Friend Recommendation in Online Social Networks: Perspective of Social Influence Maximization

Huanyang Zheng and Jie Wu
Department of Computer and Information Sciences, Temple University, USA
Email: {huanyang.zheng, jiewu}@temple.edu

***Abstract*—In online social networks, people may want to make new friends to maximize their social influences. For example, business page owners on Facebook want to influence as many people as possible for commercial advantages. Hence, we study a friend recommendation strategy with the perspective of social influence maximization. For the system provider (e.g., Facebook), the objective is to recommend a fixed number of new friends to a given user, such that the given user can maximize his/her social influence through making new friends. Our problem is proved to be NP-hard. A greedy friend recommendation algorithm with an approximation ratio of $1 - \frac{1}{e}$ is proposed, according to the submodular property. It involves a sub-problem of computing the influence spread. A novel method, which considers the multipath effect, is proposed to compute the influence spread. Experiments demonstrate the efficiency and effectiveness of our algorithms.**

***Index Terms*—Online social network, friend recommendation, social influence maximization, multipath effect.**

I. Introduction

Online Social Networks (OSNs) mainly focus on building social relations among users who share interests, activities, backgrounds, stories, and real-life connections. They are valuable tools used by many people to extend their daily contacts. Most OSNs are web-based and provide means for users to interact with each other over the Internet. OSNs are not only a way to keep in touch, but also a way of life. Existing OSNs such as Facebook, Twitter, and VK account for three of the top ten most-visited web sites in the world [1]. As of January 2014, 74% of online adults use OSNs [2].

As one of the essential components in OSNs, the friend recommendation system aims to seek appropriate people with whom users can make new friends. Classic approaches make recommendations according to the social proximities among the users, hypothesizing that people with close social circles are potential friends. For instance, the Facebook "People You May Know" feature recommends people to connect with each other, according to a friend-of-a-friend strategy [3]. In other words, if two unconnected users share many common friends, then they are recommended to become new friends. Friend recommendations on Facebook prioritize friends-of-friends over strangers (i.e., people who are not friends or friends-of-friends). Content-based and location-based recommendation strategies have also been proposed. In these approaches, people who share similar contents, or are geographically nearby, are recommended to connect with each other [4, 5].

We observe that people may want to make new friends with the objective of maximizing their social influences. The world-famous best-selling book, "How to Win Friends and Influence People," considers making friends and influencing people to be closely interrelated [6]. Consequently, people can use OSNs to advance their careers and businesses. For example, Facebook provides business page services [7] for their owners to influence other people for commercial advantages (selling and advertising). Twitter also provides page promotion services for salesmen to attract high-value followers as potential customers. A successful story is that of Drew Ressler, who dramatically gained 1,300 new followers while targeting audiences interested in music and photography [8].

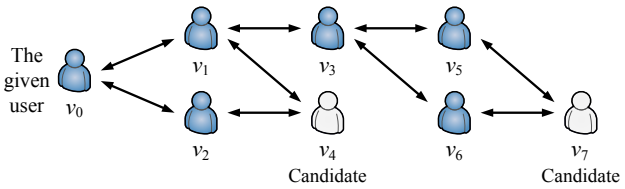


Fig. 1. The tradeoff in the friend recommendation strategy.

This paper focuses on the friend recommendation strategy with the perspective of social influence maximization. For the system provider, the objective is to recommend a fixed number of new friends to a given user, such that the given user can maximize his/her social influence through new friends. Our problem is proved to be NP-hard. New challenges arise from the tradeoff between the *friend acceptance probability* and the *propagation capability*. The friend acceptance probability is self-explanatory. The propagation capability measures the influence spread beyond the given user that is brought by the recommended friend. A motivational example is shown in Fig. 1, where v_0 wants to maximize his/her social influence through making a new friend (candidates are v_4 and v_7). Arrows in Fig. 1 are bidirectional friendships. In terms of the friend acceptance probability, v_4 is a friend-of-a-friend of v_0, but v_7 is a stranger. Hence, v_4 is more likely to accept the friend request from v_0 and then propagate v_0's influence. However, in terms of the propagation capability, v_4 cannot further propagate v_0's influence to the users v_5 and v_6 (since they do not connect with each other). On the other hand, v_7 is influential and can propagate v_0's influence to v_5 and v_6, but v_7 is not likely to accept the friend request from v_0. This is because v_0 and v_7 are not socially proximal to each other. The tradeoff between the friend acceptance probability and the propagation capability should be investigated.

A greedy friend recommendation strategy is proposed to balance the above tradeoff. Moreover, it is an extension of the classic social influence maximization problem [9]. It involves an NP-hard sub-problem of the influence spread computation [10]: given the OSN topology and a specified user, how many people can this user influence? Since OSNs are typically large, the scalability issue challenges classic solutions [11]. Through leveraging the structural properties of OSNs, we propose a novel method to efficiently compute the influence spread, based on the multipath effect.

Our main contributions are summarized as follows:

- We address a novel friend recommendation problem with the perspective of social influence maximization. Our problem is proved to be NP-hard. A greedy approximation algorithm with a ratio of $1 - \frac{1}{e}$ is discussed.
- We propose a novel influence spread computation method to support greedy friend recommendations. The multipath effect is explored.
- Extensive real data-driven experiments are conducted to evaluate the proposed algorithms. Evaluation results are shown from different perspectives to provide insightful conclusions for real-world applications.

The remainder of this paper is organized as follows. Section II surveys related works. Section III formulates the problem. Section IV describes the greedy recommendation algorithm. Section V computes the influence spread. Section VI includes experiments. Finally, Section VII concludes the paper.

II. Related Work

Social Influence Propagation. Independent cascade is one of the most classic models for describing social influence propagations [9, 12]. It starts with a set of initially-influenced people, and then executes a probabilistic rule to propagate influences. The number of eventually-influenced people is denoted as the influence spread, the computation of which is NP-hard [10]. Several time-efficient methods were proposed to estimate the influence spread. Chen et al. [10, 11] simplified the influence spread computation by restricting the influence to propagate along the maximum influence path. Borgs et al. [13] studied the influence spread by sampling methods, assuming that the statistical properties of an OSN is stable. This paper uses the existing independent cascade model to address a novel friend recommendation problem. Our contributions also include a novel influence spread computation method.

Friend Recommendation. The friend recommendation system is an essential component of an OSN. Authors in [3, 14] hypothesized that people with close social circles are potential friends. Bu et al. [4] proposed that users who like the same music should be recommended to become friends with each other. Zhang et al. [15] considered that users make friends by sharing the same profiles. As for location-based approaches [5, 16], geographically nearby users are recommended to connect with each other. Differing from previous approaches, our friend recommendation strategy focuses on the perspective of maximizing the social influence of a specified user.

III. Preliminaries and Problem Formulation

A. Independent Cascade

The scenario of this paper is an OSN, which is modeled as a directed weighted graph $G = (V, E)$. Here, V is a set of nodes (users), and $E \subseteq V^2$ is a set of weighted edges (friendships between users). An edge from user v to user u is denoted by e_{vu} with the edge weight of w_{vu}. Edge weights serve as probabilities for influence propagations, depending on the friendship closeness and the content popularity [17]. We use existing works [9–12] to determine edge weights. The *independent cascade model* is used to simulate influence propagations. It starts with a set of initially-influenced nodes (called seed nodes). The other nodes are initially-uninfluenced. The influence propagation process unfolds in discrete time steps according to the following probabilistic rule [9]. When a node v first becomes influenced in a time step, it has a single chance to influence each currently uninfluenced neighbor u. Then, the probability that u is influenced by v depends on w_{vu}. If u has multiple newly-influenced neighbors, their influence propagations can be sequenced in an arbitrary order. Once u is influenced, it will influence its neighbors in the next time step; however, u does not propagate any further influences in subsequent time steps. The above process terminates until no more uninfluenced nodes are influenced. We have:

***Definition** 1:* The expected number of nodes eventually-influenced by seed nodes is defined as the *influence spread*.

Unfortunately, the computation of the influence spread is known as NP-hard [10]. Currently, it could be computed by time-consuming Monte-Carlo simulations, information-lossy heuristics, or sampling methods with strong assumptions.

B. Problem Formulation

This paper studies the friend recommendation strategy with the perspective of social influence maximization (i.e., *maximizing influence spread*). The motivation is that people may want to make new friends to maximize their social influence. People, such as political party leaders, film stars, and business salesmen, sometimes combine making friends and influencing people as a lifestyle. Therefore, OSNs have a strong potential for adopting our friend recommendation strategy. For example, a salesman on Facebook or Twitter would like to influence as many people as possible for commercial advantages [18, 19].

This paper designs a friend recommendation strategy for an OSN system provider (e.g., Facebook). The objective is to recommend a fixed number (denoted by k) of new friends to a given user (denoted by v_0), such that v_0 can maximize his/her social influence spread through making new friends. The independent cascade model [9] is adopted as the influence cascade model, where v_0 is initially-influenced and the other nodes are initially-uninfluenced. The existing friends of v_0 are eliminated in the recommendations. In other words, we want to maximize the influence spread of v_0 through k new friendships (new edges). The friend acceptance probabilities depend on the social proximities between v_0 and the recommended people. This is because a stranger is less likely to accept the friend request from v_0 than a friend-of-friend of v_0.

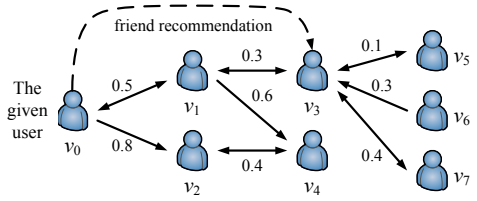


Fig. 2. An example for the friend recommendation.

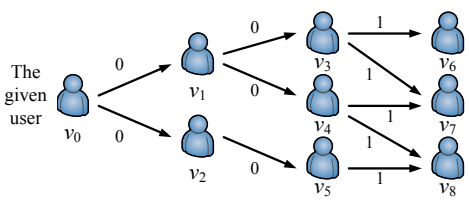


Fig. 3. Proof of NP-hardness.

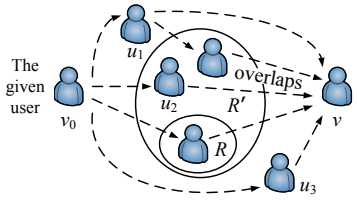


Fig. 4. Proof of submodular property.

An example is shown in Fig. 2, where edges are directed and the numbers on edges are their weights. If v_3 is recommended to v_0, a new edge of $e_{v_0v_3}$ may be added, depending on the friend acceptance probability that is estimated by their social proximity. Similar to the other edge weights, the weight, $w_{v_0v_3}$, of the new edge is also determined by existing works [9–12], according to the friendship closeness and the content popularity. The challenge comes from the tradeoff between the friend acceptance probability and the propagation capability. Although influential strangers can effectively propagate v_0's influence, they are less likely to accept the friend request from v_0. On the other hand, friends-of-friends are likely to accept the friend request from v_0, but may not effectively propagate v_0's influence. This tradeoff should be explored.

IV. Combining Friend Recommendation and Social Influence Maximization

A. Friend Acceptance Probability

This subsection describes the friend acceptance probability. If v is recommended to v_0 as a new friend, v_0 may influence more people through forming a new friendship with v. Let f_{v_0v} denote the friend acceptance probability that v accepts the friend request from v_0 (i.e., a new edge of e_{v_0v} is formed). Strangers are people who are not friends or friends-of-friends of v_0. We have the following justifications for f_{v_0v}:

- f_{v_0v} can be different for different strangers of v. This is because v_0 could judge their friendship potential from multiple fields such as common interests, school of graduation, place of work, and so on.
- Compared to the event in which v is a stranger of v_0, f_{v_0v} should be larger if v is a friend-of-a-friend of v_0. This is because people are more likely to accept a friend request from friends-of-friends than from strangers.

These two unique properties are the major differences between our metric and existing classic metrics (e.g., Jaccard's coefficient and Katz coefficient) [9]. Based on above justifications, f_{v_0v} is formally defined as follows:

$$f_{v_0v} = \alpha_v \times \frac{|N(v_0) \cap N'(v)|}{|N(v_0) \cup N'(v)|} + \beta_v \tag{1}$$

In Eq. 1, $N(v)$ and $N'(v)$ denote the sets of outgoing and incoming neighboring nodes of v, respectively. The fraction of $\frac{|N(v_0)\cap N'(v)|}{|N(v_0)\cup N'(v)|}$ is the common neighbor similarity from v_0 to v. It measures the number of common friends of v_0 and v. α_v and β_v are coefficients. To guarantee $0 \leq f_{v_0v} \leq 1$, we have $0 \leq \alpha_v \leq 1$, $0 \leq \beta_v \leq 1$, and $0 \leq \alpha_v + \beta_v \leq 1$. While α_v measures the impact of friends-of-friends, β_v measures the probability that people form a friendship with a stranger. α_v and β_v can vary among different people. When v is a friend-of-a-friend of v_0, f_{v_0v} is no smaller than β_v. This is because $N(v_0) \cap N'(v) \neq \emptyset$. On the other hand, when v is a stranger to v_0, f_{v_0v} reduces to β_v, which is its minimum value. An example is shown in Fig. 2. If we have $\alpha_{v_3} = 0.5$ and $\beta_{v_3} = 0.1$, then $f_{v_0v_3} = 0.5 \times \frac{1}{5} + 0.1 = 0.2$.

Note that f_{v_0v} does not depend on the edge weight w_{v_0v}. This is simply because the edge weight measures the influence propagation probability rather than the friendship closeness. Eq. 1 can be improved by considering the friendship closeness. Another notable point is that we use existing works [9–12] to determine the weights of edges, including the weights of new edges that results from friend recommendations.

B. NP-hardness, Submodularity, and Greedy Approximation

This subsection explores the friend recommendation problem. Let R denote the set of users that are recommended to v_0 for making new friends. The constraint is $|R| \leq k$, and $|R|$ is the set cardinality of R. Once a user (say v) is recommended to v_0, the probability that v accepts the friend request from v_0 is f_{v_0v}. The independent cascade model is adopted stimulate influence propagations. In our problem, only v_0 is initially-influenced, and the other nodes are initially-uninfluenced. Let $\sigma(R)$ denote the influence spread. Based on Definition 1, $\sigma(R)$ is the expected number of nodes eventually-influenced by v_0 with friend recommendations in R. Note that, when $R = \emptyset$, $\sigma(R)$ is not necessarily 0, since v_0 can still influence existing friends. We have the following theorem:

***Theorem** 1:* Our friend recommendation problem, which selects R to maximize $\sigma(R)$, is NP-hard.

Proof: The proof is done by reducing the *Maximum Coverage Problem* (MCP) to a special case of our problem. The MCP is NP-hard [20], and is based on sets of elements. Given a number of k, its objective is to select k sets, such that the number of covered elements are maximized. If a set is selected, its elements are covered.

The reduction is done by mapping sets and elements to 2-hop and 3-hop friends of v_0, respectively. Edge weights in our problem are specially designed. Only weights of edges from 2-hop friends to 3-hop friends of v_0 are 1, and the others are 0. As a special case of our problem, let G be a directed acyclic graph with the source of v_0. Fig. 3 shows such an example: set $\{v_6, v_7\}$ is mapped to node v_3, set $\{v_7, v_8\}$ is mapped to node v_4, and set $\{v_8\}$ is mapped to node v_5. We use $f_{v_0v} = 1$ and $w_{v_0v} = 1$ for all $v \in R$. At this time, the

Algorithm 1 Greedy Friend Recommendation

Input: The graph, G, and the given user, v_0;
Output: The set of recommended people, R;

1: Initialize $R = \emptyset$;
2: **for** $i = 1$ to k **do**
3: Select $v = \arg\max_u \left[\sigma(R \cup \{u\}) - \sigma(R)\right]$;
4: $R = R \cup \{v\}$;
5: **return** R;

optimal recommendation will only recommend 2-hop friends of v_0, since 3-hop friends can be influenced through 2-hop friends. For example, the optimal recommendation will not recommend v_6 and v_7, since the recommendation of v_3 is better. Hence, people, who are influenced by v_0 in the optimal recommendation, are composed of exactly k 2-hop friends and some 3-hop friends that correspond to the optimally-covered elements in the MCP. Therefore, the MCP reduces to a special case of our problem. Since the MCP can be reduced from the set cover problem that is NP-complete [20], our friend recommendation problem is NP-hard. ■

The idea of our proof is to weaken the tradeoff between the friend acceptance probability and the propagation capability by using $f_{v_0 v} = 1$ and $w_{v_0 v} = 1$ for all $v \in R$. We have:

***Definition** 2:* $\sigma(R)$ is submodular, if it satisfies a natural diminishing returns property: the marginal gain from adding a node to the set R is at least as high as the marginal gain from adding the same node to a superset of R.

***Theorem** 2:* $\sigma(R)$ is submodular with respect to R.

Proof: Let $p_R(v)$ denote the probability that the node v is eventually-influenced by v_0 with R. Clearly, we have $\sigma(R) = \sum_{v \in V \setminus \{v_0\}} p_R(v)$. We show that $p_R(v)$ is submodular with respect to R, through considering the influence propagation paths from v_0 to v. To see this, let R' denote an arbitrary superset of R, i.e., $R \subseteq R'$. Submodularity means that

$$p_{R \cup \{u\}}(v) - p_R(v) \geq p_{R' \cup \{u\}}(v) - p_{R'}(v) \qquad (2)$$

where $u \in V \setminus R$. This inequality in Eq. 2 clearly holds, when the influence propagation paths from v_0 to v via u have some overlaps with those via $R' \setminus R$. An example of such overlaps is shown as u_1 and u_2 in Fig. 4, where dashed arrows are influence propagation paths. The equality in Eq. 2 holds, only when the influence propagation paths from v_0 to v via u are fully independent with those via $R' \setminus R$. An example is u_3 in Fig. 4. Therefore, for all v, $p_R(v)$ is submodular with respect to R. Considering that a non-negative linear combination of submodular functions is also submodular, we can conclude that $\sigma(R)$ is submodular with respect to R. ■

The insight behind Theorem 2 is very intuitive: the social influences brought by the people who are recommended later have potential overlaps with those who are recommended earlier. Hence, the marginal gain of the influence spread satisfies the law of diminishing returns. According to [21], a greedy algorithm with a submodular objective function guarantees an approximation ratio of $1 - \frac{1}{e}$ to the optimal algorithm. This algorithm is shown as Algorithm 1. It iteratively selects the user, who can maximize the marginal influence spread, into the recommendations. Note that the tradeoff between the friend acceptance probability and the propagation capability has been automatically considered in the computation of $\sigma(R)$. Existing friends of v_0 are eliminated in the friend recommendations.

A critical drawback of Algorithm 1 is that the computation of the influence spread is NP-hard [10]. Consequently, line 3 in Algorithm 1 cannot be optimally computed within a polynomial time complexity. State-of-the-art [10, 11, 22] cannot decently solve this problem. Hence, in the next section, a novel method is proposed to efficiently and accurately compute the influence spread. It serves as a sub-algorithm for line 3 in Algorithm 1, i.e., it computes $\sigma(R)$ for a given R.

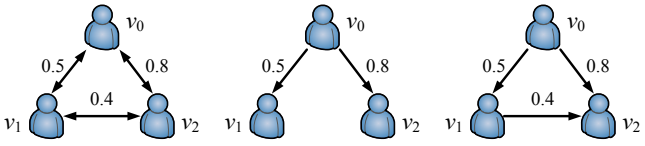
V. Influence Spread Computation

A. Classic Approaches and Their Limitations

The influence spread computation is NP-hard [10], and it is usually done by Monte-Carlo simulations. The most classic approaches for this problem are proposed by Chen et al. [10, 11]. They simplified the influence spread computation by restricting the influence to propagate along the maximum influence path. In other words, the graph is reduced to an arborescence structure (called maximum influence arborescence model). Later, their method is improved by reducing the graph to a directed acyclic graph (DAG). An example is shown in Fig. 5. The original graph is shown in Fig. 5(a), where v_0 is initially-influenced and the other nodes are initially-uninfluenced. The probability that the node v is eventually-influenced is denoted by $p_R(v)$. We have $\sigma(R) = \sum_{v \in V \setminus \{v_0\}} p_R(v)$. The challenge comes from the *multipath effect*, meaning that there may exist an exponential number of paths to propagate the influence from v_0 to an arbitrary node. In Fig. 5(a), v_1 can be possibly influenced via the v_0-v_1 path, or the v_0-v_2-v_1 path. Note that the number of paths in a graph increases exponentially with respect to the number of nodes, leading to the NP-hardness for the influence spread computation. To restrict the number of paths, the approach in [10] reduces the graph to a tree, as shown in Fig. 5(b). We get $p_R(v_1) = 0.5$ and $p_R(v_2) = 0.8$ by this approach. It is information-lossy since $e_{v_1 v_2}$ and $e_{v_2 v_1}$ are discarded. The improved approach in [11] reduces the graph to a DAG, as shown in Fig. 5(c). We get $p_R(v_1) = 0.5$ and $p_R(v_2) = 0.84$ for this approach, where only $e_{v_2 v_1}$ is discarded. Considering the multipath effect, the optimal result in Fig. 5(a) is that $p_R(v_1) = 0.5 + 0.8 \cdot 0.4 - 0.5 \cdot 0.8 \cdot 0.4 = 0.66$ and $p_R(v_2) = 0.8 + 0.5 \cdot 0.4 - 0.8 \cdot 0.5 \cdot 0.4 = 0.84$. Classic approaches are inaccurate, since some edges are removed to restrict the number of paths for influence propagations.

B. Multipath Effect in Influence Propagations

Our key idea for the influence spread computation is to mitigate the multipath effect by considering several top influence propagation paths within a non-exponential time complexity. There exists a tradeoff between the number of paths and the computation accuracy of the influence spread. If we consider more paths in the computation of the influence spread, then the



(a) Original graph. (b) Tree reduction. (c) DAG reduction.

Fig. 5. An example for the classic approaches and their limitations.

result is more accurate, at the cost of a higher time complexity. Through a coarse estimation, this subsection shows how many paths are sufficient to obtain an accurate $p_R(v)$ for node v.

Our coarse estimation is based on the existing literature for scale-free networks [23]. OSNs are typically scale-free [24]. Let $\langle\cdot\rangle$ denote the mean value of a variable. For example, $\langle w\rangle$ is the average edge weight and $\langle d\rangle$ is the average out-degree. Let $|V|$ denote the number of nodes in G. Let $N_L(v)$ denote the expected number of paths from v_0 to v that have L intermediate nodes. We assume that each node (excluding v_0 and v) has a probability of λ_v to be an intermediate node in a path from v_0 to v. Note that λ_v ($0 \leq \lambda_v \leq 1$) represents the graph skewness among different users. We have:

$$N_L(v) = \left[\frac{(|V|-2)!}{(|V|-2-L)!} \cdot \lambda_v^L \cdot (1-\lambda_v)^{|V|-L}\right] \cdot \frac{\langle d\rangle^L}{|V|^L} \quad (3)$$

Here, $\frac{(|V|-2)!}{(|V|-2-L)!}$ is the number of permutations for selecting L nodes from $|V|-2$ nodes (excluding v_0 and v) as *ordered* intermediate nodes on the path from v_0 to v. The probability of the corresponding permutation is $\lambda_v^L \cdot (1-\lambda_v)^{|V|-L}$. Then, $\frac{\langle d\rangle}{|V|}$ is the average probability that a predecessor node on the path connects to the successor node. In scale-free networks, we typically consider that L is small with respect to $|V|$ [23], meaning that $\frac{(|V|-2)!}{(|V|-2-L)!} \cdot \frac{1}{|V|^L} \approx \frac{1}{|V|}$. We rewrite Eq. 3:

$$N_L(v) \approx \frac{(1-\lambda_v)^{|V|}}{|V|} \cdot \left[\frac{\lambda_v}{1-\lambda_v}\langle d\rangle\right]^L \quad (4)$$

When $\frac{\lambda_v}{1-\lambda_v}\langle d\rangle > 1$, $N_L(v)$ is expected to grow exponentially with respect to L, and thus, traversing all the paths from v_0 to v is time-consuming (the general case). Let L^* denote the length of the unweighted shortest path from v_0 to v, we have:

$$N_{L^*}(v) \approx \frac{(1-\lambda_v)^{|V|}}{|V|} \cdot \left[\frac{\lambda_v}{1-\lambda_v}\langle d\rangle\right]^{L^*} \quad (5)$$

L^* and $N_{L^*}(v)$ can be obtained by Dijkstra's algorithm. $|V|$ and $\langle d\rangle$ can be obtained through network statistics. Therefore, λ_v can be adaptively computed through Eq. 5.

The expected probability that v is influenced by v_0 through a path of length L is $\langle w\rangle^L$. All the paths with length L, in total, should bring an expected influence propagation probability of $1-(1-\langle w\rangle^L)^{N_L(v)}$. If these paths are independent of each other and $\langle w\rangle^L \ll 1$, then we have the following estimation:

$$\begin{aligned} &1-(1-\langle w\rangle^L)^{N_L(v)} \approx \langle w\rangle^L N_L(v) \\ &\approx \frac{(1-\lambda_v)^{|V|}}{|V|} \cdot \left[\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle\right]^L \end{aligned} \quad (6)$$

Eq. 6 implies two insights. When $\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle \geq 1$, v is likely to be eventually-influenced by v_0, since Eq. 6 becomes close to one. In this case, several top paths are sufficient to propagate v_0's influence to v. This is because these paths already bring an influence probability that is close to one. Consequently, we do not need to consider longer paths for the computation of $p_R(v)$. The hard scenario is $\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle < 1$. In this case, Eq. 6 decreases exponentially with respect to L. This indicates that v_0's influence is much more likely to propagate along the shorter paths than the longer paths. Top paths can dominate the influence propagation. When $\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle < 1$, the following equation can show such dominations:

$$\begin{aligned} &\frac{1-\prod_{L=L^*}^{L^*+l}(1-\frac{(1-\lambda_v)^{|V|}}{|V|}\left[\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle\right]^L)}{1-\prod_{L=L^*}^{\infty}(1-\frac{(1-\lambda_v)^{|V|}}{|V|}\left[\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle\right]^L)} \\ &\geq \frac{\sum_{L=L^*}^{L^*+l}\frac{(1-\lambda_v)^{|V|}}{|V|}\left[\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle\right]^L}{\sum_{L=L^*}^{\infty}\frac{(1-\lambda_v)^{|V|}}{|V|}\left[\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle\right]^L} = 1-\left[\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle\right]^{l+1} \end{aligned} \quad (7)$$

The fraction in the top line of Eq. 7 is the ratio of (i) the expected influence propagation probability brought by paths with lengths from L^* to L^*+l to (ii) the expected influence propagation probability brought by all paths. Eq. 7 shows that, if we consider a little bit more paths (in addition to the shortest path), then the computational error of $p_R(v)$ can be exponentially reduced. Although the estimation in Eq. 7 is coarse-grained, it still provides an important intuition for the influence spread computation: if we rank all the paths from v_0 to v by their influence probabilities, then the several top paths are sufficient to approximate $p_R(v)$. The next subsection computes the influence spread based on this intuition.

C. Multipath-sensitive Influence Spread Computation

Motivated by Eq. 7, we propose a Polynomial-Time Approximation Scheme (PTAS) for the influence spread computation, assuming $\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle < 1$. The key idea to mitigate the multipath effect by considering several top influence propagation paths within a non-exponential time complexity. We argue that this PTAS should also work well when $\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle \geq 1$, since several top paths already bring an influence probability that is close to one (as analyzed in Eq. 6). Let ε denote a control parameter ($0 \leq \varepsilon \leq 1$). The PTAS is expected to obtain a ratio, $1-\varepsilon$, to the optimal algorithm (under some assumptions in the derivation of Eq. 7). Based on Eq. 7, we have:

$$1-\varepsilon = 1-\left[\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle\right]^{l+1} \quad (8)$$

Since parameters ε, $\langle w\rangle$, $\langle d\rangle$, and λ_v are known, we have:

$$l = -1 + \ln\varepsilon \,/\, \ln\left[\frac{\lambda_v}{1-\lambda_v}\langle w\rangle\langle d\rangle\right] \quad (9)$$

Algorithm 2 Influence Spread Computation

Input: The graph, G, and the given user, v_0;
The set of recommended people, R;
The control parameter, ε;
Output: The influence spread, $\sigma(R)$;

1: **for** each node v in R **do**
2: Add the edge, e_{v_0v}, with weight, w_{v_0v}, to the graph, G;
3: Compute f_{v_0v} based on Eq. 1;
4: Update $w_{v_0v} = f_{v_0v} \times w_{v_0v}$;
5: Call unweighted Dijkstra's algorithm from v_0 to determine the unweighted shortest paths from v_0 to the other nodes; then, L^* and $N_{L^*}(v)$ can be obtained for each v;
6: Parameters L^*, $N_{L^*}(v)$, $\langle d \rangle$, V, and ε are known; based on Eq. 5, compute λ_v for each v; based on Eq. 10, compute the total number of paths, θ_v, for each v;
7: Convert the edge weights to distance weights through a negative $\log(\cdot)$ operation; call weighted Yen's algorithm to compute the loopless weighted shortest paths from v_0 to the other nodes; for each v, θ_v paths from v_0 are obtained;
8: **for** each node v in G except v_0 **do**
9: Based on θ_v loopless paths, construct a DAG from v_0 to v; based on the topological order, compute $p_R(v)$;
10: **return** $\sigma(R) = \sum_{v \in V \setminus \{v_0\}} p_R(v)$;

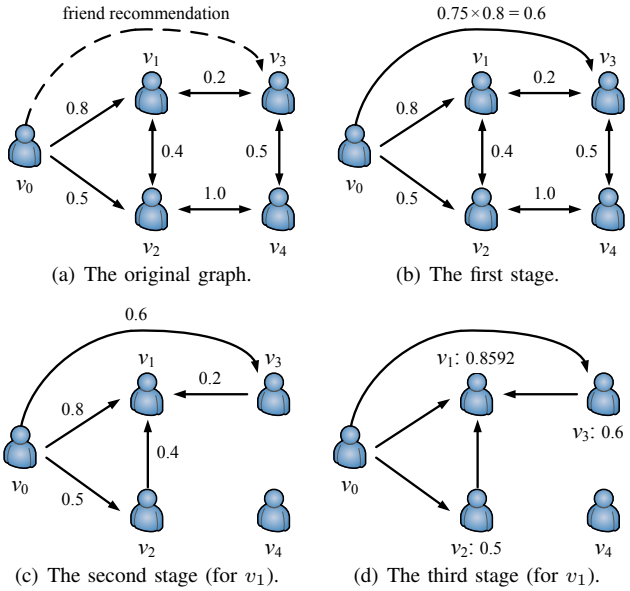
The total number of paths from v_0 to v, denoted by θ_v, is:

$$\theta_v = \sum_{L=L^*}^{L^*+l} N_L(v) = N_{L^*}(v) \cdot \sum_{i=0}^{l} \left[\frac{\lambda_v}{1-\lambda_v}\langle d \rangle\right]^i$$
$$= N_{L^*}(v) \cdot \frac{\left[\frac{\lambda_v}{1-\lambda_v}\langle d \rangle\right]^{\frac{\ln \varepsilon}{\ln[\frac{\lambda_v}{1-\lambda_v}\langle w \rangle \langle d \rangle]}} - 1}{\left[\frac{\lambda_v}{1-\lambda_v}\langle d \rangle\right] - 1} \quad (10)$$

θ_v is rounded to an integer for the algorithm implementation.

Algorithm 2 is proposed as the PTAS. It is a sub-algorithm for line 3 in Algorithm 1. It include three stages:

- The first stage (lines 1 to 4) processes the friend recommendations. For $v \in R$, a new edge is added. The edge weight is determined by the existing works [9–12]. f_{v_0v} is computed based on Eq. 1. Edge weights are updated to show the impact of the friend acceptance probability.
- The second stage (lines 5 to 7) determines the set of paths to compute the influence spread. Through a coarse estimation, the number of paths, θ_v, is *adaptively* determined. Note that θ_v can be different for different v, since the graph skewness is considered by λ_v. We convert the edge weights to the distance weights through a negative $\log(\cdot)$ operation. Yen's algorithm [25] is called to compute the set of loopless weighted shortest paths from v_0 to v. For each v, we obtain θ_v paths from v_0. If $\theta_v < 1$, we use only one path. If θ_v is larger than the number of available loopless paths, then all loopless paths are used.
- In the third stage (lines 8 to 10), for each v, we compute the probability that v is influenced by v_0, i.e., $p_R(v)$. Since θ_v loopless paths are obtained by the second stage, a DAG can be constructed. Then, $p_R(v)$ can be computed following a topological order in the DAG.



(a) The original graph. (b) The first stage.
(c) The second stage (for v_1). (d) The third stage (for v_1).

Fig. 6. An example for the influence spread computation.

An example of Algorithm 2 is shown in Fig. 6. Fig. 6(a) shows G with $R = \{v_3\}$. In the first stage of Fig. 6(b), $e_{v_0v_3}$ is added with $w_{v_0v_3} = 0.75$. Existing works [9–12] are used to determine all the edge weights, including $w_{v_0v_3}$. Suppose $\alpha_{v_3} = 0.9$ and $\beta_{v_3} = 0.5$, then we compute $f_{v_0v_3} = 0.9 \times \frac{1}{3} + 0.5 = 0.8$ based on Eq. 1. To incorporate the friend acceptance probability, $w_{v_0v_3}$ is updated to be 0.6. In the second stage of Fig. 6(c), we determine the set of paths for the influence spread computation. The number of paths, θ_v, is determined by Eqs. 5 and 10. Our example only includes the process for v_1. For simplicity, let $\theta_{v_1} = 3$. Then, Yen's algorithm determines three loopless weighted shortest paths from v_0 to v_1, i.e., v_0-v_1, v_0-v_2-v_1, and v_0-v_3-v_1. Since these paths are loopless, the third stage of Fig. 6(d) computes $p_R(v_1)$ with a topological order, $p_R(v_1) = 1-(1-0.8)\times(1-0.5\cdot0.4)\times(1-0.6\cdot0.2) = 0.8592$.

Due to Yen's algorithm [25], the time complexity of Algorithm 2 is $O(\theta|V|\cdot(|E|+|V|\log|V|))$. Here, θ is the maximum number of paths for a node, i.e., $\theta = \max_v \theta_v$. The first stage takes $O(k)$ to go through each recommendation. The third stage takes $O(|V|\cdot(|V|+|E|))$ due to the topological sort. The key insight of Algorithm 2 is to mitigate the multipath effect by considering several top influence propagation paths within a non-exponential time complexity. To guarantee accuracies, the number of need paths is estimated by Eqs. 5 and 10.

VI. Experiments

A. Dataset Information

Our experiments use three datasets: Facebook [26], Epinions [27], and Wiki [28]. Facebook is an online social networking service launched in 2004. Users on Facebook can create a user profile, add other users as friends, post status updates and photos, share videos, and receive notifications when others update their profiles. The Facebook "People You May Know" feature

TABLE I
DATASET STATISTICS

	Facebook	Epinions	Wiki
Number of nodes	63,731	18,098	7,115
Number of edges	817,035	355,754	103,689
Average degree	25.6	19.6	14.6
Network Diameter	15	11	7
Global clustering coefficient	0.148	0.138	0.141
Average edge weight	0.0271	0.0285	0.0076

recommends new friends based on a friend-of-a-friend strategy [3]. Facebook also provides business page services [7] to users for social influence maximizations, meaning that our approach can be potentially applied on Facebook. Epinions is a general consumer review site launched in 1999. Epinions users could read new and old reviews about a variety of products to help them decide on a purchase. Our approach can be applied on users who want to disseminate their product reviews [29]. Wiki is a free encyclopedia written collaboratively by volunteers (i.e., users). A small portion of users are administrators. In order for a user to become an administrator, a request must be issued and voted. A directed edge is represented by that a user votes for another user. Our approach can be applied on users who want to get more votes. Note that these three datasets do not include information on edge weights. Following existing works [9–12], we use directed common neighbor similarities [9] to determine edge weights. Finally, the statistics of these three datasets are shown in Table I.

B. Comparison Algorithms

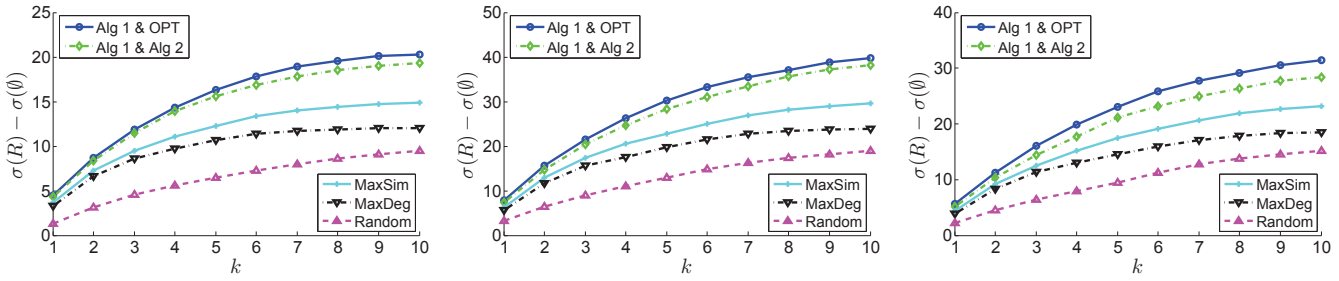
Our experiments focus on the increased influence spread brought by the recommendations, under different settings (e.g., numbers of recommended people, values of α_v and β_v, and outgoing degrees of v_0). The increased influence spread can be computed by $\sigma(R) - \sigma(\emptyset)$. Note that $\sigma(\emptyset)$ is not necessarily 0, since v_0 can still influence some people through existing friends. For comparison, six algorithms are included. (i) Alg 1 & OPT stands for Algorithm 1 with the optimal influence spread computation. The influence spread is computed through time-consuming Monte-Carlo simulations. Alg 1 & OPT guarantees an approximation ratio of $1 - \frac{1}{e}$ to the optimal algorithm. (ii) Alg 1 & Alg 2 is Algorithm 1 with a non-optimal influence spread computation. The influence spread is computed through Algorithm 2 (we set $\varepsilon = 0.1$ by default). We want to check the gap between Alg 1 & OPT and Alg 1 & Alg 2. (iii) MaxSim is for a greedy algorithm that iteratively selects a user with a maximum common neighbor similarity (in terms of v_0). It is currently used on Facebook [3]. Recommended people are friends-of-friends of v_0. (iv) MaxDeg stands for a greedy algorithm that iteratively selects a user with a maximum outgoing degree. Recommended people have the highest outgoing degrees. (v) Random is a baseline algorithm that recommends friends uniform-randomly. All these algorithms are implemented in a centralized manner.

We also set up experiments to compare different algorithms for the influence spread computation. For comparison, five algorithms are included, as described in the following. (i) Tree [10]. This algorithm computes the influence spread through a maximum influence arborescence model, where influences are restricted to propagate along maximum influence paths. (ii) DAG [11]. This algorithm reduces the graph to a DAG to compute the influence spread. The influence probabilities are sequentially determined following a topological order in the DAG. (iii) IMRank [22]. This algorithm computes the influence spread through iterative neighborhood exchanges. The probability that a given node is influenced depends on that of its neighbors. The drawback is that IMRank ignores path dependency issues. (iv) Alg 2 denotes Algorithm 2, which computes the influence spread through considering additional paths. (v) OPT. This algorithm optimally computes the influence spread by time-consuming Monte-Carlo simulations [30].

C. Evaluation Results on Friend Recommendations

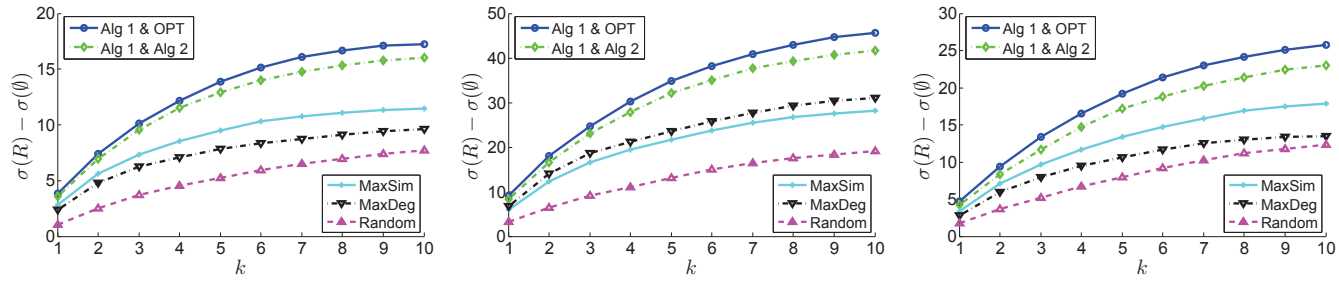
In this subsection, we report the evaluation results on friend recommendations. First, we investigate the impact of k (i.e., the number of recommended friends). We set $\alpha_v = 0.9$ and $\beta_v = 0.1$ for all people. v_0 is uniform-randomly picked from all the nodes. The results are averaged over 10,000 times for smoothness and are shown in Fig. 7. Three subfigures represent results in three different datasets, respectively. It can be seen that Alg 1 & OPT performs better than Alg 1 & Alg 2, since the former one computes the influence spread optimally at the cost of time consumption. However, the gap between their performances is limited, meaning that Alg 1 & Alg 2 estimates the influence spread accurately. Meanwhile, MaxSim outperforms MaxDeg, meaning that we are more likely to recommended friends-of-friends than strangers (users who are not friends or friends-of-friends). For all the algorithms, the increased influence spread, $\sigma(R) - \sigma(\emptyset)$, satisfies the diminishing return effect with respect to k. This is because people influenced by later recommendations have potential overlays with those influenced by earlier recommendations.

We also study the impact of v_0. For convenience, we classify users into two types by their outgoing degrees. If a user has an outgoing degree that is no larger than 100, it is called a normal user. Otherwise, it is called a popular user. We would like to see the difference between recommendations for normal users and those for popular users, as shown in Fig. 8. An interesting phenomenon is that, our recommendations are more effective for popular users than normal users on Facebook and Wiki, but are less effective for popular users than normal users in Epinions. This is because popular users are already influential in Epinions. We also observe that, for popular users, MaxDeg outperforms MaxSim in Facebook and Wiki, while MaxSim outperforms MaxDeg in Epinions. A possible explanation for the above phenomenon is that, when we intend to recommend influential strangers over friends-of-friends (MaxDeg outperforms MaxSim), users could greatly improve their social influence, since their influences are no longer limited to their current social circles. Recommendations for popular users are not necessarily more effective than those for normal users, depending on the application scenario.

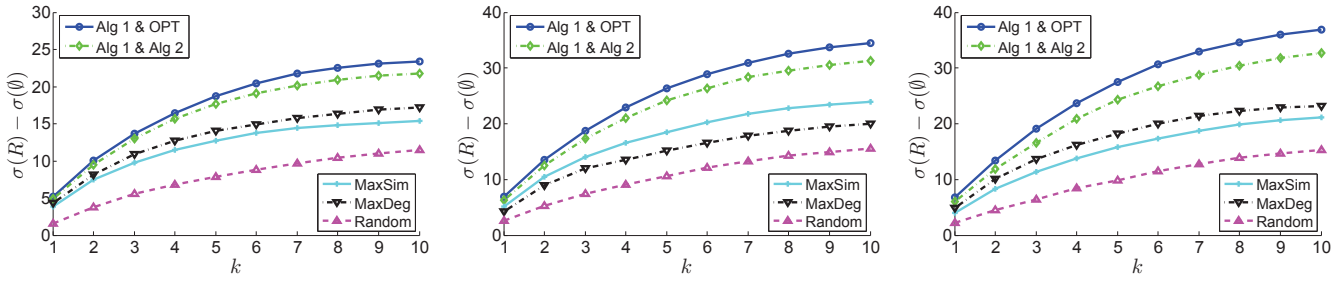


(a) Facebook. (b) Epinions. (c) Wiki.

Fig. 7. The evaluation results on the impact of k (the number of recommended friends).



(a) Facebook (normal users). (b) Epinions (normal users). (c) Wiki (normal users).



(d) Facebook (popular users). (e) Epinions (popular users). (f) Wiki (popular users).

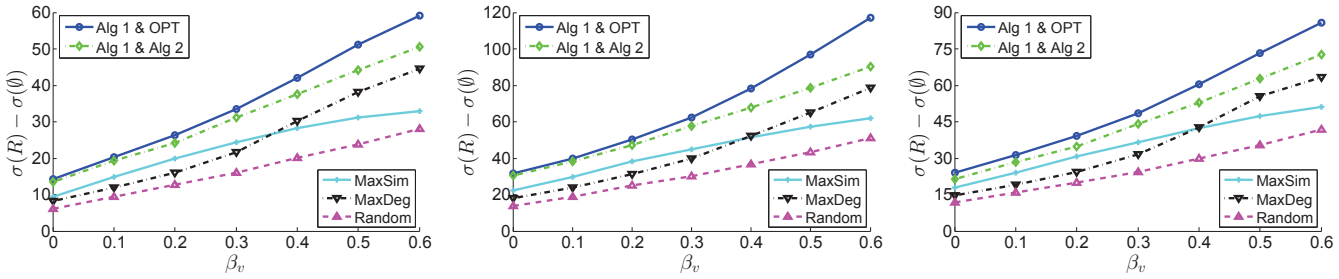
Fig. 8. The evaluation results on the impact of v_0 (normal users or popular users).

Finally, the impacts of α_v and β_v are studied. β_v is set to be the same for all people and then tuned. We set $\alpha_v = 1 - \beta_v$ and $k = 10$. v_0 is uniform-randomly picked from all the nodes. The results are averaged over 10,000 times for smoothness and are shown in Fig. 9. It can be seen that $\sigma(R) - \sigma(\emptyset)$ in all the algorithms increases with respect to β_v. MaxSim outperforms MaxDeg for small β_v, but MaxDeg outperforms MaxSim for large β_v. The reason is stated as follows. When β_v is small, we favor recommending friends-of-friends of v_0. For example, in Eq. 1, only friends-of-friends are recommended to v_0, when $\beta_v = 0$. On the other hand, when β_v is large, we would favor recommending influential strangers. This is because, when β_v is large, influential strangers are likely to accept the friend request from v_0 and then propagate v_0's influence. Fig. 10 shows the percentage of strangers recommended by Alg 1 & Alg 2. When $\beta_v = 0.1$, less than 30% of the recommendations are strangers. In contrast, when $\beta_v = 0.5$, about half of the recommendations are strangers.

D. Evaluation Results on Influence Spread Computations

This subsection evaluates the influence spread computation. v_0 is uniform-randomly picked from all the nodes. We use $k = 0$, i.e., the result is the number of people that v_0 can influence through existing friends. Users are classified into normal users and popular users, according to the same rule in the previous subsection. The impact of the control parameter, ε, is studied. For Algorithm 2, we set $\varepsilon = 0.1$ and $\varepsilon = 0.3$, respectively. They are denoted as Alg 2 (0.1) and Alg 2 (0.3). The results are also averaged over 10,000 times.

The evaluation results are shown in Fig. 11. On average, popular users have much larger influence spreads than normal users. The algorithm of Tree significantly underestimates the influence spread. Although DAG improves Tree by considering more paths, it still underestimates the influence spread. The latest approach of IMRank is outperformed by Alg 2 (0.1), since IMRank does not consider the number of paths in the influence spread computation. In contrast, Alg 2 (0.1) has an accurate estimation of the true influence spread, through considering the dominating influence propagation paths. Errors of Alg 2 (0.1) are within 10% for both normal users and popular users in the three datasets. The impact of the control parameter is also significant. Note that Algorithm 2 is expected to obtain a ratio, $1 - \varepsilon$, to the optimal algorithm (under some assumptions in Eqs. 5 and 10). A smaller ε brings a more



(a) Facebook. (b) Epinions. (c) Wiki.

Fig. 9. The evaluation results on the impact of α_v and β_v. We set $\alpha_v = 1 - \beta_v$.

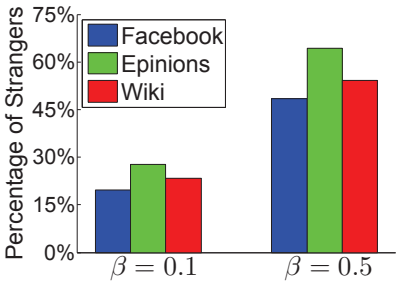
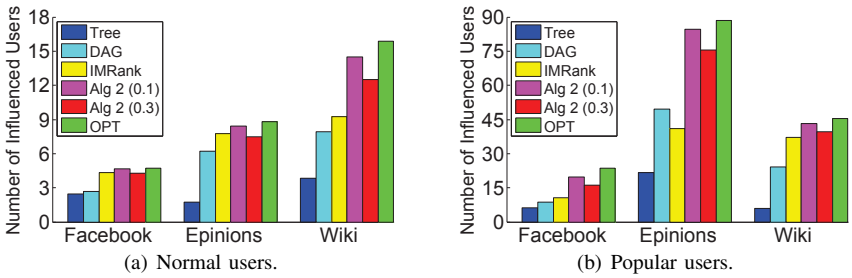


Fig. 10. Fractions of recommended strangers.



(a) Normal users. (b) Popular users.

Fig. 11. Evaluations of influence spread computation methods.

accurate computation of the influence spread. While $\varepsilon = 0.3$ brings an inaccurate influence spread computation, $\varepsilon = 0.1$ can bring a result that is close to the optimal algorithm.

VII. Conclusion

This paper studies the friend recommendation strategy with the perspective of social influence maximization. For the system provider, the objective is to recommend k new friends to a given user, such that the given user can maximize his/her social influence through new friends. Our problem is proved to be NP-hard. A greedy approximation algorithm with a ratio of $1 - \frac{1}{e}$ is proposed based on the submodular property. By considering the multipath effect, a novel method is proposed to accurately compute the influence spread. Experiments verify the efficiency and effectiveness of our approach.

References

[1] http://www.similarweb.com/global.

[2] http://www.pewinternet.org/fact-sheets.

[3] https://www.facebook.com/notes/facebook/people-you-may-know/15610312130.

[4] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, and X. He, "Music recommendation by unified hypergraph: combining social media information and music content," in *ACM Multimedia*, 2010.

[5] M. Ye, P. Yin, and W.-C. Lee, "Location recommendation for location-based social networks," in *ACM SIGSPATIAL*, 2010.

[6] D. Carnegie, *How to win friends and influence people*. Simon and Schuster, 2010.

[7] https://www.facebook.com/business/success/.

[8] https://business.twitter.com/success-stories/rukes.

[9] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *ACM SIGKDD*, 2003.

[10] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *ACM SIGKDD*, 2010.

[11] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *IEEE ICDM*, 2010.

[12] C. Budak, D. Agrawal, and A. El Abbadi, "Limiting the spread of misinformation in social networks," in *ACM WWW*, 2011.

[13] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *ACM-SIAM SODA*, 2014.

[14] X. Yang, H. Steck, and Y. Liu, "Circle-based recommendation in online social networks," in *ACM SIGKDD*, 2012.

[15] Y. Zhang, B. Cao, and D.-Y. Yeung, "Multi-domain collaborative filtering," *UAI*, 2010.

[16] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W.-Y. Ma, "Recommending friends and locations based on individual location history," *ACM Transactions on the Web*, 2011.

[17] M. B. Zafar, P. Bhattacharya, N. Ganguly, K. P. Gummadi, and S. Ghosh, "Sampling content from online social networks: Comparing random vs. expert sampling of the twitter stream," *ACM Transactions on the Web*, 2015.

[18] M. Ye, X. Liu, and W.-C. Lee, "Exploring social influence for recommendation: a generative model approach," in *ACM SIGIR*, 2012.

[19] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, "Time-aware point-of-interest recommendation," in *ACM SIGIR*, 2013.

[20] V. V. Vazirani, *Approximation algorithms*. Springer Science & Business Media, 2013.

[21] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, 1978.

[22] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng, "IMRank: influence maximization via finding self-consistent ranking," in *ACM SIGIR*, 2014.

[23] G. Mao and N. Zhang, "Analysis of average shortest-path length of scale-free network," *Journal of Applied Mathematics*, 2013.

[24] G. Song, X. Zhou, Y. Wang, and K. Xie, "Influence maximization on large-scale mobile social network: a divide-and-conquer method," *IEEE Transactions on Parallel and Distributed Systems*, 2015.

[25] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, 1971.

[26] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *ACM WOSN*, 2009.

[27] J. Tang, H. Gao, H. Liu, and A. Das Sarma, "etrust: Understanding trust evolution in an online world," in *ACM SIGKDD*, 2012.

[28] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *ACM WWW*, 2010.

[29] P. Chalermsook, A. Das Sarma, A. Lall, and D. Nanongkai, "Social network monetization via sponsored viral marketing," in *ACM SIGMETRICS*, 2015.

[30] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized pagerank on mapreduce," in *ACM SIGMOD*, 2011.