Using Influence for Navigation in Online Social Networks

Bastien Lebayle[1], Mehran Asadi[2] and Afrand Agah[1]
(Corresponding author: Afrand Agah)

(Email: aagah@wcupa.edu)
Department of Computer Science, West Chester University[1]
West Chester, PA 19383
Department of Business and Entrepreneurial Studies, The Lincoln University[2]
Lincoln University, PA 19352

Abstract

If influence is the capacity to have an effect on someone, then who are the influential people in an Online Social Network?

In this paper, we investigate the role of influential people in an Online Social Network. Then we present an algorithm that take advantage of influential people to reach a target in the network. Our navigation algorithm returns a path between two nodes in an average of 10% less iterations, with a maximum of 83% less iterations, and only relies on public attributes of a node in the network.

1 Introduction

Over the course of human history, the collections of social ties among friends have grown steadily in complexity. When people live in neighborhoods or attend schools, the social environment already favors opportunities to form friendships with others like oneself.

In the most basic sense, a network is any collection of objects in which some pairs of these objects are connected by links. In a network of objects, objects can be people or computers, which we refer to them as nodes of the network. Have the people in the network adapted their behaviors to become more like their friends, or have they sought out people who were already like them [4]?

Here we are interested in connectedness at the level of behavior - the fact that each individual's actions have implicit consequences on the outcomes of everyone in the system [4]. We investigate prediction of peoples behavior and influences in online social networks. Our focus is on how different nodes can play distinct roles in information flow through an online social network.

This paper is organized as follows. Section 2 reports the related work. In Section 3 we define three different types of crawlers for navigation in Online Social Networks. Section 4 evaluates the performance of the proposed protocol, and Section 5 concludes the paper.

2 Related Work

Targeted crawler algorithms [5] allow a crawler to find a path from a source node to a target node assuming: (i) the crawler controls at least one node in the Online Social Networks (OSN), which is attainable by simply registering on most OSN, and (ii) the crawler knows when its target is reached, either by comparing attributes or unique ids. The targeted crawler algorithms can be seen as a frontier [5] that is expanding toward the target node using a distance function. The distance function represents how different two nodes are: the more they have in common, the less distant they are.

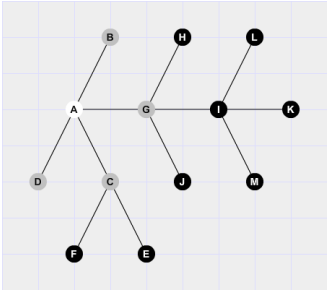

Figure 1: TargetedCrawler at first iteration

For each step in the algorithm, the frontier spreads toward the node that has the least remaining distance to the target. The frontier expansion can be seen in Figure 1 and Figure 2. Assume we want to find a path from node A to node K. At the first step (Figure 1), the frontier consists of all A's friends. Then using the distance function, we find G is the closer to the target (Figure 2). Hence G goes to the explored set represented in white, and the frontier represented in grey is extended to G's friends.

Two procedures to find key players by (i) the identification of key players for the purpose of optimally diffusing a property through the network by using the key players as seeds and (ii) the identification of key players for the purpose of disrupting or fragmenting the network by removing the key nodes, are discussed in [2]. Our work will try to find a procedure to find key players locally for a path between two nodes while [2] focuses on key players for the network as a whole.

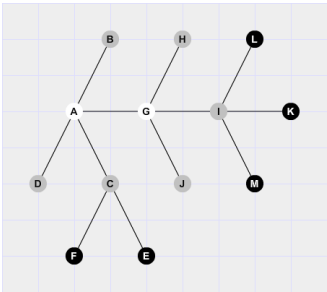

Figure 2: TargetedCrawler at second iteration

The definition of influential nodes changes with application and the type of commodity flowing through a network. While our work focuses on navigation paths, identifying influential nodes in OSN using principal component centrality are developed in [6].

3 Prediction in Online Social Network

Our focus is on how different nodes can play distinct roles in information flow through an OSN. If two people in an OSN have a friend in common, then there is an increased likelihood that they will become friends themselves at some point in the future [4]. The terminology of OSN reflects a largely similar views, through its emphasis on the connections one forms with friends, fans, followers, and so forth.

An Online Social Network can be viewed as a graph G = (V, E). Vertices (nodes) are representing people and edges are representing social links. Social links can be undirected (e.g. friends on Facebook) or directed (e.g. followers on Twitter). An OSN allows people to have attributes that are included in the Social Network list of attributes. For each attribute, people can define a privacy policy: visible to all, visible to friends, or private.

Our work is based on the following assumption: from a graph G with privacy policies on attributes and links, we can deduce a public sub-graph G' based on public attributes and public links [1] and [7] . Because of the default behavior of OSN is to share the most and hide the less, we can deduce G', thanks to users that do not change this default behavior. Each user that is changing the default settings contributes to shrink our graph G'.

We need a targeted crawler algorithm for our influence prediction in order to be able to navigate efficiently in an OSN. This is for the following two main reasons: (i) Online Social Networks are huge (billions of users for Facebook as of today) and should not been seen as random graphs. (ii) If people are connected, there is a high chance they share something in common, either a friendship, a location, a job, etc. We should not use algorithm like breadth-first search in Online Social Networks because they do not take advantage of the probability that people are connected.

3.1 Monte Carlo method

The Monte Carlo method is used when a distribution of an unknown probabilistic entity is close to impossible to determine in a deterministic way. Instead, we compute a non-deterministic algorithm a certain number of times until enough numerical results are collected to generalize the distribution of the unknown probabilistic entity.

3.2 Predicting Influence

We want to be able to locate those people, who are more influential. People are considered influential if when they do an action, their friends, friends of their friends, etc, replicate their actions.

The goal here is to understand influence not just as a property of nodes in a network, but in social interactions as the roles people play in groups of friends in communities or in organizations. Influence is not so much a property of an individual as it is a property of a relation between two individuals. Influence may be almost entirely the result of the personalities of the two people involved. But it may also be a function of the larger social network in which the two people are embedded. One person may be more powerful in a relationship because he occupies a more dominant position in the social network with greater access to social opportunities outside this single relationship.

One way to define important people is as follow: the more paths are going through a node, the more important this node is. This is different from the degree of a node as it also takes into consideration nodes that are not directly linked to it. In figure 3, node A and E have the same degree: 3. However because E's friends are more connected, there are more paths that go through E than paths that go through A.

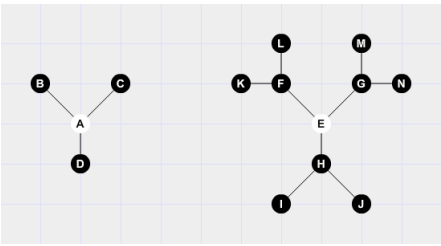

Figure 3: Degree vs. influence

If for 100 computed paths a particular node N_1 is part of 50 paths and another node N_2 is part of 10 paths, then we say N_1 is more important than N_2 because it connects more people together. The influence of a node N in the network is defined in equation 1:

$$Influence(N_i) = \#ofPathsGoingThrough(N_i) \tag{1}$$

Computing (1) is highly time consuming. Once we have the influence associated to each node, a navigation path can be defined by hoping from influent node to influent node until the target is reached.

4 Performance Evaluation

The dataset used for following experimentation comes from Stanford university [3]. Data was collected from users who had manually shared their circles in the Google+ Social Network. The original dataset consisted in 107 thousands nodes and 13.7 millions edges. In order to reduce running time of the experimentation, the dataset has been reduced to 700 nodes and 35 thousands edges. Moreover, bidirectional links between 2 nodes have been made unidirectional. This allows us to enlighten the different behaviors of our crawlers with a smaller data set. Each node of our sample networks is part of at least one link, either the head or the tail of the link. Each node has between 0 and 452 friends. Nodes that have at least one friend have in average 66 friends.

4.1 Simple Crawler

Assume the online network has N nodes. The first way to compute our influence is to compute all possible paths as defined in equation (1). The TargetedCrawler algorithm, which corresponds to the navigation between 2 nodes, is executed $O(N^2)$ times. If no path is found between source and target node, the influence table is not updated. If a path is found between source and target node, for each node N_i being part of the path, $Influence(N_i)$ is incremented by one.

Assuming the TargetedCrawler algorithm runs in $O(1)$ time, computing the full set of paths will be done in $O(N^2)$ time. For a 1 ms computation time

per iteration and 10^6 nodes (which is not unrealistic for OSN, and even far from the truth for some), the computation would take approximately 10^{12} ms or 31 years. We can easily realize how this is infeasible.

4.2 Monte Carlo Crawler

The first approach showed that it is possible to define and influence value for each node in the network. The method described in 4.1 can return very precise results. However the method is also highly time consuming and infeasible for large OSN. Can we reduced the computation time and still have correct results? second approach aims to use the Monte

Node Id	Simple Crawler	Monte Carlo Crawler
206	16.3 (1)	17.2 (1)
422	9.8 (2)	7.87 (4)
170	9.6 (3)	8.91 (2)
938	8.4 (4)	4.48 (5)
675	7.1 (5)	8.91 (2)

Table 1: Monte Carlo Crawler vs. Simple Crawler

Carlo method to approximate the influence of a node. Instead of running the Targeted Algorithm on all possible N^2 paths, we are going to restrict the execution time of the influence calculation to $O(N)$. To do that, our second approach selects two randoms nodes and then runs the Targeted Algorithm N times. With this procedure, the time spent to compute our influence defined in (1) is much shorter. For $N = 10^6$, this approach would end in less than 10^6 ms or 20 minutes which is very feasible. Table 1 is extracted by running the two approaches on our sample network [3]. For the top 5 nodes, it shows the influence defined in equation1 normalized by the number of path computed and the overall ranking of the nodes are in parentheses. We can see that after only N iterations of the Monte Carlo Crawler and instead of the N^2 of our Simple Crawler, the influence of a node is already close to what it will be after N^2 iterations.

4.3 Our approach

Simple Crawler and Monte Carlo Crawler are based on the Targeted Crawler algorithm [5]. The issue with this algorithm is that sometimes the frontier spread in the wrong direction and this can lead to unnecessary time consumption to find the path from a source to a target. Simple Crawler showed that it is possible to compute an influence value for a node. Monte Carlo Crawler showed that it is possible to compute this value in a reasonable amount of time. From the two previous approaches, we can conclude the goal for our third approach should be finding a navigation algorithm that (i) should not go back or expand in the wrong direction, (ii) should be computable in a reasonable amount of time and (iii) should hop using influential nodes of the network. Authors in [9] showed that mobility measures alone yield surprising predictive power, comparable to traditional network and similarity between two individuals' movements strongly correlates with their proximity in the social network. We use some quantities which have been proven to perform reasonably well in previous studies [9].

Computing the Influence Value (IV) of a node N_i in a path P_i from source node S to target node T is defined as a function with multiple parameters:

$$IV(N_i) = \lambda_1 a + \lambda_2 b + \lambda_3 c + \lambda_4 d + \lambda_5 e + \lambda_6 f \quad (2)$$

where each λ_i is a predefined weight parameter and (a) The number of direct friends (DF) of N_i: with V the set of vertices and E the set of edges of the network, direct friends of a node N_i are the friends that are reachable with a path of length one. They are defined by equation (3):

$$DF(N_i) = \{f | f \in V \text{ and } (N_i, f) \in E\} \quad (3)$$

(b) Shared neighbors (SN): If N_i and T share a direct friend, then there is a path of length 2 going from N_i to T. Common neighbors are defined in equation (4):

$$SN(N_i, T) = \{f | f \in DF(N_i) \text{ and } f \in DF(T)\} \quad (4)$$

(c) The number of attributes in common.

(d) The number of unique attributes.

(e) Distance to T: The distance between N_i and T is computed using attributes of the nodes. The more attribute they share, the smaller their distance is. See equation 5. Here J is the set of available attributes in OSN and $A_j(N_i)$ is the attribute j of

node N_i. The distance function can be simplified as in equation (5):

$$dist(N_i, T) = |J| - \sum_{j \in J} bool(A_j(N_i) == A_j(T_i)) \quad (5)$$

(f) The already crawled path: this is a list of nodes that have already been visited by the algorithm.

Our Influence Crawler algorithm is defined in Algorithm 1. Each time from the source node, we choose the highest influential friend and then consider this friend as the next hop until the target is reached.

Algorithm 1 Influence Crawler

```
Input: N_i, S, T
path = ∅
current = S
while current ∉ DF(T) do
    f* = friend of current with maximum influence
    if IV(f*) > 0 then
        add f* to path
        current = f*
    else                                  ▷ Dead end case
        Failure
    end if
end while
```

As our Influence Crawler is executed, multiple issues have to be resolved. In the early stage of routing, the distance between N_i and T decreases rapidly until we are in an virtual area where nodes are highly similar. People that share a lot are usually connected. Because the diameter of the Facebook graph is around 6 [8], we can consider that N_i has a high probability of having friends in common with T once our algorithm has run for 4 to 5 iterations. At this point, we can use the set of shared neighbors between N_i and T to help our algorithm to select the next hop more efficiently. The similarity between N_i and T defined in (c) is also a measure of relative proximity. In Online Social Networks, we are generally connected to people that look like us. Hence the more similar N_i and T are, the higher chance there is a short path between them.

Our influence values should be higher for nodes that are in the direction of the target. The already crawled path is passed as an argument of our influence function defined in equation (2). This allows our algorithm to select those nodes that are close to the source. The crawler can now return the next hop more efficiently by knowing what are the previous hops.

The correct direction of spreading can be defined as the direction that returns one of the shortest path between S and T, but not necessarily the shortest as it is impossible to know the shortest path without exploring the whole graph. The next hop should be chosen carefully because with our "can't go back" feature, we can't risk our algorithm to go in a dead-end direction. This can be avoided by making a compromise between two factors: (i) the next hop should be closer to the target and (ii) the next hop should be as connected as possible. If we favor (i) we are at risk to go straight and found ourselves in an impasse. If we favor (ii), we risk to found ourselves with an inefficient algorithm that is running in circle.

Algorithm 2 Influence Value

```
function IV(N_i)
    if N_i ∈ path then                          ▷ uses (f)
        influ = 0
    else if N_i ∈ DF(T) then                    ▷ uses (b)
        influ = K
    else if N_i ∈ DF(DF(T)) then                ▷ uses (b)
        influ = L
    else                                        ▷ uses (a)(c)(d)(e)
        influ = M − α * Dist(N_i, T) + β * |DF(N_i)|
    end if
    return influ
end function
```

with $K > L > M$

4.4 Simulation Results

The following table compares our Influence Crawler with the Targeted Crawler. Data have been extracted by running the two algorithms on the same random source and target nodes. We can see that our Influence Crawler founds the target node in a less number of iterations. However it is also possible that our Influence Crawler doesn't find a path when one exists. Our Influence Crawler stops in two cases:

(i) it reaches a dead-end in the graph because the algorithm made bad choices during its execution or (ii) we arbitrarily stop it because it takes too long comparing to the Targeted Crawler and our goal is to make an algorithm that is more efficient.

	Targ. Craw.	Influ. Craw.	Diff.
# iterations	17.746	2.92	-83.5%
Path length	2.495	2.436	-2.4%
Time (ms)	1.303	0.393	-69.8%
Success rate	80.4%	51.6 %	-28.8%

Table 2: Average results for 1000 iterations of the crawlers.

	Targ. Craw.	Influ. Craw.	Diff.
# iterations	19.327	2.754	-85.8%
Path length	3.103	2.754	-11.3%

Table 3: Success only

	Targ. Craw. + Infl. Craw.	Diff.
# iterations	14.243	-10.2%
Time (ms)	1.249	-4%
Success rate	80.4%	0%

Table 4: Average results for 1000 iterations of the Influence Crawler followed by the Targeted Crawler if failure.

Influence Crawler is constructed such as it adds a node to the general path at each iteration. Therefore the length of the returned path will always be equal to the number of iterations of the algorithm whereas in the Targeted Crawler, the length of the path is usually much smaller than the number of iterations.

Considering results in tables 2, 3 and 4we have a success rate of 50% with our Influence Crawler for 2.4 iterations in average. This means our Crawler fails 50% of the time. The number of iteration for a fail case is at most 8. In the case our Influence Crawler fails, we decided to run the Targeted Crawler to find a path instead. With this in mind, we can compute the average number of iteration needed to find a path between two nodes. This means the combination of the Targeted Crawler and our Influence Crawler runs in average 10% less iterations and has 50% chance of finding a path in 83% less iterations.

5 Conclusion and Future Work

We hope to develop a network perspective as a powerful way of looking at complex systems in general and a way of thinking about social dynamics, internal structure and feedback effects of the social networks.

References

[1] Lada A. Adamic and Eytan Adar, "Friends and neighbors on the web," *ScienceDirect*, 2003.

[2] Stephen Borgatti, "Identifying sets of key players in a social network," *Computational and Mathematical Organization Theory*, 2006.

[3] Stanford Large Network Dataset Collection.

[4] D. Easley and J. Kleinberg, *Networks Crowds and Markets*. Cambridge, 2010.

[5] Mathias Humbert, Theophile Studer, Matthias Grossglauser, and Jean Pierre Hubaux, "Nowhere to hide: Navigation around privacy in online in social networks," *The 18th European Symposium on Research in Computer Security (ESORICS)*, 2013.

[6] Muhammad U. Ilyas and Hayder Radha, "Identifying influential nodes in online social networks using principal component centrality," *IEEE International Conference*, 2011.

[7] Pedram Pedarsani, "Privacy and dynamics of social networks," *École Polytechnique Fédérale de Lausanne*, 2013.

[8] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow, "The anatomy of the facebook social graph," *Cornell University*, 2011.

[9] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi, "Human mobility, social ties and link prediction," *KDD Conference*, 2011.