be named **IEMP_Heur.py** and **IEMP_Evol.py**.

i. Input:

1. The format of the evaluator call should be as follows:

   python Evaluator.py –n < social network > -i < initial seed set > -b <balanced seed set> -k < budget > -o <object value output path>

2. The format of the solver call should be as follows:

   python IEMP_ Heur.py –n < social network > -i < initial seed set > -b <balanced seed set> -k < budget >

< social network > is the absolute path of the social network file, which is required to conform to the format of the social network specified below.

< initial seed set > is the absolute path of the two campaigns' initial seed set, which is required to conform to the format of the seed set specified below.

< balanced seed set > is the absolute path of the two campaigns' balanced seed set, which is required to conform to the format of the seed set specified below. Note that this is the input file path of the **Evaluator.py** and the output file path of the **IEMP_Heur.py** and **IEMP_Evol.py.**

< budget > is a positive integer budget.

<object value output path> is the output absolute path of the objective value.

ii. Output:

1. The output of the evaluator call should be the objective value written into the **<object value output path>**.

2. The output of the solver call should be the balanced seed set of the two campaigns found by your algorithm written into the < **balanced seed set** >.

iii. The format of the social network is as follows:

1. The first line contains the number of nodes n and edges m
2. The following m line is the edge of the social network:
   First column: ID of the source node, $u \in [0, n-1]$;
   Second column: ID of the target node, $v \in [0, n-1]$;
   Third column: the first campaign weight $p^1_{(u,v)}$ of edge;
   Fourth column: the second campaign weight $p^2_{(u,v)}$ of edge;

iv. The format of the seed set (initial seed set and balanced seed set) is as follows:

1. The first line contains the number of two seed sets k_1 and k_2;
2. The following k_1 line is the seed of campaign c_1;
3. The following k_2 line is the seed of campaign c_2;

3) Dataset

Table 1 provides an overall description of the test datasets:

1. **Type Column:** graph type, all datasets are directed graphs.
2. **Nodes Column:** number of nodes in the graph.
3. **Edges Column:** number of edges in the graph.

Table 1 Summary of Datasets

Dataset	Type	Nodes	Edges
Dataset 1 [Test 2]	Directed	475	13,289
Dataset 2 [Test 2]	Directed	36,742	49,248
Dataset 3 [Test 3]	Directed	about 10,000	about 100,000
Dataset 4 [Test 3]	Directed	about 20,000	about 1000,000

4. Grading Rules

The scoring rules for this project comprise two components: the project report and code evaluation. The total value of this project is 15 points.

1) Project Report

A report about IEM (in pdf format) must be submitted, in which you should describe the core idea of your algorithm design, entail each component of your algorithm, illustrate the algorithm structure, and give the pseudo-code. Both the heuristic algorithm and the evolutionary algorithm that you designed must be included.

Please note that your code will only be assessed and scored if you have submitted the project report.

2) Code Evaluation

In the programming aspects, there are three components: the objective evaluation (2 points), the heuristic algorithm (6.5 points), and the evolutionary algorithm or simulated annealing algorithm (6.5 points). After you submit your project, we will run some scripts to test your IEM solvers on different IEM problem instances. You will get the corresponding score once your solvers have passed a test.

The scoring rule for the objective evaluation is as follows. We will use two instances to test your evaluator, with each instance worth 1 point.

i. **Usability Test (0.2 points):** In this test, we will check whether your evaluator meets the input and output requirements. *Please keep in mind that only those estimators that have passed test **i** will be evaluated in test **ii**.*

ii. **Accuracy Test (0.5 points):** In this test, we will assess the estimation error of your estimator. To be specific, we will establish a common cutoff time and an acceptable range for estimation values. We will then evaluate whether your estimated values fall within the specified range. *Please keep in mind that only estimators that have successfully completed test **ii** will undergo evaluation in test **iii**.*

iii. **Efficiency Test (0.3 points):** In this test, we will focus on how fast your evaluator is. To be specific, we will set a common cutoff time and measure the running time of a baseline. Subsequently, we will assess whether your evaluator can complete its task in less time compared to the baseline.

The score rules for the heuristic algorithm and the evolutionary algorithm (or simulated annealing algorithm) are consistent and outlined as follows, with each algorithm worth 6.5 points.

i. **Efficacy Test (3.9 points):** In this test, we will focus on how good your solvers are on three instances, each worth 1.3 points.

   Specifically, we will first check whether your solvers can satisfy the input and the output requirements **(test a, 0.1 points)**. Subsequently, we will establish a common cutoff time and the solution quality of a baseline. We will then evaluate whether your solver can get a larger value of balanced information exposure within the given time **(test b, 0.9 points)**. If your solver can get significantly better solution quality within the given time or complete the task in significantly less time compared to the baseline, you can get the points for your advanced performance **(test c, 0.3 points)**.

   *Please keep in mind that only solvers that have successfully completed test **a** will undergo evaluation in test **b**, and only solvers that have successfully completed test **b** will undergo evaluation in test **c**.*

ii. **Robustness Test (2.6 points):** A robust solver is said to be applicable to a wide range of problem instances. Although your solvers may show good performance in the efficacy test, they may perform badly in unseen instances. In this test, for each participant solver, we will generate two new instances, each worth 1.3 points.

   On each instance, like the efficacy test, we will set a common cut-off time for all participant solvers and the balanced information exposure of a baseline. Subsequently, we will first check whether your solvers can satisfy the input and the output requirements **(test a, 0.1 points)** and evaluate whether your solver can get a larger value of balanced information exposure within the given time **(test b, 0.9 points)**. If your solver can get significantly better solution quality within the given time or complete the task in significantly less time compared to the baseline, you can get the points for your advanced performance **(test**

c, 0.3 points).

*Please keep in mind that only solvers that have successfully completed test **a** will undergo evaluation in test **b**, and only solvers that have successfully completed test **b** will undergo evaluation in test **c**.*

In summary, the total score of this project is $15 = 2 \times (0.2 + 0.5 + 0.3) + 2 \times (3 \times (0.1 + 0.9 + 0.3) + 2 \times (0.1 + 0.9 + 0.3))$.

2. Reference

[1] K Garimella, A Gionis, N Parotsidis, N Tatti. Balancing information exposure in social networks. NeurIPS 2017: 4663-4671

[2] S Cheng, H Shen, J Huang, W Chen, X Cheng. IMRank: influence maximization via finding self-consistent ranking. SIGIR 2014: 475-484

[3] M Gong, J Yan, B Shen, L Ma, Q Cai. Influence maximization in social networks based on discrete particle swarm optimization. Inf. Sci. 367-368: 600-614 (2016)

[4] Q Jiang, G Song, G Cong, Y Wang, W Si, K Xie. Simulated Annealing Based Influence Maximization in Social Networks. AAAI 2011