Influence Maximization in Messenger-based Social Networks

Yuanxing Zhang[1], Yichong Bai[1], Lin Chen[2], Kaigui Bian[1], Xiaoming Li[1]
[1]Peking University, [2]Yale University

***Abstract*—Many online social networks have provided a messenger app (e.g., facebook messenger, direct message on Twitter) to facilitate communication between strong-tied friends. Meanwhile, some messenger apps (WeChat) also start to offer social-networking services ("WeChat Moments" (WM), a.k.a. friend circle[1]) that allow users to post pictures, texts, links of webpages, on their walls, which is called the *messenger-based social network (Msg-SN)*. In online social networks, Key Opinion Leaders (KOLs) with millions of followers are easy to identify for helping viral marketing/advertising. However, most users of a messenger app have a small number of friends (e.g., hundreds of friends), which makes it challenging to detect a KOL in Msg-SN by only counting the number of his/her friends. In this paper, we study the influence maximization problem in the Msg-SN of finding the set of most influential KOL nodes that maximize the spread of information. We develop a novel efficient heuristic algorithm that calculates the influence by looking at the user's local contribution to the information diffusion process, which scales to large datasets with provable near-optimal performance. Experiment results using the real-world WeChat Moments data (on January 14th, 2016, 100 thousand users) show that our algorithms can identify the set of KOL nodes with a low time complexity.**

I. Introduction

The interpersonal tie in social networks is defined as the information-carrying connection between people [5]. It has been implemented as many Internet applications such as online social networks (OSN), and messenger apps.

- **OSN+messenger**. People with strong (interpersonal) ties are no longer satisfied with the current connection means in conventional online social networks, and they are more likely to communicate via a messenger app (e.g., facebook messenger, direct message on Twitter).
- **Messenger+OSN**. Meanwhile, some messenger apps (WeChat) has started to provide social-networking services in the hope of strengthening interpersonal ties between users. For instance, "WeChat Moments" (WM), a.k.a. friend circle allows users to post pictures, texts, links of webpages, on their walls.

We call the social network derived from messenger apps as the *messenger-based social network (Msg-SN)*.

In conventional social networks, Key Opinion Leaders (KOLs) with millions of followers (or a large number of friends) are easy to identify for helping spread information for viral marketing/advertising [3]. The information overload problem may arise since there is no any restriction that prevents KOLs from connecting to many friends and delivering excessive information to them, which has undermined the user experience of many OSNs applications [2].

However, it is challenging to detect a KOL in Msg-SN by counting the number of his/her friends, because the social networking service is derived from the messenger app where most users only have a limited number of friends (approximately hundreds of friends). Moreover, WM as a Msg-SN, differs from the conventional OSNs by imposing restrictions on the information diffusion process [10]: (1) Each user is limited to have no more than 5000 friends; (2) a user cannot see or access posts of non-friend users who are not in the contact list of the messenger app. These restrictions confine the information diffusion within a community of acquaintances, which mitigates the information overload problem. In contrast, on Twitter or Weibo, a user has no upper bound on the number of followers, and one can access others' posts by following the re-tweeting paths.

In this paper, we study the KOL detection problem in Msg-SN, which we formulate as the influence maximization problem of finding the set of most influential KOL nodes that maximize the spread of information. To address this problem, we present a strategy that selects influential nodes by looking at the user's local contribution to the information diffusion process, which is different from existing greedy solutions using Lazy Evaluation strategies that seek nodes with a large marginal influence [6]. The contributions of this work are summarized as follows.

1) *Establishment of the information diffusion model in WM*: Instead of counting the number of friends, we define one's influence by the range of his posts can spread, and the number of clicks of his posts.
2) *Design of Heuristic algorithms*: We develop a novel heuristic algorithm that calculates the influence by looking at the user's local contribution to the information diffusion process, which scales to large datasets with provable near-optimal performance.
3) *Experiments using large real-world WM datasets*: Using the real-world WM data (on January 14th, 2016, 100 thousand users), experiment results show that our algorithms can identify the set of KOL nodes with a small overhead.

The rest of this paper is organized as follows. We review the related work in Section II. We formally formulate the problem in Section III. In Section IV, we discuss the greedy algorithm with Lazy Evaluation and present our heuristic algorithms. We evaluate the performance of heuristic algorithms using real-world data experiments in Section V. Finally, we conclude with Section VI.

[1]WeChat is a mobile messaging app that has 650 million active users as of Q1 2016; and "WeChat Moments" serves social-networking functions. WeChat has a PC client, while the WM is only available on its mobile client.

978-1-5090-1328-9/16/$31.00 ©2016 IEEE

II. Related Work

In conventional OSNs, the information diffuses at different transmission rates across different links in the network under certain diffusion models.

Diffusion model. Given the underlying network graph, a diffusion model can represent the transmission rate of the information over an directed edge from node u to node v. Users in the network may have two states, either *infected* or *susceptible*, and note that a susceptible user may get infected but an infected user will not change to be susceptible. The messages tend to transfer from the infected users to those uninfected, until no more users would change their states. There are two basic diffusion models [6]:

- *Linear Threshold (LT) Model*. Each user has an immunity from being infected, and he change his state only when the total effect on him exceeds the immunity.
- *Independent Cascade (IC) Model*. The user u being infected at time t would try to infect each of his susceptible friend v at time $t+1$, succeeding with probability $w_{e=(u,v)}$, which is a system parameter independent from the history. At other time, the state of user u remains.

Influence maximization problem and algorithms. The influence maximization problem requires to find up to K users, leading to a maximum influence (i.e., the total number of nodes that received the information) under a certain diffusion model. The problem is proved to be NP-hard [6], and certain approximate algorithms are required to address the problem.

Kempe et al. [6] proposed a greedy algorithm to do the approximation for influence maximization, promising to have a performance better than 63% of the optimum. In contrast to approximation methods, Chen et al. [1] used a heuristic method of the longest diffusion path and introduced the prefix excluding maximum influence arborescence (PMIA) model, navigating the subsequent scalable algorithm. Besides, Tang et. al. [9] come up with Influence Maximization via Martingales (IMM) model implementing reverse reachable set and provide a theoretical bound of approximation.

III. Problem Formulation

In this section, we formally define the information diffusion based on the IC model as well as one's influence in the Msg-SN. Then, we formulate the KOL detection problem as an influence maximization problem.

A. Msg-SN Applications

In the Msg-SN, the posts shared by users are HTML5 (H5) pages. Since the H5 page supports interactive operations for users, it can be an advertisement, online greeting card, a lightweight online game (e.g., flappy birds), psychological test, etc. Hence, an H5 page that can be shared over the network is also called *a Msg-SN application*. A Msg-SN application can be launched by the service provider (e.g., the WM service provider is Tencent), or a third-party H5 developer.

Suppose that users u and v are friends in the network. When user u shares an application link with his friends, user v needs to click the link first to view the post content; and then to click the hidden share button to share the post if user v finds the post worth reposting. Since Msg-SN applications are viewed and shared via clicking actions, we define a click-based diffusion model below.

B. Click-based Diffusion Model

Application diffusion graph. The overall structure of the Msg-SN can be characterized by an underlying network graph $G = (V, E)$, where V stands for the set of all users and $E \subset V \times V$ represents the diffusion edges. We denote $N = |V|$ and $M = |E|$. Here, we implement the thought of IC model, meaning that there is a probability value w_e on each edge e. Many applications are diffused over G, which we can assume that all applications are independently diffused in G.

Definition 1. *In the Msg-SN, regarding an application α, an information diffusion (application transmission) process from user u to v completes, if and only if user v views the content of application α transmitted from user u, and reposts the link of α, where α is the application ID.*

Apparently, the application transmission process from user u to v consists of two events.

1) *App view event*: user v views the content of application a transmitted from user u. We say that user v and the corresponding node v in G are *infected* by application α, when user v views application α *for the first time*; meanwhile, we define the "infected time" of user v (and the corresponding node v in G) for application a as the time user v *first* views this application.
2) *App share event*: after viewing the application, user v decides to share the link of α to his friends by reposting it. We define the "share time" of user v for application α as the time he first shares the link of this application.

It is feasible that a user clicks to view and share an application for multiple times. Thus, we need to combine the parallel edges and embody them on the probability thresholds.

Direct influence function: We define a function that describes the direct influence that a node exerts over its one-hop neighbor nodes, i.e. the probability value. Applications may have different effect between users sharing and reading because of topic, background music, etc. Denote ϵ as the past effect that can be calculated by the weighted sum of viewing times of different applications. Specially, as we do not differentiate each application in this problem, we can simply make ϵ as the total number of clicks by which a user views the applications transmitted from another specific user. Then, given an edge (u, v), let $f_{u,v}(\epsilon)$ denote the direct influence from node u to node v in the diffusion graph. Apparently, the direct influence function $f_{u,v}(\epsilon)$ needs to satisfy the following properties.

1) $f_{u,v}(0) = 0$ when there is no edge from u to v in G;
2) $f_{u,v}(\epsilon)$ is monotonically increasing as ϵ increases ($\frac{\mathrm{d}f_{u,v}(\epsilon)}{\mathrm{d}\epsilon} > 0$); a large ϵ implies that user v views applications from user u by many times;
3) $f_{u,v}(\epsilon)$ is bounded.

Group influence: Now we have a probability value on each edge, which is the direct influence of a user on one of his friends. The indirect influence from node u to node v is contributed by a set of direct influence values along the

diffusion path from node u to v. Let $\mathcal{R}_{u,v}(G)$ denote the set of diffusion paths from u to v.

Note that there may be more than one path from u to v, which may cause the exponential increasing of calculation. Thus, the total influence over paths from node u to node v, $p_p(u,v)$, can be written as:

$$p_p(u,v) = \max_{r' \subset \mathcal{R}_{u,v}(G)} \{P(r')\}, \tag{1}$$

where

$$P(r') = \prod_{e \in r'} w_e.$$

If we consider the influence on v by a group of users S, it can be seen as each of the user $u \in S$ trying to infect v. However, those maximal diffusion paths from $\forall u \in S$ to v may overlap. Therefore, we assign $p_p(S,v)$ by the maximum of influence from $u \in S$:

$$p_p(S,v) = \max_{u \in S} \{p_p(u,v)\}. \tag{2}$$

C. Influence Maximization Problem in WM

The KOL detection problem requires to find a set S of users, from whom we can get the largest set of infected users, namely $I(S)$. We define the influence function of S as:

$$\sigma(S) = |S| + \mathcal{E}(I(S)), \tag{3}$$

where $\mathcal{E}(I(S))$ means the expectation of the size of infected users caused by S. Our goal is to find the top-K most influential users in the Msg-SN, which is equivalent to identifying a set of nodes that have the maximal influence over all application diffusion graphs G in the influence maximization problem.

Problem 1. *Given a message diffusion graph $G = (V, E)$, we are interested in finding the top-K most influential users (i.e. the top-K most influential nodes in V) that can collectively cause the greatest $\sigma(S)$.*

$$\begin{aligned} &\textit{Maximize} \quad \sigma(S) \\ &\textit{subject to} \quad S \subseteq V, |S| \leq K. \end{aligned}$$

It has been proven that the influence maximization problem is #P-hard [1]; meanwhile, the influence function is monotonous and submodular [6], and thus greedy algorithms based on Lazy Evaluation have been used to address the problem.

IV. Efficient Heuristic Algorithms

In this section, we first present a greedy algorithm accelerated by the Lazy Evaluation technique (which is similar to existing approaches), and identify its limitations. Then, we propose an heuristic strategy to achieve a better performance.

A. The Greedy Algorithm

The greedy algorithm considers the marginal benefit $\Delta(u|S')$, which is the temporal influence of each user u when a set of seed users S' has already been chosen. Each time, pick

$$u^* = \arg \max_{u \in V \setminus S'} \Delta(u|S')$$

and update $S' = S' \cup \{u^*\}$ until $|S'| = K$. The original state is

$$\Delta(u|\varnothing) = \sum_{v \in V} p_p(u,v).$$

It is not wise to scan every node in $V \setminus S'$ and choose the best one for every iteration, leading to repetitive computations. Obviously, if node u leads to a small value of $\Delta(u|S')$ in every iteration, it may never be selected as one of the top-K influential nodes. Therefore, we could maintain a priority queue that can accelerate the finding of the most influential nodes in the current state. In detail, we define $\mathcal{P}(u)$, the *potential* value of $\Delta(u|S')$ for each node u, which is an estimated upper bound of $\Delta(u|S')$. Note that $\mathcal{P}(u) \geq \Delta(u|S')$ in each iteration, and the initial value of $\mathcal{P}(u) = \Delta(u|\varnothing)$. Besides, it is necessary to maintain an infection probability table (IPT), representing the infected rate. Thus, we have

$$\forall u \in S, \text{IPT}(u) = 1,$$

and

$$\sigma(S') = \sum_{u \in V} \text{IPT}(u),$$

$$\Delta(u|S') = \sum_{v \in V} \left[p_p(S' \cup \{u\}, v) - p_p(S', v) \right].$$

Time complexity. With the help of the priority queue, a large amount of repetitive calculations of $\Delta(u|S')$ are avoided. Updating a node u's influence $\Delta(u|S')$ consumes $O(M)$ time, and thus the total time complexity is $O(KN(M+\log N))$. It is a loose bound, where the worst case occurs in a near complete graph, which rarely appears in the real world datasets.

Limitations. Existing greedy algorithm chooses a node as a KOL if the node has the greatest marginal benefit in the current iteration when executing the algorithm, which ignores the contribution to the information diffusion by nodes that have not been selected as KOL nodes. However, in Msg-SN, most nodes have a similar node degree value, and the collective contribution of two nodes to the information diffusion may be more than that of two influential nodes selected by the greedy algorithm. For example, consider an information diffusion tree formed by the diffusion paths of a Msg-SN application in Figure 1, where the probability threshold on each edge is 0.5, and we want to solve the top-2 KOL selection problem in this diffusion tree.

The conventional greedy algorithm tends to choose node #1 first as $\Delta(1|\varnothing) > \Delta(2|\varnothing)$, and then it chooses node #2 or #3. The resulting value in the influence function in (3) is $\sigma(\{1,2\}) = \sigma(\{1,3\}) = 5.5$. However, the optimal solution for top-2 KOL problem is a set of nodes #2 and #3, since $\sigma(\{2,3\}) = 6$. In other words, the set of KOL nodes in $\{2,3\}$ have a higher probability of diffusing the information to all nodes in the tree, than the nodes chosen by the greedy algorithm (nodes in $\{1,2\}$ or $\{1,3\}$).

The reason for this result is that the greedy algorithm selects KOL nodes individually, which does not take into account the cooperation (or collective contribution) among the candidate KOL nodes to the information diffusion.

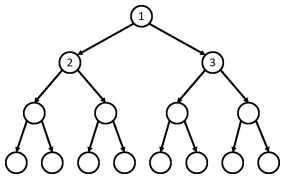


Fig. 1: An example showing the limitations of the greedy algorithm.

B. The Voting Algorithm

We define the diffusion state graph to study the local contribution of nodes to the information diffusion process.

Definition 2. *A* diffusion state graph $G' = (V', E')$ is a subgraph of $G = (V, E)$, where $V = V'$ and $E' \subset E$. Whether an edge $e \in E$ exists in E' follows Bernoulli Distribution of w_e. The probability of a diffusion state G' is calculated as

$$P(G' = (V', E')) = \prod_{e \in E'} w_e \prod_{e \notin E'} (1 - w_e).$$

Extended main diffusion tree (EMDT) in the diffusion state graph. In our clicked-based diffusion model, an infected user will never go back to the uninfected state. Thus, the effective diffusions form a DAG. Moreover, the paths with the greatest influences play the dominant role in the diffusion process, forming the main diffusion forest (MDF)—for every weak connected block, find its maximum product directed spanning tree:

$$\text{MDF}(G') = \arg \max_{\substack{E'' \subset E' \\ E'' \text{ forms a spanning forest}}} \prod_{e \in E''} w_e.$$

We can change the formation equivalently to the minimum directed spanning tree by logarithm:

$$\text{MDF}(G') = \arg \min_{\substack{E'' \subset E' \\ E'' \text{ forms a spanning forest}}} \sum_{e \in E''} -\log(Z w_e),$$

where Z is a large constant to make $\forall e \in \text{MDF}(G'), Z w_e > 1$ so that the values promise to be positive after logarithm. Note that the spanning tree will not change if we amplify parameters on each edge equally. Then we consider to merge the forest to a tree. Obviously, if we get all spanning trees, we can set a super root node $\widetilde{u}$ and connect it to every spanning tree's root and the forest is converted to a tree. $\widetilde{u}$ is useful in understanding the diffusion process, as applications can be seen as being posted from it and transferred to each spanning tree through their root nodes. Although we can not appoint those root nodes in advance or randomly, we find that the value $-\log(Z w_e)$ on each edge in the minimum spanning tree is negative. As a result, some extra edges whose values are zero will not change the final structure of the spanning tree. Thus, the following steps are introduced to calculate the merged spanning tree, named as *Extended Main Diffusion Tree (EMDT)*:

1) Set a super root node $\widetilde{u}$;
2) Do negative logarithm to every edge in E';
3) Add new edges from $\widetilde{u}$ to every node with value as zero;
4) Find the minimum arborescence $\mathcal{T}$, which is a set of edges;
5) On $\mathcal{T}$, recover the values on old edges but replace the value on new edges by one.

Importance of a node in the diffusion process based on EMDT. With the help of EMDT(G'), we can easily compute the *importance* of each node u when it serves as a KOL for diffusing information in the network.

Intuitively, the importance of a node u in the information diffusion process, namely $Q(u|G')$, can be calculated by the nodes that are infected by information diffused from node u. Say a pair of nodes x and y have viewed the same application diffused from u, and it means that the lowest common ancestor of nodes x and y in EMDT(G'), denoted by $LCA(x, y)$, is exactly u. However, there is a special case that nodes x and y are located in different weak connected blocks of G', and thus $LCA(x, y) = \widetilde{u}$.

Let $OA(x)$ denote the oldest ancestor of x in EMDT(G'). The importance of user u becomes:

$$\begin{aligned} Q(u|G') = & \sum_{\substack{x,\, y \,\in\, V' \\ LCA(x,\, y) \,=\, u}} p_p(u, x) p_p(u, y) \\ + & \sum_{\substack{x,\, y \,\in\, V' \\ OA(x) \,=\, u \text{ or } OA(y) \,=\, u \\ LCA(x,\, y) \,=\, \widetilde{u}}} p_p(OA(x), x) p_p(OA(y), y). \end{aligned} \tag{4}$$

where

$$p_p(u, v) = \prod_{e \in \widetilde{r}' \in \mathcal{R}_{u,v}(\widetilde{G}' = (V \cup \{\widetilde{u}\}, \mathcal{T}))} w_e.$$

Note that there is exactly one path in $\mathcal{R}_{u,v}(\widetilde{G}')$. Then, K users with greatest $Q(\cdot|G')$ are nominated as the top-K KOL nodes of G', i.e.

$$\tau(u|G') = \begin{cases} 1 & Q(u|G') \text{ ranks in top-}K \\ 0 & Q(u|G') \text{ does not rank in top-}K \end{cases} \tag{5}$$

Finally, the importance of u in the diffusion process is calculated as

$$\beta(u) = \sum_{G'} P(G') \tau(u|G'), \tag{6}$$

and K nodes with greatest $\beta(\cdot)$ values are recommended as KOL nodes.

Gibbs Sampling to reduce the time complexity. The calculation of $\beta(\cdot)$ values is expensive, as there are 2^M different diffusion state graphs in total. In implementation, we utilize the Gibbs Sampling to reduce the computational cost. Let s_i be the state of the i-th edge in E, namely e_i; assign 1 to e_i for existing in a diffusion state; and assign 0 to e_i for non-existing state. Then the diffusion state graphs are sampled in the sequence of

$$G'^{(t+1)} \Leftrightarrow MC(s_1^{(t+1)}, \ldots, s_M^{(t+1)})$$

subject to

$$P(s_i^{(t+1)}) = P(s_i | s_1^{(t+1)}, \ldots, s_{i-1}^{(t+1)}, s_{i+1}^{(t)}, \ldots, s_M^{(t)}) = w_{e_i},$$

where $MC(s_1^{(t)}, \ldots, s_M^{(t)})$ is the state corresponding to $G'^{(t)}$.

Similarly, it is expensive to enumerate every pair of users for calculating the importance value of their ancestors (a number of $O(N^2)$ different pairs). Thus, we only check the two users

involved in a pair and change them with uniform distribution in the next iteration.

Besides, Gabow et.al. [4] have proposed with a method calculating the minimum arborescence in $O(N \log N + M)$ time with the help of Fibonacci stack, while we could implement Heavy Path Decomposition [8] to accelerate single query on the product of $\widetilde{pp}(LCA(x,y),x)$ multiplied by $\widetilde{pp}(LCA(x,y),y)$ from $O(N)$ to $O(\log^2 N)$.

Therefore, if we do the Gibbs Sampling on diffusion state graphs for R_1 times, and on enumerating pairs of users for R_2 times in each iteration, the total time complexity is then reduced to $O(R_1(N \log N + M + R_2 \log^2 N + N \log K))$. The whole procedure of the voting algorithm is given in Algorithm 1.

Algorithm 1 A voting algorithm for calculating the importance of a node in information diffusion.

Input: G, K
Output: the target set S of most influential nodes.

for each $u \in V$ **do**
  $\beta(u) = 0$.
end for
for $i = 1$ to R_1 **do**
  Get diffusion state G' with Gibbs Sampling;
  $\mathcal{T} = \text{EMDT}(G')$;
  Construct the heavy paths and light paths of $\mathcal{T}$;
  for each $u \in V$ **do**
    $Q(u|G') = 0$;
  end for
  for $j = 1$ to R_2 **do**
    Select x and y by Gibbs Sampling;
    $\psi = LCA(x,y)$, $\kappa = \widetilde{p}_p(\psi,x)\widetilde{p}_p(\psi,y)$;
    if $\psi == \widetilde{u}$ **then**
      $Q(OA(x)|G') += \kappa$;
      $Q(OA(y)|G') += \kappa$;
    else
      $Q(\psi|G') += \kappa$;
    end if
  end for
  for each $u \in V$ **do**
    Compute $\tau(u|G')$;
    Update $\beta(u)$;
  end for
end for
Sort β and gather the top-K nodes into set S;
return S.

V. Performance Evaluation

A. Experiment Setup

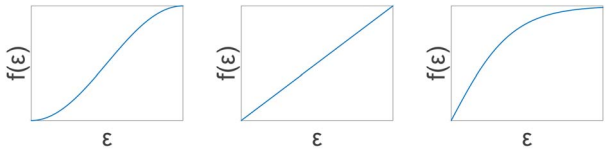
In this section, we evaluate the performance of our Voting algorithm in detecting KOL nodes in Msg-SN, by experiments over the real-world WM dataset.

WM Dataset. The WM dataset used in our experiments is provided by a data analysis company that focuses on mobile H5 page/application spreading analysis. We use 500 applications created by businesses on January 14th, 2016. Regarding these chosen applications, there are a total number of 2.2 million views recorded. After constructing diffusion graphs for the chosen applications, there are about 200 thousand edges, and approximately 100 thousand users involved.

Compared algorithms. We compare the following four algorithms by experiments.

1) VOTING: We will test the effect of each parameters of this proposed algorithm (R_1 and R_2) and compare it to the following three algorithms on the influence $\sigma(S)$ and the consuming time T.
2) GREEDY: The greedy algorithm using Lazy Evaluation is evaluated in terms of its capability of estimating the influence $\sigma(S)$.
3) K-LARGEST: This algorithm simply chooses K users with greatest individual influence.
4) DEGREE: This algorithm just chooses K users with greatest out-degree.

B. The Impact of Direct Influence Functions



(a) func 1 (b) func 2 (c) func 3

Fig. 2: Three type of functions.

The influence of a set of nodes is built upon the direct influence function $f_{u,v}(\epsilon)$. In the WM datasets, all applications are independently and diffused equally over edges in G, and thus we use the same $f_{u,v}(\epsilon)$ for all infected edges of G in the experiment. In the definition, we do not recommend any distribution for $f_{u,v}(\epsilon)$. Here, we will investigate three different forms of the function $f_{u,v}(\epsilon)$, as shown in Fig.2, and analyze whether they will lead to different results of the influence $\sigma(S)$. Since we cannot directly compare the difference of two functions, we choose to compare two target sets S_{f_1} and S_{f_2} under $f_{1\ u,v}(\epsilon)$ and $f_{2\ u,v}(\epsilon)$ respectively by Jaccard distance [7].

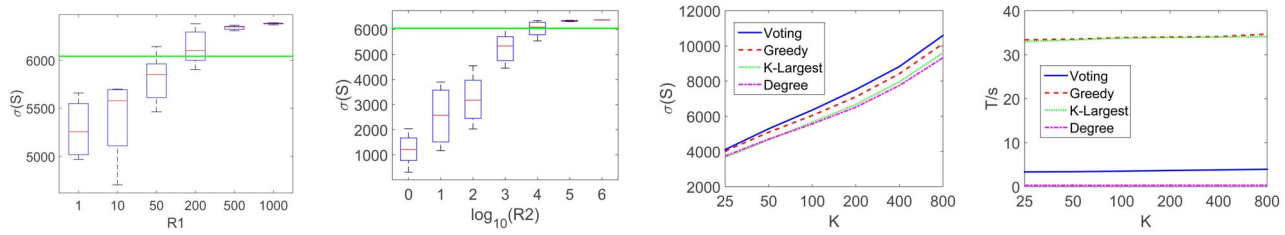
From the results listed in Table I, we are able to observe that any two distribution functions lead to very similar target sets, although the two functions differ largely from each other. It is found that the two sets differ from each other by no more than eight nodes over 100 KOL nodes. Hence, we choose the distribution as func 1 for every $f_{u,v}(\epsilon)$ in the next set of experiments.

TABLE I: Comparison of different functions of $f_{u,v}(\epsilon)$.

	func 1	func 2	func 3
func 1	1.00	0.90	0.92
func 2	0.90	1.00	0.85
func 3	0.92	0.85	1.00

C. The Impact of Algorithm Parameters

Varying the number of sampled diffusion state graphs, R_1. Figure 3a shows the value range of the influence of target



(a) $\sigma(S)$ vs. R_1; (b) $\sigma(S)$ vs. R_2. (c) $\sigma(S)$ vs. K. (d) Time vs. K.

Fig. 3: The experiment results: (a) The value range of $\sigma(S)$ with variant R_1 over 10 experiments; (b) The value range of $\sigma(S)$ with variant R_2 over 10 experiments; (c) The tendency of $\sigma(S)$ of each algorithm with variant K; (d) The tendency of consuming time of each algorithm with variant K. The green line is the performance of greedy algorithm

KOL sets under our Voting algorithm by varying R_1, while the other parameters are set as $K = 100, R_2 = 100000$. The green line is the performance of greedy algorithm when $K = 100$. According to our definition of Voting algorithm, in every iteration we choose a diffusion state graph by Gibbs Sampling. This leads to the fact that the more we sample, the more accuracy we have in the estimation of the influence of each node, and then a set of nodes with greater influence can be selected, which can be seen clearly in the figure. Besides, we observe that as R_1 grows, the variance of $\sigma(S)$ increases initially, and then eventually starts decreasing in an inverted U shape until $\sigma(S)$ reaches a stable value. The influence value of target set under Voting algorithm is greater than that under the greedy algorithm, when R_1 is greater than 200. As the fluctuation becomes relatively small when R_1 achieves 500, we fix R_1 as this value in the following experiments.

Varying the number of enumerated users pairs, R_2. As R_2 increases, more pairs of users are enumerated, thereby leading to a more accuracy evaluation of a single EMDT(G'). Let $K = 100, R_1 = 500$. Figure 3b shows the value range of the influence of nodes in the target KOL set, with R_2 growing from 10^0 to 10^6. The $\sigma(S)$ raises as R_2 increasing, while the variance of it implies that the enumeration is quite random if R_2 is small; otherwise a larger R_2 will make the $\sigma(S)$ more stable. Similar to results in Figure 3a, the influence value of target set under Voting algorithm is greater than that under the greedy algorithm, when $\log_{10}(R_2)$ is greater than 4.

D. The Performance of the KOL Detection Algorithms

Here, we compare the performances of the four KOL Detection Algorithms with variant K, where $R_1 = 500$ and $R_2 = 10^5$. For every K, we implement these algorithms and calculate each $\sigma(S)$, as shown in Figure 3c. It is obvious that our Voting algorithm performs the best, while the K-Largest and degree algorithm are the worst. This result verifies that choosing the nodes with greatest out-degree will lose considerable influential nodes. In terms of running time, the greedy algorithm consumes nearly same as the K-Largest algorithm, as the process using priority queue does not execute as many as the upper bound times. However, our proposed algorithm requires less time, for the reason that there is no initial step for calculating each $\Delta(u|\varnothing)$ that consumes $O(NM)$ time. These results are displayed in Figure 3d.

VI. Conclusion

"WeChat Moments" (WM) is an messenger-based social network (Msg-SN), which mitigates the information overload by limiting a user's maximum number of friends, access permissions, and re-tweeting actions. In this paper, we studied the influence maximization problem in Msg-SNs of finding the set of most influential KOL nodes that maximize the spread of information, using the real-world large datasets in WM networks. We proposed a click-based diffusion model and developed a voting algorithm using Gibbs Sampling, Minimum Arborescence and Heavy Path Decomposition, which can scale to large datasets with provable near-optimal performance. Experiment results showed that our algorithm outperform the K-Largest, degree and greedy algorithms in detecting the set of KOL nodes and saving the time overhead.

Acknowledgements

This work was sponsored by National Natural Science Foundation of China (NSFC) under grant number 61572051.

References

[1] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010.

[2] danah m. boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[3] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66. ACM, 2001.

[4] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[5] M. Granovetter. Economic action and social structure: The problem of embeddedness. *American journal of sociology*, pages 481–510, 1985.

[6] D. Kempe, J. Kleinberg, and Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[7] L. Michael and W. David. Distance between sets. *nature*, 234:34–35, 1971.

[8] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *Journal of computer and system sciences*, 26(3):362–391, 1983.

[9] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: a martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1539–1554. ACM, 2015.

[10] J. Wu. How wechat, the most popular social network in china, cultivates wellbeing. *University of Pennsylvania*, 2014.