Influence maximization

Víctor Emiliano Cruz Hernández

June 17, 2023

1 Introduction

Finding the most influential people is a NP-Hard problem. Authors have formalized the problem as:

Given a weighted graph in which nodes are people and edge weights represent the influence of the people on each other it is desired to find k starting nodes that their activation leads to maximum propagation based on a chosen influence maximization model.

Assume we have data on a graph which estimates which individuals influence on others. We would like to market a new product that we hope will be adopted by a large fraction of the network by initially targeting a few "influential" members of the network and giving them free samples of the product, hoping we can maximize the number of people influenced by this individual. The problem aims to find a number of people that are able to maximize the spread of influence through a target social network.

1.1 Influence Maximization model

.

In this section we will explain how in our model we say a person is influenced by it's peers. There are two most prevalent diffusion models in computer science; the independent cascade and linear threshold model, both have different properties that concern our influence maximization problem. We will not take further detail on which one we will use, as we just lead the following idea:

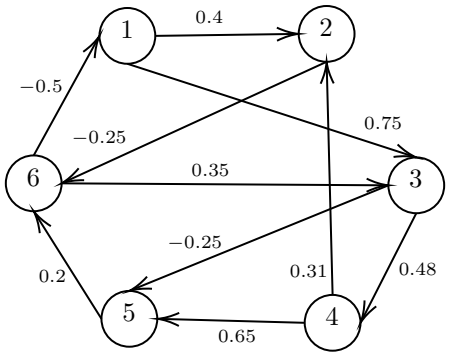
Given a *threshold* value θ and our di-graph $G = (V, E)$ where $v_i \in V$ is a person and $W = \{u_0, \ldots, u_k\} \subset V$ is the set of people such that for $j \in \{0, \ldots k\}$, $(u_j, v_i) \in E$. We say that v_i is influenced iff:

$$\sum_{u \in W} w((u, v_i)) \geq \theta$$

Where $w((u, v_i))$ is the weigth of the (u, v_i) edge.

Instead of pursuing a "cascade of influence" were influencers can spread their influence further it's own neighbors. We restrict their influence in the scope of their immediate neighbors in G.

We can take for example the following digraph with $\theta = 0.5$. In this example, we should be able to find for $k = 2$ the set $\{v_1, v_4\}$ as the set of k influencers who reach the most influenced nodes $\{v_2, v_3, v_5\}$.



Notice that v_2 is influenced. For $W = \{v_1, v_4\}$ we see that:

$$\sum_{u \in W} w((u, v_2)) = w((v_1, v_2) + w((v_4, v_2)) = 0.4 + 0.31 = 0.71 \geq \theta = 0.5$$

2 Heuristic

We use Ant Colony Optimization (ACO) as C.Salavati and A.Abdollahpouri presented in their paperwork [1].

2.1 Neighborhood

Let $D = (V, E)$ our initial relation graph. We use the following definitions for building the neighborhood graph N_D.

Definition 2.1 ($similarity(u, v)$)**.** Let $u, v \in V$ where $outN(u), outN(v)$ are the out-neighbors of u, v respectively. And $outN2(u)$ as the set of all neighbors of the neighbors of u:

$$outN2(u) = \bigcup_{w \in outN(u)} outN(w)$$

We define the similarity of u, v, $similarity(u, v)$ as:

$$similarity(u, v) = \begin{cases} 1 & u \in outN(v) \\ \alpha \frac{|outN(u) \cap outN(v)|}{|outN(u) \cup outN(v)|} + \beta \frac{|outN2(u) \cap outN2(v)|}{|outN2(u) \cup outN2(v)|} & otherwise \end{cases}$$

Note that this relation is not symetric since we have a digraph and β and α have to satisfy $\alpha + \beta = 1$ so we can tune the impact of direct users over second hand users.

Definition 2.2 (Neighborhood Graph: N_D)**.** Given a relational wighted digraph $D = (V, E)$ We define our neighborhood graph $N_D = (V_{ND}, E_{ND}, W_{ND}, S_{ND})$ where:

- $V_{ND} = V$
- $E_{ND} = \binom{V_{ND}}{2}$
- $W_{ND} = E$
- $S_{ND} = \{similarity(u, v) | u, v \in V_{ND}\}$
- $P_{ND} = \{pheromone(u, v) | u, v \in V_{ND}\}$

We define our graph as a complete digraph where the edges between are calculated with the original weight in D and the similarity between the two nodes and the pheromone value. Initialy for all $u, v \in V_{ND}$, $pheromone(u, v) = \tau_0$, we set an initial pheromone value to all edges in E_{ND}.

2.2 State transition rule

We use the neighborhood graph N_D to select the k influencers that maximize the influence in the original graph D. Initially we select a random node in N_D.

Let $T = [v_0, \ldots, v_i]$ be the current selected nodes and v_i the last one selected. Then we have to select the next node v_j with a *state transition rule* . It is a greedy rule defined by:

- The pheromone value of (v_i, v_j): $phermomone(v_i, v_j) \in P_{ND}$.
- The *profit value rule*: $profitValue(T \cup \{v_j\})$.
- The *mutual similarity rule*: $S(T \cup \{v_j\})$.

We proceed to define such rules:

Definition 2.3 (Profit value rule: $profitValue(T, v_j)$)**.** Let T be the set of current selected nodes and v_j the next node to select. We define the *profit value rule* of T and v_j as:

$$profitValue(T) = r \sum_{u \in V_{ND}} influenced(T, u) - ck$$

where

$$influenced(T, u) = \begin{cases} 1 & \sum_{r \in T} w((r, u)) \geq \theta \\ 0 & otherwise \end{cases}$$

where:

- r is a parameter which is the revenue from buying a product.
- c is the cost of selecting a node as influence node.

- k is the size of our desire influence node set.

The *profit value rule* allows us to calculate how many nodes are influenced by a given set of nodes T.

Definition 2.4 (Mutual similarity rule: $S(T)$)**.** We define the *mutual similarity rule* as :

$$S(T) = \sum_{u \in T} \sum_{v \in T} similarity(u, v)$$

This is the amount of similarity that a set of nodes T has between each pair of themselves.

Finally we can define the *state transition rule* as follow:

Definition 2.5 (State transition rule: $greedTransition(T = (v_0, \ldots, v_i))$)**.** Let T be the current set of selected nodes, we select the next node greedly by:

$$greedTransition(T = [v_0, \ldots, v_i]) = max_{v_j \in V_{ND} - T} \{edgeValue(T, v_j)\}$$

where:

$$\begin{aligned} edgeValue(T = [v_0, \ldots, v_i], v_j)] = & pheromone(v_i, v_j)^{\eta} * \\ & profitValue(T \cup \{v_j\})^{\psi} * \\ & S(T \cup \{v_j\})^{\gamma} \end{aligned}$$

where η, ψ, γ are a parameters used to control the importance of pheromone trail versus heuristic information.

2.3 Fitness

After an ant has finished constructing its path, we evaluate the suitability og each path scrolled with a *fitness function*. It is a combination of the *profit value* and *mutual similarity*.

Definition 2.6 (Fitness function: $fitness(T)$)**.** Given a path T we evaluate the path with te *fitness function* as:

$$fitness(T) = profitValue(T)^{\psi} * \left(\frac{1}{S(T)}\right)^{\gamma}$$

2.4 Pheromone update

After an ant has completed it's path we update the pheromome of each edge in P_{ND} as follows: Let T be the path build by the ant.

$$pheromone(v_i, v_j)_t = (1 - \rho) pheromone(v_i, v_j)_{t-1} + fitness(T)$$

where $t - 1$ is the inmediate past iteration of iteration t.

3 Implementation

3.1 Generation of the graph

We used a real dataset for testing our system. The Extended Epinions collection, collected by Paolo Massa. The datasets we use can be found in `epinions/`. It contains two files `rating_relation.csv` where we can found users that qualify other user's job and `user_rating.csv` where we have the relation between two users as the trust/distrustment a user have on another.

3.1.1 rating_relation.csv

This file represent the ratings a user has given to other user. A posible line in the file can be like this: `317856,234885,5`. We can read the line as the edge $(317856, 234885)$ where user 317856 rated 234885 with a 5.

In the database, a user can rate another user in a natural number between 1 and 5. The dataset is not consistent as it has some ratings over 5, for the generation of the graph we take the following scale over the rating:

Scale setting	Rating
1	5
0.75	4
0.25	3
-1	2
-1	1

Note that a user u can rate other user v multiple times. For convenience, we denote as $A_{uv}[i]$ as the i-th rate user u gives to v.

3.1.2 user_rating.csv

This file contains the relation of trust/distrustment one user have over another. A line in this file looks like `361404,81580625796,-1` meaning that user 361404 distrust user 81580625796. If we change it to `361404,81580625796,1` now it means user 361404 trust user 81580625796.

With the previous information we can generate the relation sub-graph with the weights representing both datasets. Given a pair of users u, v we define the weigth of the edge (u, v) as:

$$w(u,v) = \begin{cases} 1 & \textit{if } v \textit{ trust } u \\ -1 & \textit{if } v \textit{ distrust } u \\ \frac{1-e^{-x}}{1+e^{-x}} & \textit{if user } v \textit{ rates } u \\ 0 & \textit{otherwise} \end{cases}$$

where:

$$x = f(u,v) = \sum_{i \in A_{uv}} A_{uv}[i]$$

$f(u, v)$ will be positive if the sum of all ratings that u has given to v is positive and negative if the sum is negative. This way the edges of the graph we generate have weights between $[-1, 1]$.

3.1.3 Generation of the sub-graph

With those datasets, given a seed s we can generate a random graph of n nodes, for example, we can create a graph of `500` nodes with seed `10` in the file `./graph500S10.csv`.

```
$ cargo run --release graph 500 10 ./graph500S10.csv
```

We show the last 10 lines of the file:

```
200396,369121,1
255170,200396,0.46211717
500873,2147483647,1
2147483647,217293,1
255170,525533,0.12435301
530381,483904,-1
253067,2147483647,1
341013,329733,0.90514827
315491,262868,0.46211717
322084,2147483647,0.9969976
```

We can take the first line `200396,369121,1` wich represent the directed edge $(200396, 369121)$ of weight 1.

With any of the graph generated this way we can excecute the second part of the system searching the desire k influencers in the graph.

3.2 Path finding in a graph

Given a graph generated previously and a seed s we can find the k influencers in the graph as follows:

```
$ cargo run --release expr ./graph500S10.csv 20 5000
```

This way we search for the set of $k = 20$ nodes 5,000 times and show the path T when it finishes with a given seed. We show some ot the previous execution's results:

```
   FITNESS: 2.891806, SEED: 41
SOLUTION: [Node  id:  489989 , Node  id:  561193 , Node  id:  282884
, Node  id:  2147483647 , Node  id:  538395 , Node  id:  610716 , Node
id:  514960 , Node  id:  548930 , Node  id:  374388 , Node  id:  211440
, Node  id:  536828 , Node  id:  605439 , Node  id:  500873 , Node
id:  437453 , Node  id:  245729 , Node  id:  440719 , Node  id:  540403
, Node  id:  223319 , Node  id:  569385 , Node  id:  406551 ]
```

```
FITNESS: 2.652228, SEED: 42
SOLUTION: [Node id: 523809 , Node id: 450802 , Node id: 304628
, Node id: 538395 , Node id: 221034 , Node id: 408214 , Node
id: 326696 , Node id: 768511876 , Node id: 211440 , Node id:
497699 , Node id: 257170 , Node id: 488144 , Node id: 2147483647
, Node id: 390490 , Node id: 494303 , Node id: 507426 , Node
id: 368476 , Node id: 499006 , Node id: 557444 , Node id: 518224
]
```

A lower fitness function represent better results

4 Results

This problem is widely studied and used. It can take several approaches such as first selecting the influence maximization model. The way we present this problem and a solution is simple enough to compare results with brute force results yet complicated enough it is unfeasible to scale this comparison in larger graphs.

There is a unfinished approach in representing visually the results, this could help in the experimentation stage. A complete tuning of the system requires this step, furthermore complete this project adequately.

References

[1] Chiman Salavati and Alireza Abdollahpouri. Identifying influential nodes based on ant colony optimization to maximize profit in social networks. *Swarm and Evolutionary Computation*, 51:100614, 2019.

[2] Paulo Shakarian, Abhinav Bhatnagar, Ashkan Aleali, Elham Shaabani, and Ruocheng Guo. *The Independent Cascade and Linear Threshold Models*, pages 35–48. Springer International Publishing, Cham, 2015.

[3] J. Blandy, J.; Orendorff. *Programming Rust*. O'Reilly, 1005 Gravenstein Highway North, Sebastopol, 2017.