Impact of Online Activities on Influence Maximization: A Random Walk Approach

Pengpeng Zhao[†], Yongkun Li[†], Hong Xie[‡], Zhiyong Wu[†]
Yinlong Xu[†], Richard T. B. Ma[‡], John C. S. Lui[§]
[†]University of Science and Technology of China
[‡]National University of Singapore
[§]The Chinese University of Hong Kong
{roczhau, wzylucky}@mail.ustc.edu.cn, {ykli, ylxu}@ustc.edu.cn
{hongx87, tianbai.ma, johncslui}@gmail.com

ABSTRACT

With the popularity of OSNs, finding a set of most influential users (or nodes) so as to trigger the largest influence cascade is of significance. For example, companies may take advantage of the "word-of-mouth" effect to trigger a large cascade of purchases by offering free samples/discounts to those most influential users. This task is usually modeled as an influence maximization problem, and it has been widely studied in the past decade. However, considering that users in OSNs may participate in various kinds of online activities, e.g., giving ratings to products, joining discussion groups, etc., influence diffusion through online activities becomes even more significant. Thus, it necessitates to revisit the influence maximization problem by taking online activities into consideration.

In this paper, we study the impact of online activities by formulating the influence maximization problem for social-activity networks (SANs) containing both users and online activities. To address the computation challenge, we define an influence centrality via random walks on hypergraph to measure influence, then use the Monte Carlo framework to efficiently estimate the centrality in SANs, and use the Hoeffding inequality to bound the approximation error. Furthermore, we develop a greedy-based algorithm with two novel optimization techniques to find the most influential users. By conducting extensive experiments with real-world datasets, we show that compared to the state-of-the-art algorithm IMM [20], our approach improves the accuracy of selecting the top k most influential users in SANs by up to an order of magnitude, and our approach has a comparable computation time even though it needs to handle a large amount of online activities. Furthermore, our approach is also general enough to work under different diffusion models, e.g., IC and LT models, as well as heterogeneous systems, e.g., multiple types of users and online activities in SANs.

1. INTRODUCTION

Due to the popularity of online social networks (OSNs), viral marketing which exploits the "word-of-mouth" effect is of significance to companies which want to promote product sales. In particular, if a user in an OSN buys a product, she may influence her friends in the OSN to purchase the same product, and this influence may continue throughout the whole network in a cascade behavior. Therefore, it is of interest to find the best initial set of users so as to trigger the largest influence spread, i.e., to maximize the number of final buyers. This viral marking problem can be modeled as an influence maximization problem, which was first formulated by Kempe et al. [15]. That is, given an OSN and an information diffusion model, how to select a set of k users, which is called the seed set, so as to trigger the largest influence spread. This problem is proved to be a NP-hard problem [6,8], and it has received a large body of work in the past decade [6–8,20,21].

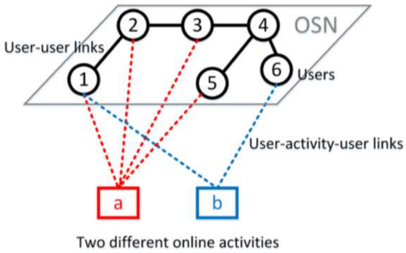


Figure 1: An example of social-activity network with six users and two online activities.

Considering that users in today's OSNs may participate in various kinds of online activities, e.g., forwarding friends' posts, joining a discussion group, and clicking `like` on a public page in Facebook, etc., so users not only can create friendship relatioships, which we call *user-user* links, but can also form relationships by participating in online activities, which we call *user-activity-user* links. For example, if two users in Facebook express like to the same public page, then they form a user-activity-user link no matter they are friends or not in the OSN. We call this kind of networks which contain both user-user relationships and user-activity-user relationships as *social-activity networks* (SANs). Other forms of SANs also include social rating systems which contain a social network and ratings on products given by users, as well as online social games which also consist of a social network among users and various kinds of online activities like joining a real-time battle team. Figure 1 shows a simple SAN which contains six users and two different online activities. In particular, the online activities a and b may represent two different products if the SAN denotes an online rating system, and users connecting to the same activity means that they give ratings to the product, e.g., links $(1, b)$ and $(6, b)$ denote that both user 1 and user 6 give a rating to product b.

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

With the consideration of online activities, the influence may also spread through the user-activity-user links besides the user-user links. For example, for the SAN in Figure 1, user 1 may also influence user 6, because they both give a rating, say high, to product b. Furthermore, users participating in the same online activity implies that they share the same interest, and as a result, influence diffusion through the user-activity-user links becomes even more significant, and so using the conventional influence maximization problem defined in OSNs alone may not trigger the largest influence spread. To further illustrate, consider the example in Figure 1, if we focus on the OSN only, then user 4 may be the most influential node by using the degree centrality to measure influence. If activities are further taken into account, user 1 may be the most influential node, because user 1 participated in online activities together with all users except for user 4, and so user 1 has the opportunity to directly influence other users through the user-activity-user links. Existing works on influence maximization (see §7 for review) usually focus on OSNs only and do not take the impact of online activities into consideration. *This motivates us to re-formulate the influence maximization problem for SANs, and find the most influential nodes by taking online activities into consideration*.

However, solving the influence maximization problem in SANs with online activities being considered is challenging. First, influence maximization in OSNs without online activities was already proved to be NP-hard, and considering online activities makes this problem even more complicated. Second, the amount of online activities in a SAN is very large even when the OSN is small, this is because online activities happen more frequently than friendship formation in OSNs. As a result, the underlying graph which characterizes users and their relationships may become extremely dense if we transform the user-activity-user links to user-user links, so it requires highly efficient algorithms for finding the most influential nodes. Lastly, users and online activities in a SAN may be classified as multiple types as different online activities may have different impacts on different users. Thus, it necessitates a general computation model due to the inherent system heterogeneity.

To address the above challenges, in this paper, we propose a random walk based approach to solve the influence maximization problem in SANs with online activities, and we develop a greedy-based algorithm with two novel optimization techniques to speed up the computation. With our approach and algorithm, we can find a set of nodes which trigger a much larger influence spread than the state-of-the-art algorithm, e.g., IMM [20], while still use a comparable computation time even though our algorithm needs to handle a large amount of online activities. In particular, we make the following contributions in this work.

- We generalize the influence diffusion models for OSNs to SANs by taking online activities into account, and then formulate a new influence maximization problem for SANs.
- We develop a random walk framework on hypergraphs to model the influence diffusion process in SANs, and then define an influence centrality measure based on random walks to measure the influence of nodes in SANs.
- We employ the Monte Carlo framework to efficiently estimate the influence centrality in SANs, and also use the Hoeffding inequality to bound the approximation error. Furthermore, we develop a greedy-based algorithm with two novel optimization techniques to solve the influence maximization problem for SANs.
- We conduct extensive experiments with real-world datasets to validate the effectiveness, efficiency, and generality of our random walk based approach. Results show that our approach greatly improves the seed selection accuracy in SANs, compared to the state-of-the-art algorithm, while only uses a comparable computation time.

The rest of this paper is organized as follows. In §2, we provide the background on influence models in OSNs, and formulate the influence maximization problem in SANs. In §3, we present our random walk based framework on hypergraphs and define the influence centrality measure. In §4, we present a Monte Carlo based approximation algorithm to measure the influence centrality in SANs. In §5, we present our greedy-based algorithm and optimization techniques to solve the influence maximization problem in SANs. In §6, we present the experimental results. At last, we review related work in §7, and conclude the paper in §8.

2. PROBLEM FORMULATION

In this section, we first describe two widely used influence diffusion processes, then we describe the SAN model, and finally we formulate the influence maximization problem for SANs by generalizing the influence diffusion processes for OSNs to SANs with the consideration of online activities.

2.1 Influence Diffusion Processes in OSNs

We first describe the influence diffusion processes in OSNs. The most popular and widely used influence diffusion processes are the independent cascade model (IC) and the linear threshold model (LT), which are proposed by Kempe et al. in [15].

To describe these two influence diffusion models, suppose that each user has two states, either active or inactive, and if an inactive user is influenced by her neighbors, then she becomes active. Now the process of how influence spreads under the IC model can be stated as follows: At first, we initialize a set of users as active. For an active user i, she will activate each of her inactive neighbor j ($j \in N(i)$ where $N(i)$ denotes the neighbor set of user i) with probability q_{ij} ($0 \leq q_{ij} \leq 1$), where q_{ij} represents the influence probability of user i to user j. One common setting of q_{ij} is $q_{ij} = \frac{1}{d_j}$ $[6, 7, 15, 20, 21]$, where d_j denotes the degree of user j, i.e., $d_j = |N(j)|$. If a neighbor j is activated, then she will further activate her inactive neighbors in the set $N(j)$, and this diffusion process continues until no user can change her state in the whole network. We call the expected size of the final set of active users the *influence spread*, and denote it as $S(k)$ if the number of initial active users is equal to k.

The LT model can be described as follows: Every edge (i, j) between user i and j is assigned a weight w_{ij}, and it satisfies the condition $\sum_{i \in N(j)} w_{ij} \leq 1$ for user j. For example, w_{ij} is set as $\frac{1}{d_j}$ for a simple graph [15]. Given the weights, an inactive user j is activated if and only if $\sum_{i \in N(j) \& i \text{ is active}} w_{ij} \geq \theta_j$, where θ_j denotes the threshold of user j and it is selected uniformly at random from the interval $[0, 1]$.

2.2 Model for SANs

We formulate a SAN as a multi-graph $G(V_0, V_1, \cdots, V_l, E_0, E_1, \cdots, E_l)$, where V_0 denotes the set of users, V_i ($i = 1, 2..., l$) denotes the set of type i online activities, E_0 represents the set of edges between users, i.e., the user-user links in OSNs, and E_i ($i = 1, 2..., l$) represents the set of links between users and the type i online activities. In particular, if user j ($j \in V_0$) participated in a type i online activity a ($a \in V_i$), then there is a link $(j, a) \in E_i$). For ease of presentation, we denote $N(j)$ as the set of neighbors of user j, i.e., $N(j) = \{i | (i, j) \in E_0\}$, $N_t(j)$ ($t = 1, 2, ..., l$)

as the set of type t online activities that user j participated, i.e., $N_t(j) = \{a|a \in V_t \,\&\, (j,a) \in E_t\}$, and denote $N_a(j)$ as the set of users except for user j who participated in the online activity a.

To illustrate the SAN model, we use Figure 1 as an example. Assume that we have two types of online activities, i.e., $l = 2$, then we have $V_0 = \{1,2,3,4,5,6\}$, $V_1 = \{a\}$, $V_2 = \{b\}$, $E_0 = \{(1,2),(2,1),(2,3),(3,2),(3,4),(4,3),(4,5),(4,6),(5,4),(6,4)\}$, $E_1 = \{(1,a),(2,a),(3,a),(5,a)\}$, and $E_2 = \{(1,b),(6,b)\}$. We want to point out that even though we consider the OSN in a SAN as an undirected graph in this paper, our model can be easily generalized to the case of directed graph.

2.3 Influence Maximization in SANs

We first generalize the influence diffusion processes (IC and LT) for OSNs to SANs. The key issue in the generalization is to define the influence between user i and user j (i.e., p_{ij} and w_{ij} for IC and LT models) after taking online activities into consideration. Our generalization is based on three criteria:

- A user may be influenced either through user-user links or through user-activity-user links, and the total influence probability in one step is c $(0 < c < 1)$, which is called the decay parameter. Since we have l types of online activities, we define α_{jt} (where $0 \leq \alpha_{jt} \leq 1$ and $0 \leq \sum_{t=1}^{l} \alpha_{jt} \leq 1$) as the proportion of influence to user j through the type t online activities, and call it *weight of activities*. Clearly, $1 - \sum_{t=1}^{l} \alpha_{jt}$ indicates the proportion of influence from direct neighbors.
- For the influence to user j from direct neighbors, we define the weight of each neighbor i $(i \in N(j))$ as u_{ij}. Without loss of generality, we assume that $\sum_{i \in N(j)} u_{ij} = 1$.
- For the influence to user j through the type t online activities, we define the weight of each online activity a as v_{aj}, and assume that $\sum_{a \in N_t(j)} v_{aj} = 1$. Besides, considering that maybe multiple users participated in the online activity a, we define the weight of each user i who participated in a as u_{ij}^{a} $(i \in N(a)\backslash\{j\})$, and assume that $\sum_{i \in N(a)\backslash\{j\}} u_{ij}^{a} = 1$.

For simplicity, we let $u_{ij} = 1/|N(j)|$ in this paper. In fact, this uniform setting is exactly the same as the IC model in OSNs, which has been widely studied in [6,7,15,20,21]. Similarly, we also let $v_{aj} = 1/|N_t(j)|$ and $u_{ij}^{a} = 1/|N_a(j)|$ by following the uniform setting. We would like to point out that our random walk approach in this paper also applies to general settings. Now we can define the influence of user i to user j, which we denote as g_{ij}, as follows.

$$g_{ij} = c \times \Big\{ \Big[\frac{1 - \sum_{t \in [1,l]} \alpha_{jt}}{|N(j)|} \times \mathbf{1}_{\{i \in N(j)\}} \Big] + \Big[\sum_{t \in [1,l]} \sum_{a \in N_t(j)} \frac{\alpha_{jt}}{|N_t(j)|} \times \frac{1}{|N_a(j)|} \times \mathbf{1}_{\{i \in N_a(j)\}} \Big] \Big\}. \quad (1)$$

The first part in the right hand side of Equation (1) denotes the influence diffusion through user-user links, and the second part represents the influence diffusion through user-activity-user links. If we substitute q_{ij} and w_{ij} in IC and LT models with g_{ij} defined above, we get the generalized IC and LT model in SANs, respectively.

Now we re-formulate the influence maximization problem for SANs, which we denote as **IMP(SAN)**, as follows.

Definition 1. **IMP(SAN)**: *Given a SAN $G(V_0, V_1, \cdots, V_l, E_0, E_1, \cdots, E_l)$ and an influence diffusion model (generalized IC or LT), find k nodes, where k is an integer number, so as to make the influence spread of the k nodes $S(k)$ maximized.*

3. METHODOLOGY

In this section, we present our methodology to solve the influence maximization problem in SANs (**IMP(SAN)**). Specifically, we first map the multi-graph which represents a SAN to a hypergraph, then we develop a random walk framework on the hypergraph to estimate the influence diffusion process. Next, we define a centrality measure based on the random walk to estimate the influence of a node set. Finally, we transform the influence maximization problem in SANs (**IMP(SAN)**) to a centrality maximization problem on the hypergraph.

3.1 Multigraph to Hypergraph

We first use a hypergraph model to represent the multi-graph defined in §2.2. The reason why we need this mapping is because the multi-graph model takes both users and online activities as nodes in the graph, but we only need to capture the influence between users. Note that our multi-graph model contains multiple types of edges as we have many different types of online activities. We propose a general hypergraph model which is denoted as $G(V, E, \mathcal{E}_1, ..., \mathcal{E}_l)$, where V denotes the node set, E denotes the set of edges in OSN, and $\mathcal{E}_i$ $(i = 1, 2, ..., l)$ denotes the set of type i hyperedges for characterizing online activities.

We now describe how to construct V, E, and $\mathcal{E}_i$'s from the multigraph $G(V_0, V_1, \cdots, V_l, E_0, E_1, \cdots, E_l)$. Since V_0 denotes the set of users in a SAN, we let $V = V_0$, and let $E = E_0$ which denotes the friendship links between users. For ease of presentation, we also call the edges in E hyperedges. To construct hyperedges in set $\mathcal{E}_i$'s, we define each hyperedge as a set of users who participated in the same online activity, and represent it with a tuple. We use $\mathcal{E}_i$ to denote all the hyperedges related to the type i online activities. Mathematically, $\mathcal{E}_i = \{(j_1, ...j_k)|j_1, ..., j_k \in V_0, (a, j_1), ..., (a, j_k) \in E_i \text{ where } a \in V_i\}$. For ease of presentation, we still denote $N(j)$ as the set of neighbors of user j in the OSN, i.e., $N(j) = \{i|i \in V, (i,j) \in E\}$. We denote $M_e(j)$ as the set of users except for user j who connected to the hyperedge e, i.e., $M_e(j) = \{i|i \in e, \& i \neq j\}$, and denote $\mathcal{E}_t(j)$ as the set of type t hyperedges that are connected to user j, i.e., $\mathcal{E}_t(j) = \{e|e \in \mathcal{E}_t \& j \in e\}$.

3.2 Random Walk on Hypergraph

Here, we present our random walk based framework, which is extended from the classical random walk on a simple unweighted graph $G(V, E)$, which can be stated as follows. For a random walk at vertex $i \in V$, it uniformly selects at random a neighbor j ($j \in N(i)$), and then moves to j in the next step. Mathematically, if we denote $Y(t)$ as the position of the walker at step t, then $\{Y(t)\}$ constitutes a Markov chain where the one-step transition probability p_{ij} is defined as $p_{ij} = 1/|N(i)|$ if $(i,j) \in E$, and 0 otherwise.

We now define the one-step transition probability p_{ij} when performing a random walk on the hypergraph $G(V, E, \mathcal{E}_1, ..., \mathcal{E}_l)$. Note that each hyperedge may contain more than two vertices, so we take the one-step random walk from user i to user j as a two-step process, which is described as follows.

- **Step one:** Choose a hyperedge associated to user i. Precisely, according to the influence diffusion models in §2.3, we set the probability of selecting type t hyperedges as α_{it}, and choose hyperedges of the same type uniformly at random. Mathematically, if the walker is currently at user i, then it chooses a hyperedge e of type t with probability $\frac{\alpha_{it}}{|\mathcal{E}_t(i)|}$.
- **Step two:** Choose a user associated to the hyperedge e selected in step one as the next stop of the random walk. We consider random walks without backtrace. In particular, if

a walker is currently at node i, then we select the next stop uniformly from the vertices that are connected to the same hyperedge with user i. That is, we define the probability of choosing user j ($j \in e$) as $1/|M_e(i)|$.

By combing the two steps defined above, we can derive the transition probability from user i to j as follows.

$$\begin{aligned} p_{ij} &= \frac{1-\sum_{t=1}^{l}\alpha_{it}}{|N(i)|} \times \mathbf{1}_{\{j \in N(i)\}} + \\ & \sum_{t\in[1,l]} \sum_{e \in \mathcal{E}_t(i)} \frac{\alpha_{it}}{|\mathcal{E}_t(i)|} \times \frac{1}{|M_e(i)|} \times \mathbf{1}_{\{j \in M_e(i)\}}. \quad (2) \end{aligned}$$

Considering the SAN in Figure 1, if we let $\alpha_{j1} = 0.5$ and $\alpha_{j2} = 0.3$ for all $j \in V$, then the one-step transition matrix of the random walk on the hypergraph is defined as follows.

$$P = \begin{pmatrix} 0 & 11/30 & 1/6 & 0 & 1/6 & 3/10 \\ 4/15 & 0 & 4/15 & 0 & 1/6 & 0 \\ 1/6 & 4/15 & 0 & 1/10 & 1/6 & 0 \\ 0 & 0 & 1/15 & 0 & 1/15 & 1/15 \\ 1/6 & 1/6 & 1/6 & 1/5 & 0 & 0 \\ 3/10 & 0 & 0 & 1/5 & 0 & 0 \end{pmatrix}.$$

3.3 Influence Centrality Measure

To address the **IMP(SAN)** problem, one key issue is to measure the influence of a node set. To achieve this, we define a centrality measure based on the random walks on hypergraphs to estimate the influence of a node set S, and we call the centrality measure influence centrality, and denote it as $I(S)$. In particular, for a node set S, the influence centrality of S is defined as follows.

$$I(S) = \sum_{j \in V} h(j, S), \quad (3)$$

where $h(j, S)$ captures the influence of S to j, and it is defined as

$$h(j,S) = \begin{cases} \sum_{i \in V} c p_{ji} h(i,S), j \notin S, \\ 1, j \in S, \end{cases} \quad (4)$$

where c is the decay parameter defined in §2.3, and p_{ji} is the one-step transition probability defined in Equation (2).

The definition of the centrality measure $I(S)$ can be further justified as follows. Note that $h(j, S)$ indicates the probability of hitting a user in S when starting from j for random walks defined on the hypergraph, and the parameter c implies the decayed feature, so we call $h(j, S)$ decayed hitting probability. In particular, if $\alpha_t = 0$ ($t = 1, 2..., l$), i.e., considering only walks on the OSN, then $h(j, S)$ is the decayed hitting time [11]. We note that $h(j, S)$ indicates how easily the influence can spread from S to j, and by summing up the influence from S to all users in the whole network, we can well approximate the influence spread of a node set S.

To solve the influence maximization problem of **IMP(SAN)**, we use the influence centrality measure $I(S)$ to estimate the influence of the node set S, and our goal is to find a set S of k users so that $I(S)$ is maximized. In other words, we transform the influence maximization problem **IMP(SAN)** to a centrality maximization problem **CMP** defined as follows.

Definition 2. **CMP**: *Given a hyperghraph $G(V, E, \mathcal{E}_1, ..., \mathcal{E}_l)$, find a set S of k nodes, where k is an integer number, so as to make the influence centrality of the set S of k nodes $I(S)$ maximized.*

In the following sections, we introduce how to estimate $h(j, S)$ by using a random walk approach on hypergraphs (see §4), then we address the influence centrality maximization problem **CMP** by using a greedy approach and develop two novel optimization techniques to speed up the computation (see §5).

4. CENTRALITY COMPUTATION

We note that the key challenge of solving the centrality maximization problem **CMP** is how to efficiently estimate the influence centrality of a node set $I(S)$, or the decayed hitting probability $h(j, S)$. However, deriving the accurate value of $h(j, S)$ in a large SAN is time consuming. To illustrate this, we note that based on the recursive definition of the decayed hitting probability in Equation (4), computing $h(j, S)$ requires to compute $h(i, S)$ for all nodes $i \in V$. Furthermore, estimating the influence centrality $I(S)$ also needs to compute the decayed hitting probability of all nodes. This motivates us to develop an efficient approximation framework to estimate $h(j, S)$.

The high-level idea of our approximation framework can be described as follows. We first rewrite $h(j, S)$ in a linear expression with matrix form, then expand the expression to an infinite converging series, and finally truncate the converging series to save computation time (see §4.1). To further estimate the truncated series, we first explain the expression with a random walk approach, and then use a Monte Carlo framework via random walks to estimate the decayed hitting probability efficiently (see §4.2).

4.1 Linear Expression

Let us transform $h(j, S)$ defined in Equation (4) to a linear expression with matrix notations. The result is stated as follows.

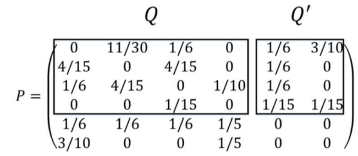
Theorem 1. *The decayed hitting probability $h(j, S)$ defined in Equation (4) can be rewritten as*

$$h(j, S) = c\boldsymbol{e}_j^T(\boldsymbol{I} - c\boldsymbol{Q})^{-1}\boldsymbol{Q}'\boldsymbol{e}, \quad (\textit{for } j \notin S), \quad (5)$$

where $\boldsymbol{Q}$ is a $(|V| - |S|) \times (|V| - |S|)$ dimensional matrix which describes the transition probabilities between two nodes in the set $V - S$, $\boldsymbol{Q}'$ is a $(|V| - |S|) \times |S|$ dimensional matrix which describes the transition probabilities from a node in $V - S$ to a node in S, $\boldsymbol{I}$ is an identity matrix, $\mathbf{e}$ is a column vector with all elements being 1, and finally $\boldsymbol{e}_j$ is a column vector with only the element corresponding to node j being 1 and 0 for all other elements.

Proof: Please refer to the Appendix. ■

To illustrate the linear expression in Equation (5), we give an example to illustrate the computation of $\boldsymbol{Q}$ and $\boldsymbol{Q}'$ in Figure 2.



$$\begin{pmatrix} 0 & 11/30 & 1/6 & 0 & 1/6 & 3/10 \\ 4/15 & 0 & 4/15 & 0 & 1/6 & 0 \\ 1/6 & 4/15 & 0 & 1/10 & 1/6 & 0 \\ 0 & 0 & 1/15 & 0 & 1/15 & 1/15 \\ 1/6 & 1/6 & 1/6 & 1/5 & 0 & 0 \\ 3/10 & 0 & 0 & 1/5 & 0 & 0 \end{pmatrix}$$

Figure 2: Transition matrix $\boldsymbol{Q}$ and $\boldsymbol{Q}'$ when $S = \{5, 6\}$.

Note that the largest eigenvalue of $c\boldsymbol{Q}$ is less than one, so by further expanding the expression in Equation (5) with an infinite series, we can rewrite $h(j, S)$ as follows.

$$h(j, S) = c\boldsymbol{e}_j^T\boldsymbol{Q}'\boldsymbol{e} + c^2\boldsymbol{e}_j^T\boldsymbol{Q}\boldsymbol{Q}'\boldsymbol{e} + c^3\boldsymbol{e}_j^T\boldsymbol{Q}^2\boldsymbol{Q}'\boldsymbol{e} + \cdots. \quad (6)$$

Since the decay parameter c is defined as $0 < c < 1$, and $\boldsymbol{e}_j^T\boldsymbol{Q}^n\boldsymbol{Q}'\boldsymbol{e} \leq 1$ for $n \geq 0$, the infinite series converges, and we can truncate the series to save computation time. In particular, we only keep the L leading terms of the infinite series, and denote the truncated result as $h^L(j,S)$, so we have

$$h^L(j,S) = c\boldsymbol{e}_j^T\boldsymbol{Q}'\boldsymbol{e} + c^2\boldsymbol{e}_j^T\boldsymbol{Q}\boldsymbol{Q}'\boldsymbol{e} + ... + c^L\boldsymbol{e}_j^T\boldsymbol{Q}^{L-1}\boldsymbol{Q}'\boldsymbol{e}. \quad (7)$$

We can also bound the error caused by the series truncation, and mathematically, we have

$$0 \le h(j,S) - h^L(j,S) \le \frac{c^{L+1}}{1-c}. \quad (8)$$

Based on the above error bound, we can see that $h^L(j,S)$ converges to $h(j,S)$ with rate c^{L+1}. This implies that if we want to compute $h(j,S)$ with a maximum error ϵ ($0 \le \epsilon \le 1$), we only need to compute $h^L(j,S)$ by taking a sufficiently large enough L, or $L \ge \lceil \frac{\log \epsilon(1-c)}{\log c} \rceil - 1$.

4.2 Monte Carlo Algorithm

In this subsection, we present a Monte Carlo algorithm to efficiently approximate $h^L(j,S)$. Our algorithm is inspired from the random walk interpretation of Equation (7), and it can achieve a high accuracy by using only a small number of random walks.

We first introduce the random walk interpretation of a particular term $\boldsymbol{e}_j^T\boldsymbol{Q}^{t-1}\boldsymbol{Q}'\boldsymbol{e}$ ($t = 1, ..., L$) in Equation (7). Let us consider a L-step random walk starting from $j \notin S$ on the hypergraph $G(V, E, \mathcal{E}_1, ..., \mathcal{E}_l)$. At each step, if the walker is currently at node k ($k \notin S$), then it selects a node i and transits to i with probability p_{ki}, which is defined in Equation (2). As long as the walker hits a node in S, then it stops. We let $j^{(t)}$ ($t = 1, ..., L$) be the t-th step position, and define an indicator $X(t)$ as follows.

$$X(t) = \begin{cases} 1, & j^{(t)} \in S, \\ 0, & j^{(t)} \notin S. \end{cases}$$

We can see that $\boldsymbol{e}_j^T\boldsymbol{Q}^{t-1}\boldsymbol{Q}'\boldsymbol{e}$ is the probability that a random walk starting from j hits a node in S at the t-th step. We have

$$\boldsymbol{e}_j^T\boldsymbol{Q}^{t-1}\boldsymbol{Q}'\boldsymbol{e} = E[X(t)]. \quad (9)$$

By substituting $\boldsymbol{e}_j^T\boldsymbol{Q}^{t-1}\boldsymbol{Q}'\boldsymbol{e}$ in Equation (7) with Equation (9), we can rewrite $h^L(j,S)$ as

$$h^L(j,S) = cE[X(1)] + c^2E[X(2)] + \cdots + c^LE[X(L)]. \quad (10)$$

Now we estimate $h^L(j,S)$ by using a Monte Carlo method with random walks on the hypergraph based on Equation (10). Specifically, for each node j where $j \notin S$, we set R independent L-step random walks starting from j. For each of the R random walks, if it is currently at node k ($k \notin S$), then it transits to node i in the next step with probability p_{ki} defined in Equation (2). The walk stops until it hits a node in S or already runs L steps. We denote the t-th step position of the R random walks as $j_1^{(t)}, j_2^{(t)}, \ldots, j_R^{(t)}$, respectively, and use $X_r(t)$ to indicate whether $j_r^{(t)}$ belongs to set S or not. Precisely, we set $X_r(t) = 1$ if $j_r^{(t)} \in S$, and 0 otherwise. Now the term $c^tE[X(t)]$ in Equation (10) can be estimated as

$$c^tE[X(t)] \approx \frac{c^t}{R}\sum_{r=1}^{R} X_r(t).$$

By substituting $c^tE[X(t)]$ in Equation (10), we can approximate $h^L(j,S)$, which we denote as $\hat{h}^L(j,S)$, as follows.

$$\hat{h}^L(j,S) = \frac{c}{R}\sum_{r=1}^{R}X_r(1) + \frac{c^2}{R}\sum_{r=1}^{R}X_r(2) + \cdots + \frac{c^L}{R}\sum_{r=1}^{R}X_r(L). \quad (11)$$

Algorithm 1 presents the process of the Monte Carlo method with random walks described above. We can see that its time complexity is $O(RL)$ as the number of types of online activities l is usually a small number. In other words, we can estimate $h^L(j,S)$ in $O(RL)$ time and compute $I(S)$ in $O(nRL)$ time as we need to estimate $h^L(j,S)$ for all nodes. The main benefit of this Monte Carlo algorithm is that its running time is independent of the graph size, so it scales well to large graphs.

Algorithm 1 Monte Carlo Estimation for $h^L(j,S)$

```
function $h^L(j,S)$
    $\sigma \leftarrow 0$;
    for $r$ = 1 to $R$ do
        $i \leftarrow j$;
        for $t$ = 1 to $L$ do
            Generate a random number $x \in [0, 1]$;
            for $T$ = 0 to $l$ do
                if $x \le \alpha_{iT}$ then          ▷ $\alpha_{0T} = 1 - \sum_{T=1}^{l} \alpha_{iT}$;
                    $E \leftarrow \mathcal{E}_T(i)$;
                    break;
                $x \leftarrow x - \alpha_{iT}$;
            Select a hyperedge $e$ from $E$ randomly;
            $i \leftarrow$ select a user from $\{k|k \in e, k \neq i\}$ randomly;
            if $i \in S$ then
                $\sigma \leftarrow \sigma + c^t/R$;
                break;
    return $\sigma$;
end function
```

Note that $\hat{h}^L(j,S)$ that is computed with Algorithm 1 is an approximation for $h^L(j,S)$, and the approximation error depends on the sample size R. To estimate the number of samples required to compute $h^L(j,S)$ accurately, we derive the error bound by applying Hoeffding inequality [12], and the results are stated as follows.

Theorem 2. *Let the output of Algorithm 1 be $\hat{h}^L(j,S)$, we have*

$$P\{|\hat{h}^L(j,S) - h^L(j,S)| > \epsilon\} \le 2L\exp(-2(1-c)^2\epsilon^2R). \quad (12)$$

Proof: Please refer to the Appendix. ■

Based on Theorem 2, we see that Algorithm 1 can estimate $h^L(j,S)$ with a maximum error ϵ with least probability $1 - \delta$ $(0 < \delta, \epsilon < 1)$ by setting $R \ge \log(2L/\delta)/(2(1-c)^2\epsilon^2)$.

5. CENTRALITY MAXIMIZATION

In this section, we develop efficient algorithms to address the centrality maximization problem **CMP** defined in §3.3. Noted that even though we can efficiently estimate the decayed hitting probability $h(j,S)$ by using random walks (see §4), finding a set S of k nodes in a SAN to maximize its influence centrality $I(S)$ is still computationally difficult as it requires to estimate the influence centrality of all combinations of k nodes. In particular, **CMP** is NP-complete, and the result is stated in the following theorem.

Theorem 3. *The centrality maximization problem* **CMP** *is NP-complete.*

Proof: Please refer to the Appendix. ■

To solve the the centrality maximization problem **CMP**, we develop greedy-based approximation algorithms by exploiting the submodularity property of $I(S)$. Specifically, we first show the submodularity property and present a baseline greedy algorithm to maximize $I(S)$, and then develop two novel optimization techniques to further accelerate the greedy algorithm.

Algorithm 2 Baseline Greedy Alg. for Maximizing $I(S)$

```
Input: A hypergraph, and a parameter $k$;
Output: A set $S$ of $k$ nodes for maximizing $I(S)$;
1: $S \leftarrow \emptyset, I(S) \leftarrow 0$;
2: for $s = 1$ to $k$ do
3:     for $u \in (V - S)$ do
4:         $I(S \cup \{u\}) \leftarrow 0$;
5:         for $j \in (V - S \cup \{u\})$ do
6:             $I(S \cup \{u\}) \leftarrow I(S \cup \{u\}) + h(j, S \cup \{u\})$;
7:     $v \leftarrow \arg\max_{u \in (V-S)} I(S \cup \{u\}) - I(S)$;
8:     $S \leftarrow S \cup \{v\}$;
```

Algorithm 3 Optimized Greedy Algorithm

```
Input: A hypergraph and a parameter $k$;
Output: A set $S$ of $k$ nodes for maximizing $I(S)$;
1: $S \leftarrow \emptyset, score[1...n] \leftarrow 0, P[1...n] \leftarrow 0$;
2: for $j \in V$ do
3:     for $r$ = 1 to $R$ do
4:         $i \leftarrow j$;
5:         $visited \leftarrow \emptyset$;
6:         for $t$ = 1 to $L$ do
7:             $visited \leftarrow visited \cup \{i\}$;
8:             $i \leftarrow$ Select a user according to the transition probabilities;
9:             $RW[j][r][t] \leftarrow i$;
10:            if $i \notin visited$ then
11:                $index[i].add(item(j, r, t))$;
12:                $score[i] \leftarrow score[i] + \frac{c^t}{R}$;
13: $v \leftarrow \arg\max_{u \in V} score[u]$;
14: for $s$ = 2 to $k$ do
15:     Update ($RW, index, P, score, S, v, L$);
16:     $S \leftarrow S \cup \{v\}$;
17:     $v \leftarrow \arg\max_{u \in (V-S)} (1 - P[u])(1 + score[u])$;
18: $S \leftarrow S \cup \{v\}$;
```

Algorithm 4 Update Function

```
function UPDATE ($RW, index, P, score, S, v, L$)
    for $w \in index[v]$ do
        $k \leftarrow L$;
        for $t$ = 1 to L do
            if $RW[w.j][w.r][t] \in S$ then
                $k \leftarrow t$;
                break;
        if $k == L$ then
            $P[w.j] \leftarrow P[w.j] + c^t/R$
        for $i = w.t + 1$ to $k$ do
            $u \leftarrow RW[w.j][w.r][i]$
            $score[u] \leftarrow score[u] - c^t/R$;
end function
```

5.1 Baseline Greedy Algorithm

Before presenting the greedy-based approximation algorithm for maximizing $I(S)$, we first show that $I(S)$ is a non-decreasing submodular function, and the result is stated in the following theorem.

Theorem 4. *The centrality measure $I(S)$ is a non-decreasing submodular function.*

Proof: Please refer to the Appendix. ■

Based on the submodularity property, we develop a greedy algorithm for approximation when maximizing $I(S)$, and we call it *the baseline greedy algorithm*. Algorithm 2 describes this procedure. In particular, to find a set of k nodes to maximize $I(S)$, the algorithm works for k iterations. In each iteration, it selects the node which maximizes the increment of $I(S)$.

Recall that the time complexity for estimating the influence of a set S to a particular node $j \notin S$, i.e., $h(j, S)$, is $O(RL)$ (see §4.2). Thus, the total time complexity for the baseline greedy algorithm is $O(kn^2RL)$ where n denotes the total number of users in the SAN, because estimating the influence of a set $S \cup \{u\}$ requires us to sum up its influence to all nodes, and we need to check every node u so to as select the one which maximizes the increment of $I(S)$. Although the baseline greedy algorithm gives a polynomial time complexity, it is inefficient when the number of users in a SAN becomes large. To further speed up the computation, we present two novel optimization techniques in the next subsection.

5.2 Optimizations

The two optimization techniques can be described as follows.

- **Parallel Computation**: Note that the key component in the greedy algorithm is to measure the marginal increment of the influence centrality when adding a node u, i.e., $\Delta(u) = I(S \cup \{u\}) - I(S)$ in Line 5-6 in Algorithm 2. We rewrite $\Delta(u)$ as follows (see the proof of Theorem 4 in the Appendix for derivation).

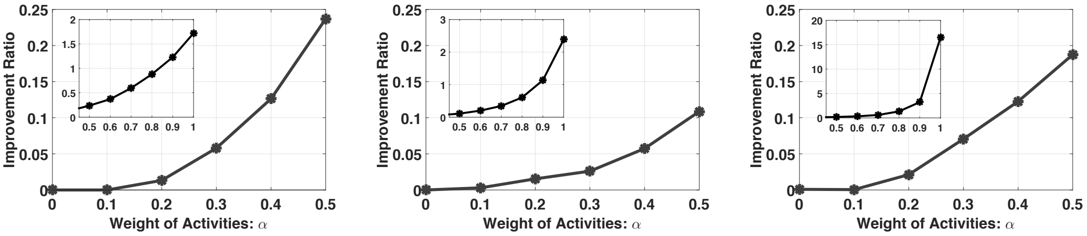
$$\Delta(u) = \left[1 - \sum_{h=1}^{\infty} c^h P(u, S, h)\right] \times \left[1 + \sum_{j \in (V - S \cup \{u\})} \sum_{h=1}^{\infty} c^h P^S(j, \{u\}, h)\right].$$

  In the baseline greedy algorithm, $\Delta(u)$'s are computed sequentially, which as a result incurs a large time overhead.

  Our main idea to speed up the computation is to estimate the marginal increment of all nodes, i.e., $\Delta(u)$ for every u, *in parallel*. Specifically, when performing R random walks from a particular node j, we measure the contribution of j to the marginal increment of every node. In other words, we obtain $P^S(j, u, h)$ for every u by using only the R random walks starting from j. As a result, we need only $O(nR)$ random walks to derive the marginal increment of all nodes, i.e., $\Delta(u)$ for every u, instead of $O(n^2R)$ random walks as in the baseline greedy algorithm.

- **Walk Reuse**: The core idea is that in each iteration of choosing one node to maximize the marginal increment, we record the total $O(nR)$ random walks in memory, and apply the updates accordingly after one node is added into the result set. By doing this, we can reuse the $O(nR)$ random walks to derive the marginal increment in the next iteration instead of starting new random walks from each node again.

By incorporating the above optimization techniques, we can reduce the time complexity to $O(nRL)$, where L denotes the maximum walk length. In other words, we can use the L leading terms to estimate $\sum_{h=1}^{\infty} c^h P^S(j, \{u\}, h)$ and $\sum_{h=1}^{\infty} c^h P(u, S, h)$ as described in §4. Thus, we let each walk runs for L steps at most. Algorithm 3 states the procedure. We use $score[u]$ and $P[u]$ to record $\sum_{j \in V - S \cup \{u\}} \sum_{h=1}^{\infty} c^h P^S(j, \{u\}, h)$ and $\sum_{h=1}^{\infty} c^h P(u, S, h)$ for computing $\Delta(u)$, respectively. Algorithm 3 runs in two phases. The first phase (line 1-13) is to select the first seed node by running random walks and also record all the walking information for reuse. The second phase (line 14-18) is to select the remaining $k-1$ nodes based on the stored information which requires to be updated after selecting each node. We give the update function in Algorithm 4.



(a) Ciao

(b) Yelp

(c) Flixster

Figure 3: Impact of online activities on influence spread.

The update function is to update the walk information stored in $score$ and P. Every time after we selecting a seed node v, the random walk in the following iterations should stop when it encounters v, and the values stored in $score$ and P should change accordingly. To achieve this, for each random walk that hits v (line 2), we first check if it has visited any node in S already (line 4-7). If not, then we increase $P[w.j]$ after adding v in S (line 8,9). Since the following walks should stop when hitting v, we update $score[u]$ if node u is visited after v (line 10-12).

6. EXPERIMENTS

We conduct experiments on real-world datasets to evaluate our approach. We first show that incorporating the online activities in seed selection can lead to a significant improvement on the influence spread, i.e., influence more users with the same seed size. Then we show that our IM-RW algorithm attains a good tradeoffs between the performance (in terms of influence spread) and the running time cost as compared to the state of the art influence maximization algorithms. We then show that our IM-RW algorithm is quite general, i.e., it applies to multiple types of users and multiple types of online activities, and also applies to both IC and LT influence diffusion models. Lastly, we show that under different seed sizes, our IM-RW algorithm outperforms the state-of-the-art algorithm, and it is also time efficient.

6.1 Datasets

We consider three datasets from social rating systems: Ciao [1], Yelp [2] and Flixster [13]. Such social rating networks are composed of a social network, where the links can be interpreted as either friendships (undirected link) or a following relationship (directed link), and a rating network, where a link represents that a user assigns a rating (or writes a review) to a product. Assigning a rating corresponds to an online activity, and multiple users assigning ratings to the same product means that they participate in the same online activity. In the rating network, we remove rating edges if the associated rating is less than 3 so as to filter out the users who dislike a product. Through this we guarantee that all the remaining users who give ratings to the same product have similar interests, e.g., they all like the product. Since the original Flixster dataset is too large to run the state of the art influence maximization algorithms, we extract only a subset of the Flixster dataset for comparison studies. In particular, since the OSN of Flixster is almost a connected component, we randomly select a user, and run the breadth-first search algorithm until we get 300,000 users. We state the statistics of the three datasets in Table 1. All algorithms are implemented in C++, and run on a server with two Intel Xeon E5-2650 2.60GHz CPU and 64GB memory.

6.2 The Benefit of Incorporating Activities

We first show that incorporating the online activities in seed selection can lead to a significant improvement on the influence spread.

Dataset	Users	Links in OSN	Products	Ratings	OSN Type
Ciao	2,342	57,544	15,783	32,783	directed
Yelp	174,100	2,576,179	56,951	958,415	directed
Flixtser	300,000	6,394,798	28,262	2,195,134	undirected

Table 1: Datasets Statistics.

We fix the seed size k as 50, and use two algorithms to select the seed set: our IM-RW algorithm, and the state-of-the-art influence maximization algorithm IMM [20]. After selecting the seed sets, we use simulations to estimate the expected influence spread of the selected k users on SANs, i.e., the expected number of users that will be eventually influenced, and denote the results as $S(\text{IM-RW})$ and $S(\text{IMM})$, respectively. Finally, we define the improvement ratio on the expected influence spread as follows.

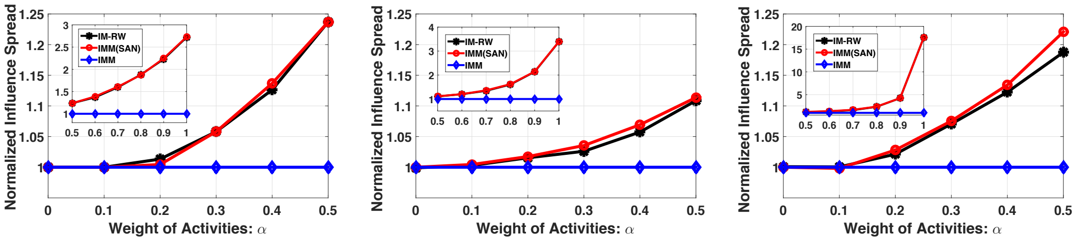
$$\text{Improvement Ratio} = [S(\text{IM-RW}) - S(\text{IMM})]/S(\text{IMM}).$$

Remark: The IMM [20] algorithm is the state-of-the-art influence maximization algorithm in terms of time complexity, while it does not consider online activities and can return a solution with theoretical performance guarantees in near-optimal time. Thus, $S(\text{IM-RW})$ and $S(\text{IMM})$ denote the expected influence spreads with/without considering online activities, respectively. In later experiments, we use IMM as our benchmark algorithm.

To present the key insights, let us consider the simple case in which there is only one type of users and one type of online activities. Namely, all users have a same value of α which indicates the weight of activities to influence users. We will study the general case of multiple types of users and online activities later. Let us focus on the IC model here, we emphasize that our algorithm also applies to the LT model as well and we will discuss it later. We vary the value of α from 0 to 1. Figure 3 depicts the improvement of incorporating online activities by varying the weight of activities α from 0 to 1. The horizontal axis shows the value of α, and the vertical axis presents the corresponding improvement ratio. From Figure 3, one can observe that the improvement ratio is 0 when $\alpha = 0$. This is because users are not affected by other users through online activities when $\alpha = 0$. As α increases, the improvement ratio also increases. This shows that as users are more prone to be affected by other users through online activities, incorporating online activities bring larger benefit. When $\alpha = 0.5$, the improvement ratio is around 25% for Ciao dataset. That is, we can influence 25% more users when incorporating online activities in the seed selection. Similar conclusions can also be observed for the datasets of Yelp and Flixster. It is interesting to observe that as α approaches to one, the improvement ratio reaches up to 16 for Flixster, which implies a more than an order of magnitude improvement. In summary, incorporating online activities in the seed selection by using IM-RW significantly improves the selection accuracy.

6.3 Tradeoffs of Comparison

Note that IMM is originally developed to solve the influence

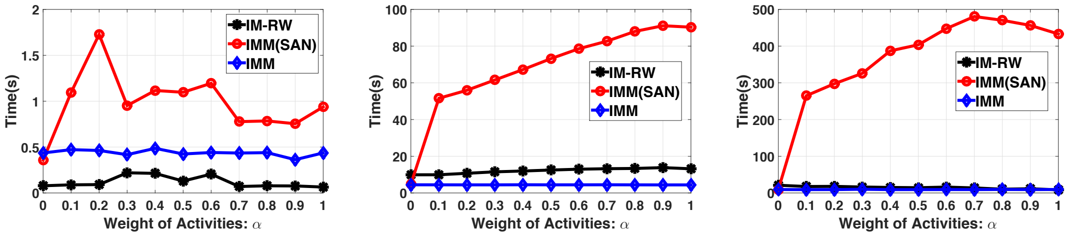


(a) Ciao

(b) Yelp

(c) Flixster

Figure 4: Normalized influence spread of IM-RW, IMM and IMM(SAN).



(a) Ciao

(b) Yelp

(c) Flixster

Figure 5: Running time of IM-RW, IMM and IMM(SAN).

maximization problem in OSNs, while it may also be extended to work on SANs. For example, one can first transform the SANs to a weighted OSN by mapping online activities shared by any two users to a weight on the link of the two users in the OSN, then apply the IMM algorithm on the weighted OSN. However, we note that the weighted OSN may become very dense as many pairs of users may participate in a same online activity. As a result, the time complexity of IMM may become very high as it depends on the number of edges in a graph. In this subsection, we study the tradeoffs between the performance (in terms of the expected influence spread) and the running time. We consider the same setting as in §6.2.

For comparison studies, we consider three algorithm: (1) IM-RW, (2) IMM which works on OSN as in §6.2, and (3) IMM(SAN) which denotes applying the IMM algorithm on SANs by transforming online activities to weights on links in OSNs. To run the IMM(SAN) algorithm, we first map online activities into weights based on the influence models in §2.3, then we run IMM on the weighted OSN. Thus, IMM(SAN) can be viewed as an improved version of IMM algorithm with online activities also being considered. Note that mapping a SAN to a weighted OSN may take a long time if the number of online activities is large, and for fair comparison, we exclude the preprocess time for IMM(SAN).

To study the tradeoffs, we first show the influence spread performance of the three algorithms in Figure 4. For comparison, we normalize the results by setting the influence spread of IMM, i.e., $S(\text{IMM})$, to one. The horizontal axis shows the value of α, and the vertical axis presents the corresponding normalized influence spread. We see that by taking online activities into consideration, IMM(SAN) and IM-RW achieve almost the same performance, while both of them perform much better than IMM, and the improvement ratio keeps increasing as α increases. However, by comparing the running time as shown in Figure 5, we see that the running time of IMM and IM-RW are comparable, while IMM(SAN) takes much longer time, especially when the network is large and online activities become more important (i.e., larger α).

Summary: There is a tradeoff between the influence spread performance and running time for the IMM algorithm on SANs. In particular, considering online activities (i.e., IMM(SAN)) improves the influence spread performance, but incurs a very large time overhead. Our IM-RW algorithm achieves much better performance in seed selection than the state-of-the-art algorithm IMM, while only requires a comparable running time even though it takes online activities into consideration and can handle larger networks.

6.4 Model Generality

We show the generality of our algorithm (IM-RW) in two aspects: (1) It is applicable to different influence diffusion models, e.g., the LT model, and (2) it is applicable to heterogeneous system settings, e.g., multiple types of users and online activities.

6.4.1 The LT model

We now show that in addition to the IC model, our IM-RW algorithm also applies to the LT model. We still consider one type of users and one type of online activities. Similar to §6.2, we show the improvement ratio (in terms of the expected influence spread) of the IM-RW algorithm over the IMM algorithm in Figure 6. We observe similar improvement ratio as in the case of IC model shown in Figure 3. This implies that our IM-RW algorithm maintains the benefit of incorporating online activities under different influence diffusion settings. Besides, the running time is around 0.1 seconds for the Ciao dataset, around 10 and 20 seconds for the Yelp and Flixster dataset, respectively, as shown in Table 2. This implies that IM-RW is still highly efficient under the LT model.

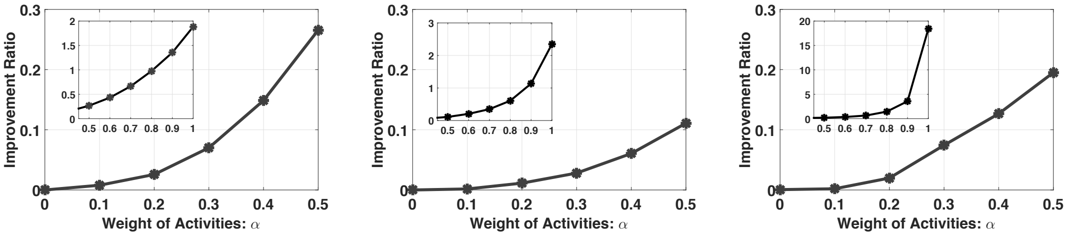
Dataset	Running Time (s)										
	$\alpha=0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Ciao	0.105	0.114	0.100	0.103	0.115	0.097	0.102	0.118	0.117	0.100	0.104
Yelp	8.396	9.838	10.655	11.267	11.940	12.321	12.828	13.362	13.382	13.639	14.147
Flixster	18.169	18.233	17.448	21.350	15.352	14.618	12.965	11.833	10.717	8.575	6.121

Table 2: Running time of IM-RW under the LT model.

6.4.2 Multiple User Types and Activity Types

To study the generality of our IM-RW algorithm, we consider three heterogeneous settings: (1) User heterogeneity: two types of users and one type of online activities, (2) Activity heterogeneity: one type of users and two types of online activities, and (3) Full heterogeneity: two types of users and online activities.

Note that in the datasets there is no information on user types and activity types, so we synthesize user and activity types by following the 80-20 rule [14], which is widely adopted in the fields of

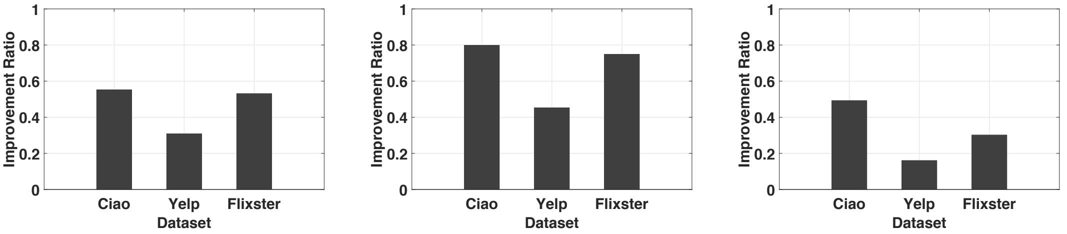


(a) Ciao

(b) Yelp

(c) Flixster

Figure 6: Improvement ratio under the LT model.



(a) User Heterogeneity

(b) Activity Heterogeneity

(c) Full Heterogeneity

Figure 7: Improvement ratio under heterogeneous settings.

economics and computer science. In particular, in the user heterogeneity case, we randomly divide users into two types where the first type accounts for 80% of population. Considering that users are usually easy to be influenced by online activities in real life, we set the weight of activities α for the first type as 0.8, and set it as 0.2 for the second type of users. In the activity heterogeneity case, we also divide online activities into two types by following the 80-20 rule and fix the total weight of activities as 0.8. Precisely, the weight of the first type of online activities is set as $0.8 \times 0.2 = 0.16$ and that of the second type is set as $0.8 \times 0.8 = 0.64$. Similarly, in the full heterogeneity case, we divide both users and online activities into two types by using the parameters in the first two cases.

Figure 7 shows the improvement ratio of IM-RW over IMM in terms of the expected influence spread. We observe that the improvement ratio is always over 18% in all cases, and in particular, it reaches to 80% for the Ciao dataset in activity heterogeneity case. This shows that our IM-RW algorithm significantly outperforms the IMM algorithm by incorporating online activities even under the heterogeneous system setting. Besides, for the running time shown in Table 3, we see that IM-RW only requires around 10 seconds. This implies that IM-RW is also efficient even though it needs to handle a lot of online activities.

Dataset	Running Time (s)		
	User Heterogeneity	Activity Heterogeneity	Full Heterogeneity
Ciao	0.078	0.059	0.060
Yelp	13.254	8.615	9.018
Flixster	11.870	8.993	10.180

Table 3: Running time of IM-RW under heterogeneous settings.

6.5 Impact of Seed Size

We now study the impact of seed size k on IM-RW. For brevity, we present only the results under IC model, and consider one type of users and online activities. We fix α as 0.8 in this experiment. Figure 8 shows the expected influence spread of IM-RW and IMM by varying the seed size k from 20 to 80. We see that as we increase the seed size, more users will eventually get influenced. In particular, IM-RW always outperforms IMM significantly under different seed sizes. Furthermore, the running time of IM-RW presented in Table 4 also indicates that IM-RW is efficient even though online activities are considered, and more importantly, the running time is independent of the seed size k, which benefits from the walk reuse optimization described in §5.2.

Dataset	Running Time (s)						
	k=20	30	40	50	60	70	80
Ciao	0.070	0.065	0.076	0.076	0.191	0.191	0.199
Yelp	14.394	13.418	13.305	13.408	13.375	16.610	13.820
Flixster	11.139	11.286	10.969	11.383	11.522	11.493	11.196

Table 4: Running time of IM-RW with different seed sizes.

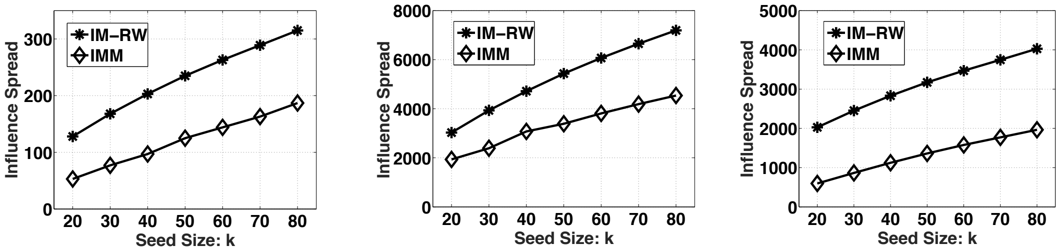
7. RELATED WORK

Influence maximization problem in OSNs was first formulated by Kempe et al. [15], and in this seminal work, the authors proposed the IC model and the LT model. Since then, this problem receives a lot of interests in academia in the past decade [6–8]. Because of the NP-hardness under both the IC model [6] and the LT model [8], many of the previous studies focus on how to reduce the time complexity. Recently, Borgs et al. [5] developed an algorithm which maintains the performance guarantee while reduces the time complexity significantly, and Tang et al. [20, 21] further improved the method and proposed the IMM algorithm, which is the state-of-the-art solution by now. Besides, influence maximization with multiple sources, which is called competitive influence maximization, was also studied, e.g., [18].

Centrality measure based approach was also studied, for example, the studies [7, 9, 10, 23] find the most influential nodes based on degree centrality and closeness centrality. In particular, these studies measure the influence of a node or a node set based on the degree, or the shortest-path distance to other nodes in the network.

In terms of random walk, it is widely used to analyze big graphs, e.g., PageRank computation [19], graph sampling [24], and SimRank [16] etc. Some studies [17, 22] also analyze the importance of a node based on random surfer model, e.g., they use the truncated hitting time or decayed hitting time to reflect the influence of a node set in OSNs. Random walk on simple hypergraphs has also been well studied in [3, 4, 25].

Our work distinguishes from previous work as follows. First, in terms of the problem, we focus on SANs and study the impact



(a) Ciao

(b) Yelp

(c) Flixster

Figure 8: Influence spread of IM-RW and IMM with different seed sizes.

of online activities on the influence maximization problem, while previous works only consider the influence diffusion through user-user links in OSNs. Second, our random walk based framework is developed on general hypergraphs where multiple types of hyperedges exist. At last, we develop an efficient algorithm with two novel optimization techniques to accelerate the computation, and the time complexity of our algorithm is independent of the number of links in the network, so it is computationally efficient in the case of SANs which may contain many online activities.

8. CONCLUSIONS

In this paper, we study the impact of online activities on influence maximization via a random walk approach. Specifically, we generalize the influence diffusion models in OSNs to SANs, and then formulate the influence maximization problem for SANs by taking online activities into account. To tackle this problem, we propose a general framework to measure the influence of nodes in SANs via random walks on hypergraphs, and develop a greedy-based algorithm to find the top k most influential nodes in SANs by using random walks. We also develop two novel optimization techniques to further speed up the computation. By conducting extensive experiments with real-world datasets, we show that our approach greatly improves the seed selection accuracy with a comparable computation time compared to IMM, the state-of-the-art algorithm. Our algorithm is computationally efficient even though it needs to handle a large amount of online activities. Furthermore, our approach is also general enough to work under different influence diffusion models and heterogeneous system settings, e.g., multiple types of users and online activities in SANs.

9. REFERENCES

[1] http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm.

[2] Yelp. https://www.yelp.com/dataset_challenge/dataset.

[3] C. Avin, Y. Lando, and Z. Lotker. Radio Cover Time in Hyper-Graphs. In *Proceedings of The 6th International Workshop on Foundations of Mobile Computing*, pages 3–12. ACM, 2010.

[4] A. Bellaachia and M. Al-Dhelaan. Random Walks in Hypergraph. In *Proceedings of The International Conference on Applied Mathematics and Computational Method*, pages 187–194, 2013.

[5] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing Social Influence in Nearly Optimal Time. In *Proceedings of The 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2014.

[6] W. Chen, C. Wang, and Y. Wang. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *Proceedings of The 16th International Conference on Knowledge Discovery and Data Mining*, pages 1029–1038. ACM, 2010.

[7] W. Chen, Y. Wang, and S. Yang. Efficient Influence Maximization in Social Networks. In *Proceedings of The 15th International Conference on Knowledge Discovery and Data Mining*, pages 199–208. ACM, 2009.

[8] W. Chen, Y. Yuan, and L. Zhang. Scalable Influence Maximization in Social Networks Under The Linear Threshold Model. In *Proceedings of ICDM*. IEEE, 2010.

[9] M. G. Everett and S. P. Borgatti. The Centrality of Groups and Classes. *The Journal of Math. Soc.*, 23(3):181–201, 1999.

[10] L. C. Freeman. Centrality in Social Networks Conceptual Clarification. *Social Networks*, 1(3):215–239, 1979.

[11] Z. Guan, J. Wu, Q. Zhang, A. Singh, and X. Yan. Assessing and Ranking Structural Correlations in Graphs. In *Proceedings of The International Conference on Management of Data*. ACM, 2011.

[12] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of The American Statistical Association*, 58(301):13–30, 1963.

[13] M. Jamali and M. Ester. A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks. In *ACM RecSys*, 2010.

[14] J. M. Juran and J. F. Riley. *The Quality Improvement Process*. McGraw Hill New York, NY, 1999.

[15] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing The Spread of Influence Through a Social Network. In *Proceedings of The 9th International Conference on Knowledge Discovery and Data Mining*, pages 137–146. ACM, 2003.

[16] M. Kusumoto, T. Maehara, and K.-i. Kawarabayashi. Scalable Similarity Search for Simrank. In *Proceedings of The International Conference on Management of Data*, pages 325–336. ACM, 2014.

[17] R.-H. Li, J. X. Yu, X. Huang, and H. Cheng. Random-Walk Domination in Large Graphs. In *Proceedings of ICDE*. IEEE, 2014.

[18] Y. Lin and J. C. Lui. Analyzing Competitive Influence Maximization Problems With Partial Information: An Approximation Algorithmic Framework. *Performance Evaluation*, 91:187–204, 2015.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to The Web. 1999.

[20] Y. Tang, Y. Shi, and X. Xiao. Influence Maximization in Near-Linear Time: A Martingale Approach. In *Proceedings of The International Conference on Management of Data*, pages 1539–1554. ACM, 2015.

[21] Y. Tang, X. Xiao, and Y. Shi. Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency. In *Proceedings of The International Conference on Management of Data*. ACM, 2014.

[22] C. Zhang, L. Shou, K. Chen, G. Chen, and Y. Bei. Evaluating Geo-Social Influence in Location-Based Social Networks. In *Proceedings of The 21st International Conference on Information and Knowledge Management*, pages 1442–1451. ACM, 2012.

[23] J. Zhao, J. Lui, D. Towsley, and X. Guan. Measuring and Maximizing Group Closeness Centrality over Disk-Resident Graphs. In *Proceedings of The Companion Publication of The 23rd International Conference on World Wide Web Companion*, pages 689–694, 2014.

[24] J. Zhao, J. Lui, D. Towsley, P. Wang, and X. Guan. A Tale of Three Graphs: Sampling Design on Hybrid Social-Affiliation Networks. In *Proceedings of ICDE*. IEEE, 2015.

[25] D. Zhou, J. Huang, and B. Schölkopf. Learning With Hypergraphs: Clustering, Classification and Embedding. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2006.

Appendix

Proof of Theorem 1: Based on the definition of $h(j,S)$ in Equation (4), we can get

$$h(j,S) = \sum_{h=1}^{\infty} c^h P(j,S,h), \text{ for } j \notin S,$$

where $P(j,S,h)$ denotes the probability that a random walk starting from j hits a node in S at the h-th step. Now if we denote p_{iS} as the probability that a random walk starting from i hits a node in S in one step, we have

$$\begin{aligned} h(j,S) &= \sum_{h=1}^{\infty} c^h P(j,S,h) = \sum_{h=1}^{\infty} c^h \sum_{i \notin S} (\boldsymbol{Q}^{h-1})_{ji} p_{iS}, \\ &= \sum_{i \notin S} \sum_{h=1}^{\infty} c^h (\boldsymbol{Q}^{h-1})_{ij} p_{iS} = \sum_{i \notin S} c(\boldsymbol{I} - c\boldsymbol{Q})^{-1}_{ji} p_{iS}, \\ &= c\boldsymbol{e}_j^T (\boldsymbol{I} - c\boldsymbol{Q})^{-1} \boldsymbol{Q}' \boldsymbol{e}. \end{aligned}$$

■

Proof of Theorem 2: Let X_1,...,X_R be R independent random variables with $X_r \in [0,1]$ for all $r = 1, \cdots, R$, and set $T = (X_1 + ... + X_R)/R$. According to Hoeffding's inequality, we have $P\{|T - E(T)| > \epsilon\} \leq 2\exp(-2\epsilon^2 R)$. By applying this inequality, we have

$$\begin{aligned} & P\{|\hat{h}^L(j,S) - h^L(j,S)| > \epsilon\} \\ = \; & P\Big\{ |\sum_{t=1}^{L} \frac{c^t}{R} \sum_{r=1}^{R} X_r(t) - \sum_{t=1}^{L} c^t E[X(t)]| > \epsilon \Big\}, \\ \leq \; & P\Big\{ \sum_{t=1}^{L} |\frac{c^t}{R} \sum_{r=1}^{R} X_r(t) - c^t E[X(t)]| > \epsilon \Big\}, \\ \leq \; & \sum_{t=1}^{L} P\Big\{ |\frac{c^t}{R} \sum_{r=1}^{R} X_r(t) - c^t E[X(t)]| > (1-c)c^t \epsilon \Big\}, \\ \leq \; & 2L \exp(-2(1-c)^2 \epsilon^2 R). \end{aligned}$$

■

Proof of Theorem 3: To prove the NP-completeness, we first consider the maximum coverage problem (weighted version): Given an integer k and a collection of sets $S = \{S_1, S_2, ..., S_m\}$, we let $U = \cup_{1 \leq i \leq m} S_i$, and let each element $e \in U$ have a weight $w(e)$. The goal is to find a subset $S' \subseteq S$, such that $|S'| \leq k$ and $\sum_{e \in E} w(e)$, where $E = \cup_{S_i \in S'} S_i$, is maximized. The maximum coverage problem is a NP-complete problem.

Now we consider a special case of the centrality maximization problem by setting $\alpha_{it} = 0$ for $t = 1, ..., l$. We define a maximization problem as follows: For every node i, we define a set $N^L(i) = \{j | d(i,j) \leq L\}$ where $d(i,j)$ is the shortest path distance from i to j, and set the weight of each node $j \in N^L(i)$ as $w(j) = h^L(j,S)$. Our goal is to find a set S of nodes, such that $|S| \leq k$ and $\sum_{e \in E} w(e)$ is maximized, where $E = \cup_{i \in S} N^L(i)$. Note that this maximization problem equivalents to the maximum coverage problem, and so it is also NP-complete. Noted that $\sum_{e \in E} w(e)$ estimates the influence centrality $I(S)$, so the centrality maximization problem is NP-complete. ■

Proof of Theorem 4: We first show the non-decreasing property. Note that since $h(j,S) = 1$ if $j \in S$, so we rewrite $I(S)$ as follows.

$$I(S) = |S| + \sum\nolimits_{j \in (V-S)} h(j,S).$$

Now suppose that a user $u \notin S$ is added into the set S, then the marginal increment of the influence centrality $\Delta(u) = I(S \cup \{u\}) - I(S)$ can be derived as

$$\begin{aligned} \Delta(u) &= \sum_{j \in V} h(j, S \cup \{u\}) - \sum_{j \in V} h(j,S), \\ &= 1 + \sum_{j \in (V - S \cup \{u\})} h(j, S \cup \{u\}) - \sum_{j \in (V-S)} h(j,S), \\ &= 1 - h(u,S) + \sum_{j \in (V - S \cup \{u\})} \Big[h(j, S \cup \{u\}) - h(j,S) \Big]. \end{aligned}$$

According to the definition of $h(j,S)$ in Equation (6) and the random walk interpretation, we rewrite $h(j,S)$ as

$$h(j,S) = \sum_{h=1}^{\infty} c^h P(j,S,h), \text{ for } j \notin S,$$

where $P(j,S,h)$ denotes the probability that a random walk starting from j hits a node in S at the h-th step for the first time. Now we can rewrite $h(j, S \cup \{u\}) - h(j,S)$ as

$$\begin{aligned} & h(j, S \cup \{u\}) - h(j,S) \\ &= \sum_{h=1}^{\infty} c^h P(j, S \cup \{u\}, h) - \sum_{h=1}^{\infty} c^h P(j,S,h) \\ &= \sum_{h=1}^{\infty} c^h \Big[P^{\{u\}}(j,S,h) + P^S(j,\{u\},h) \Big] - \\ & \quad \sum_{h=1}^{\infty} c^h \Big[P^{\{u\}}(j,S,h) + P^S(j,\{u\},h) P(u,S,h) \Big] \\ &= \sum_{h=1}^{\infty} c^h P^S(j,\{u\},h) \Big[1 - \sum_{h=1}^{\infty} c^h P(u,S,h) \Big] \\ &= \sum_{h=1}^{\infty} c^h P^S(j,\{u\},h) \Big[1 - p(u,S,h) \Big], \end{aligned}$$

where $P^T(j,S,h)$ represents the probability that a random walk starting from j hits a node in S at the h-th step for the first time without passing any node in T. Therefore, $\Delta(u)$ can be derived as follows.

$$\begin{aligned} \Delta(u) &= I(S \cup \{u\}) - I(S) \\ &= (1 - h(u,S)) \Big[1 + \sum_{j \in V - S \cup \{u\}} \sum_{h=1}^{\infty} c^h P^S(j,\{u\},h) \Big]. \end{aligned} \quad (13)$$

Now we show the non-negativity of $\Delta(u)$. Note that $0 < c < 1$ and $\sum_{h=1}^{\infty} P(u,S,h) \leq 1$, so we have $h(u,S) \leq 1$ and $1 - h(u,S) \geq 0$. That is, $\Delta(u) \geq 0$, and $I(S)$ is a non-decreasing function.

We now show that $I(S)$ is a submodular function. Mathematically, we only need to prove that the inequality $I(S \cup \{u\}) - I(S) \geq I(T \cup \{u\}) - I(T)$, for $S \subseteq T$, holds. Note that $P^S(j,\{u\},h) \geq P^T(j,\{u\},h)$ if $S \subseteq T$. Besides, according to the non-decreasing feature of $I(S)$, we have $h(u,S) \leq h(u,T)$. Based on these inequalities and Equation (13), we can obtain $I(S \cup \{u\}) - I(S) \geq I(T \cup \{u\}) - I(T)$ if $S \subseteq T$. Therefore, $I(S)$ is a submodular function. ■