# Influence Maximization in Real-World Closed Social Networks

Shixun Huang[†], Wenqing Lin[‡], Zhifeng Bao[†], Jiachen Sun[‡]
[†]RMIT University, [‡]Tencent
[†]{shixun.huang,zhifeng.bao}@rmit.edu.au, [‡]{edwlin,jiachensun}@tencent.com

## ABSTRACT

In the last few years, many closed social networks such as WhatsAPP and WeChat have emerged to cater for people's growing demand of privacy and independence. In a closed social network, the posted content is not available to all users or senders can set limits on who can see the posted content. Under such a constraint, we study the problem of influence maximization in a closed social network. It aims to recommend users (not just the seed users) a limited number of *existing* friends who will help propagate the information, such that the seed users' influence spread can be maximized. We first prove that this problem is NP-hard. Then, we propose a highly effective yet efficient method to augment the diffusion network, which initially consists of seed users only. The augmentation is done by iteratively and intelligently selecting and inserting a limited number of edges from the original network. Through extensive experiments on real-world social networks including deployment into a real-world application, we demonstrate the effectiveness and efficiency of our proposed method.

**PVLDB Reference Format:**
Shixun Huang[†], Wenqing Lin[‡], Zhifeng Bao[†], Jiachen Sun[‡]. Influence Maximization in Real-World Closed Social Networks. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/rmitbggroup/IMCSN.

## 1 INTRODUCTION

Social network platforms have been a popular way that allows people to keep in touch with friends and share contents. There are many *open* social networks where the posted content of a user will be available to all followers and even non-followers using search engines. This open sharing model is popular with millennials who are heavily impacted by the Fear Of Missing Out culture and like to put their lives on display and see everything that's going on [2].

Recently, *closed* social networks, where sharing is limited to selected persons only, have emerged and are favored by generation Z to cater for privacy issues and information overload in open social networks. At Tencent, the closed sharing model has become a predominant form for sharing information in its numerous applications. For example, users will send a limited number of friends recent news and updates (e.g., COVID-19 statistics and reports of the Olympic winter games) on WeChat, charitable activities on QQ, and event invitations and sales promotion of merchandise on Tencent Shop. Meanwhile, many tech giants (e.g., Facebook, LinkedIn and Pinterest) have started enabling closed sharing models with personalized settings as well [3, 6, 9, 10]. Additionally, the closed social network platform WhatsApp recently took the first place from Facebook in downloads [4], which again demonstrates the importance and popularity of the closed sharing model.

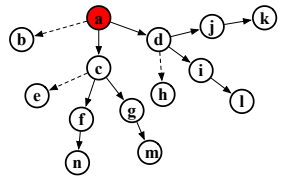



**Figure 1: The network formed by solid edges is the optimal diffusion network to maximize the influence of $a$ if each user only shares information with up to two friends.**

On the one hand, in closed social networks, users often do not want to overexpose themselves for various reasons, such as privacy concerns and psychological factors [5], or may have limited sharing opportunities due to resource constraints in events. Therefore, users are likely to share information with only a limited number of friends. For example, (1) users in WeChat tend to share their life moments or private matters only with their close friends and families, rather than colleagues with pure business relationships [1, 7]; (2) users who spend on online Tencent gaming events may obtain limited virtual coupons for sharing and they need to carefully choose friends to receive the coupons since people who use the coupons can increase the discounts of future purchase of the coupon sender [8].

Considering that online users have different capabilities of propagating information, as a consequence, the influence spread (i.e., information propagation effect) of a seed user (i.e., a user who releases the content at the very beginning) heavily depends on the friends she chooses to receive the information. Since the information propagation unfolds in a cascading manner, an information receiver can be multiple hops away from the seed and her selected friends for sharing also impacts the influence spread.

On the other hand, users hope that their released information could reach and benefit most of the users. Therefore, to fully realize the social and commercial value of the information with the greatest exposure in closed social networks, an effective strategy of recommending friends to share information is very important. Since each recommendation corresponds to a directed edge in the network, all recommendations constitute a *diffusion network*, which is essentially a subnetwork of the original network and has limit on the number of outgoing edges (i.e., paths from senders to receivers) incident on each node.


Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

In this paper, we formulate the above recommendation as the problem of Influence Maximization in Closed Social Networks (IMCSN), which aims to find the optimal diffusion network via which the seeds' influence spread is the maximum.

Example 1. *To make this problem more intuitive, we use an example for illustration. Figure 1 shows a toy network where each user will be recommended a limited number of friends, say two, to share information. Suppose we only consider the information diffusion via the friends recommended by the system and ignore the uncontrollable and unpredictable sharing with friends that are not in the recommendation list. Given the seed user $a$ who has three friends $b$, $c$ and $d$, she will send the information to two of her friends, say $c$ and $d$. Afterwards, her friends who receive the information may also choose two of their friends (e.g., the friends $f$ and $g$ of $c$) to receive this information. This process is repeated such that the information diffuses in cascades. If we assume that users will propagate the received information unconditionally, the diffusion network formed by all solid edges refer to the optimal solution where the information from the seed user $a$ can reach the maximum number of users.*

Notably, traditional influence maximization [46] in open social networks assumes that a user will send information to all her friends, and aims to select a limited number of seeds with the maximum influence spread. Here, we focus on the closed sharing model and selecting a limited number of edges such that the influence spread of *specific seeds* via these *selected edges* is maximized, which is different from the former. Please refer to Section 2 for more details.

It is very challenging to solve the IMCSN problem because the choice space is extremely large. To solve this problem, we try to insert limited links from the original network into the diffusion network that is initially empty. Unfortunately, we find that greedily inserting edges with the maximum marginal gains w.r.t. influence of seeds is not effective, due to the non-submodular property of the objective function and the expensive marginal gain computations under the classical Independent Cascade (IC) model [46]. Thus, we resort to computing and leveraging influence lower bounds, and by using such lower bounds the submodular property can be preserved. Afterwards, we propose a novel diffusion network augmentation method which consists of two stages, namely the expansion stage and the filling stage. In the expansion stage, we iteratively expand the diffusion network size by incorporating the users and the connections that are important to spreading influence. In the filling stage, we intelligently fill up link recommendations for these involved users. Our contributions are summarized as below:

- We make the first attempt to study and formalize the problem of Influence Maximization in *Closed* Social Networks (IMCSN), motivated by numerous practical needs in social network platforms such as Tencent. We also prove its NP-hardness (Section 3).
- With an influence lower bound as the quality measurement, we first propose a network augmentation *sketch* that can produce solutions for a single seed user with theoretical guarantees in a given diffusion network (Section 4.1). To boost the efficiency, we propose an effective yet scalable network augmentation method to avoid marginal gain computations (Section 4.2).
- We make some interesting observations in the influence diffusion of multiple seed users and leverage these observations to transform the IMCSN problem for multiple seed users into the one for a single 'virtual' seed user. Such a transformation enables us to utilize our method for a single seed user with minor adjustments, to produce effective solutions while maintaining high efficiency.
- We conduct extensive experiments on real-world social networks including the deployment into a Tencent application (Section 6). We have several exciting findings:
  1) In the IMCSN problem, the boosted baselines, built upon the diffusion network produced by the expansion stage of our method, can achieve up to five-orders-of-magnitude larger influence spread than their counterparts built upon the initially empty network. Despite that, our full-stage method, which includes both the expansion and filling stages, significantly beats those boosted baselines.
  2) Our full-stage method is able to identify important connections for spreading influence – it is able to build a diffusion network where seed users can achieve 90% of their full influence in the original network and this diffusion network contains only 36% of edges from the original one.
  3) We deploy our solution into an activity of an online Tencent application where each online user will be recommended some friends for interaction. We conduct online A/B testing where each user is randomly assigned to one method which produces the recommendation list for this user. Such interaction can unfold in cascades and further trigger more interaction if the recommendation is effective. The online result shows that the recommendation from our solution which achieves a notably better Click-through Rate than the rest of baselines.
  4) In solving the problem of maximizing the seeds' influence over *open* social networks, by recommending limited *new links*, our method achieves up to five-orders-of-magnitude speedup than the state-of-the-art while maintaining competitive effectiveness.

## 2 RELATED WORK

In this section, we will first describe the difference between closed and open social networks. Afterwards, we will describe the related work on influence maximization in *open* social networks, which can be broadly divided into two categories, influence maximization via node selection and influence maximization via edge insertion, respectively. Then, we will discuss the differences between them and our problem in *closed* social networks.

**Open vs. Closed Social Networks**. The two words 'open' and 'closed' are used to describe the underlying sharing model rather than the topology of the social network. Furthermore, the sharing system applied to the network is decided based on the specific application behind. For example, if Figure 1 describes a subgraph of the Twitter social network, we have an open sharing model by default and a user's post will be available to all online users. On the other hand, if Figure 1 describes a subgraph of the Wechat social network, the closed sharing model will be applied in most cases (as in Example 1) where users will make their messages or posts visible to a limited number of selected friends. For ease of presentation and following the naming convention in this domain [11, 28], we directly use 'open' and 'closed' social networks to refer to the networks with the 'open' and 'closed' sharing model respectively.

**Influence Maximization via node selection**. This problem refers to the classical influence maximization problem that aims to choose

a limited number of seed nodes with the greatest influence spread. Kempe et al. [46] prove the NP-hardness of this problem and propose a Monte Carlo simulation based greedy algorithm which iteratively chooses the node with the greatest marginal gain to the influence. Due to the importance of this problem, many subsequent studies [15–20, 22, 24–27, 33, 36, 37, 39–45, 48–50, 52, 53, 55, 57–61, 65] have been proposed to further improve the efficiency and/or effectiveness. They mainly differ in how the influence spread is defined or estimated. To name a few, Leskovec et al. [48] adopt the Monte-Carlo simluation to measure the influence spread and speed up the process in [46] with an early termination technique based on the submodular property of the influence function. Ohsaka et al. [55] measure the influence and marginal gain based on a limited number of subgraphs generated by the flipping-coin technique [46], which further improves the efficiency. Borgs et al. [19] leverage reverse reachable sets to estimate influence and inspires more recent advanced solutions [58, 59].

**Influence maximization via edge insertion**. The work falling in this category aims to insert a limited number of edges into the network so as to maximize influence spread of specific seeds. D'Angelo et al. [32] study how to maximize the influence of the seeds under the IC model by adding a limited number of edges incident to these seeds. Coró et al. [31] extend the results of [32] to the Linear Threshold (LT) model [38]. Specifically, they prove that the objective function under the LT model is submodular and leverage this property to propose an approximate algorithm. Khalil et al. [47] study how to add a limited number of edges to maximize the influence of given seeds under the LT model. Chaoji et al. [23] study how to add edges to maximize the influence spread of seeds under the constraint that at most $k$ inserted edges are incident on any node. Yang et al. [64] and Yang et al. [63] study how to add a limited number of edges from a candidate set to maximize the seeds' influence under the IC model. Yang et al. [64] derive a lower and upper bound influence function respectively to approximate the non-submodular influence under the IC model, and they use a sandwich strategy to produce approximate solutions. Yang et al. [63] derive tighter bounds than Yang et al. [64] and produce more effective results.

*Differences.* The aforementioned studies are drastically different from ours. The main reason is on the problem setting and assumption. The studies of the first category focus on selecting *seeds* based on the open social networks where the influence from the seeds is allowed to propagate via *any* edge in the network. On the other hand, we consider the *closed* social networks, and focus on selecting limited *edges* from the original network such that the influence of *specific seeds* via these *selected edges* is maximized.

The studies of the second category focus on inserting a limited number of *new* edges into the *existing* network and assume an *open-sharing* model such that the influence of the seeds will spread via all edges including the inserted ones. In contrast, we focus on inserting a limited number of *existing* edges from the original network into an initially empty network and consider a *closed-sharing* model such that the influence of seeds will spread *only* via inserted edges. As a consequence, existing work's decision making of edge insertion is based on the current graph structure and cannot be trivially extended to handle our problem, because the network which requires edge insertion in our problem is initially empty. Additionally, due to different assumptions of the sharing model, the space of candidate edges for insertion that is considered by existing work is significantly smaller than that of our problem. As a result, most existing methods will suffer from serious scalability issue even if they could be extended to handle our case. In particular, the infeasibility of extending existing work can be caused by reasons including but not limited to: (1) Most of these studies do not consider the edge insertion constraint where the number of inserted edges sharing the same source node cannot be greater than $k$. (2) The methods in [31, 32] are designed based on the assumption that the source nodes of all edge candidates must be seeds, which is not the case in our problem. (3) The method in [47] is specifically designed for the LT model which has drastically different properties from the IC model we are considering. (4) The method in [23] cannot help the seed nodes to influence the nodes that are 2-hop away in the initially empty network, because the influence path between any pair of nodes is not allowed to contain more than one inserted edge. (5) The methods in [63, 64] hold a strong assumption that the network is acyclic, which is often not true in real-world scenarios.

## 3 PROBLEM FORMULATION

In this paper, we consider both directed and undirected social network where the latter can be transformed into the directed one. In an undirected network, each edge represents a friendship between two users. Since information diffusion is directed, each friendship corresponds to two diffusion directions. Thus, we represent the undirected social network as the directed one $G = (V, E)$ where $V$ ($E$) is the node (edge) set, and each directed edge from $u$ to $v$ is denoted as $(u, v)$. We use $N^G_{in}(u)$ and $N^G_{out}(u)$ to denote the set of incoming and outgoing neighbors of $u$, respectively.

**The diffusion model**. We focus on a classic and widely-adopted information diffusion model – the *Independent Cascade* (IC) model [46]. It originates from the marketing literature [34] and independently assigns each edge $(u, v)$ with an influence probability $p_{u,v} \rightarrow [0, 1]$. Given a seed node $s$ being active at time step 0 and the influence probabilities, the diffusion unfolds in discrete steps. Each active node $u$ in time step $t \geq 1$ will have a single chance to activate each outgoing neighbor $v$, that is inactive in step $t-1$, with a probability of $p_{u,v}$. If an outgoing neighbor $v$ is activated in step $t$, it will become active in step $t+1$ and then will have a single chance to activate each of its inactive outgoing neighbors in the next time step. The diffusion instance terminates when no more nodes can be activated. The *influence spread* $\delta_G(s)$ in a graph $G$ is the expected number of activated nodes with $s$ as the seed node. Note that our proposed methods are specifically designed for the IC model. The extension on other models (e.g., the Linear Threshold model [46]) is out of the scope of this work and will be explored in future work.

In this paper, we aim to select at most $k$ outgoing edges for each online user such that the influence spread of seeds via these selected edges is maximized. These directed edges naturally form a diffusion sub-network defined as below.

Definition 1 (K-subnetwork). *Given a directed network $G = (V, E)$ and an integer $k$, a $k$-subnetwork $G_k$ is a subgraph of $G$ where there are at most $k$ outgoing neighbors for each $v \in G_k$. Formally, $G_k = (V_k, E_k)$ where $V_k \subseteq V$, $E_k \subseteq E$ and $\forall v \in V_k, |N^{G_k}_{out}(v)| \leq k$.*

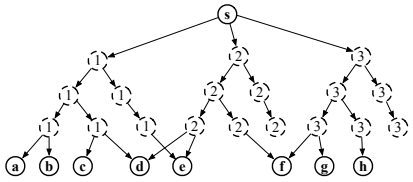



**Figure 2: An example of a graph constructed based on a set cover instance.**

For example, in Figure 1, the network that consists of solid edges only is a 2-subnetwork. Notably, based on industry practice, we make three considerations when formulating our problem (i.e., Definition 2): 1) The value of $k$ is the same for all users and is decided by the event operator. However, our solutions can easily work with the scenario where the value of $k$ depends on specific users. 2) Seed users are independent since KOLs (i.e., key opinion leaders) may have unpredictable and different information to spread over a long period, and recommendations based on this consideration help maintain a long-term usage of the recommendation system. 3) The friend recommendation list for a user is built by considering all seed users instead of being customized for individuals since (i) it is intractable to know, among all seed users sharing the same content, who will activate online users for further information propagation in real-world scenarios, and (ii) fixing the recommendations help increase the interaction rate between users with a laser focus on facilitating information diffusion.

Definition 2 (Influence maximization in closed social networks (IMCSN)). *Given a directed social network $G = (V, E)$, an integer $k$, a set $S$ of independent seed users, we aim to find the optimal diffusion $k$-subnetwork $G_k^*$ such that the aggregated influence of seed users in $S$ is maximized under $G_k^*$:*

$$G_k^* = \arg\max_{G_k \in \mathbb{S}} \sum_{s \in S} \delta_{G_k}(s)$$

*where $\mathbb{S}$ refers to the whole space of all possible $k$-subnetworks of $G$.*

If $s \notin V_k$, the influence of $s$ in $G_k$ is 0. When the context is clear, we omit the subscript of $\delta_{G_k}(\cdot)$, and we use the terms 'node' and 'user', as well as the terms 'edge' and 'link' interchangeably.

**Hardness analysis of the IMCSN**. We will show that the IMCSN problem is NP-hard via a reduction from the set cover problem defined as below.

Definition 3 (Set Cover). *Given a set $U$ of elements, a collection $C$ of subsets $A_1, A_2, ..., A_{|C|}$ of $U$ where $\cup_{1 \le i \le |C|} A_i = U$, we aim to choose the smallest number of sets from $C$ such that their union is $U$.*

Theorem 1. *The IMCSN problem is NP-hard.*

Proof. To show that IMCSN is NP-hard, we will perform a reduction from the NP-complete decision problem of set cover, which checks whether we can find $k$ sets from $C$ whose union is $U$, to the special case of IMCSN where there is only one seed user $s$.

Given the collection $C$ of a set cover instance, we construct a deterministic graph with a tree structure where the weight of each edge is 1 as below. First, we find the set $A_{max}$ with the greatest size from $C$ and create a tree with elements in $A_{max}$ as the bottom nodes (at the bottom level). In the second to last level, we introduce the set $V'$ of $\lceil |A_{max}|/k \rceil$ virtual nodes. We next create directed edges from each of the virtual nodes to at most $k$ arbitrary nodes in $A_{max}$ which have not been connected by virtual nodes in $V'$. Afterwards, we create the third to last level based on the second to last level by adopting the similar rules. We repeat this step until the root node is constructed. We also create trees for the rest of the sets in $C$ similarly. One difference is that, when we create the tree for a set $A$, we need to make sure that the number of virtual nodes in each level is consistent with the corresponding level of the tree constructed based on $A_{max}$. It means that some virtual nodes may not have outgoing neighbors. After we create trees for all sets in $C$, we introduce directed edges from $s$ to all root nodes.

Figure 2 shows an example where a graph is constructed from the set cover instance where $k = 2$ and $C = \{\{a, b, c, d, e\}, \{d, e, f\}, \{f, g, h\}\}$. Dashed nodes refer to virtual nodes. Virtual nodes with the same number come from the tree constructed based on the same set from $C$. Since $A_{max} = \{a, b, c, d, e\}$, the number of virtual nodes in the second to last level of all trees are $\lceil |A_{max}|/k \rceil = 3$.

Since the time cost for creating such a tree is at most $O(|A_{max}| \log_k |A_{max}|)$, the total reduction process is polynomial in the total size of the sets in the collection. To solve the IMCSN problem in the constructed graph, we only need to focus on selecting $k$ outgoing edges for the root node $s$ since the out-degree of the rest of the nodes in the graph is at most $k$ and edges not incident on $s$ can all be chosen. Considering that the trees constructed from each set in $C$ have the same number of virtual nodes, the quality of the outgoing edge selection for $s$ only depends on bottom nodes being reached. Thus, selecting $k$ outgoing edges for $s$ corresponds to selecting $k$ sets from $C$. Suppose the number of virtual nodes in each tree is $x$. If the optimal influence spread is $k \times x + 1 + |U|$ (including $s$) which is also the maximum possible influence, we have a 'Yes' answer to the set cover instance. Otherwise, the answer is 'No'. Therefore, if we can find the optimal solution for the IMCSN instance in polynomial time, the set cover problem can be solved in polynomial time which is not possible unless P=NP. Thus, the IMCSN problem is NP-hard. □

## 4 SOLVING IMCSN FOR A SINGLE SEED

To facilitate the illustration of our methodology, we first describe how to solve the IMCSN problem for a single seed user. In the next section, we will describe how to extend the method to handle multiple seed users. Essentially, we try to solve the IMCSN problem by inserting edges from the original network into a $k$-subnetwork which initially contains the seed user only. As mentioned in Section 1, we resort to the influence lower bounds to preserve submodularity, and the lower bounds are computed with the Restricted Maximum Probability Path (RMPP) model [23] as defined below.

Definition 4 (Influence Probability of a path). *The influence probability of a path is the product of influence probabilities of all edges in this path.*

Definition 5 (Restricted Maximum Probability Path (RMPP) [23]). *Given a $k$-subnetwork $G_k$ where edges are either native (i.e., originally exist) or inserted, and a seed node $s$, the restricted maximum probability path $RMPP(s, u)$ is the directed path from $s$ to $u$ whose probability $p^{RP}_{(s,u)}$ is the maximum among all paths from $s$ to $u$ containing at most one inserted edge. Ties are broken arbitrarily.*

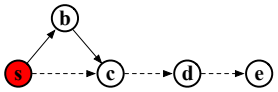



**Figure 3: A simple network where dashed edges are candidates for insertion.**

**Algorithm 1:** SubnetworkAugmentationSketch

**Input** : The input network $G$, the seed user $s$ and an integer $k$.
**Output**: The $k$-subnetwork $G_k$.

$G_k = (V_k.E_k) \leftarrow k$-subnetwork containing the seed $s$ only;
**while** $\exists v \in V_k, |N^{G_k}_{out}(v)| < min(k, |N^{G}_{out}(v)|)$ **do**
  $G_k$ = EdgeInsertionSketch($G$, $G_k$, $s$, $k$);
Return $G_k$;

Definition 6 (RMPP model [23]). *In the RMPP model, the influence of a seed node $s$ to a node $u$ is the influence probability $p^{RP}_{(s,u)}$ of RMPP$(s,u)$, and the total influence of $s$ is the sum of influence probabilities of all RMPPs from $s$ to the rest of nodes in the graph.*

Example 2. *As shown in Figure 3, we have a network which contains five nodes and two native edges $(s,b)$ and $(b,c)$ with the same influence probability of 0.5. Suppose the seed node is $s$, three edge candidates $(s,c)$,$(c,d)$ and $(d,e)$ have the same influence probability of 1, $T_1 = \emptyset$ and $T_2 = \{(s,c),(c,d)\}$. The marginal gain of $(s,c)$ over $T_1$ is 1-0.25=0.75 since RMPP$(s,c)$ is edge $(s,c)$. However, the marginal gain of $(d,e)$ over $T_2$ is 0 since $s$ needs to traverse at least two inserted edges to reach $e$, which is not allowed in the RMPP model.*

RMPP provides an efficient lower bound estimation of influence spread under the IC model and preserves the submodularity [23]. Unfortunately, directly adopting the RMPP model to insert edges into the $k$-subnetwork which initially contains only the seed node $s$ does not help to achieve great influence spread. This is because $s$ has to traverse two inserted edges to reach the nodes that are two hops away, but this is not permitted in the RMPP model, as illustrated in Example 2.

To mitigate this issue without missing the opportunity of leveraging the submodularity property of the RMPP model, we propose a subnetwork augmentation sketch in Algorithm 1. Specifically, this sketch iteratively calls a strategy called EdgeInsertionSketch in the pseudocode. This strategy tries to insert edges to increase the influence spread of the seed node $s$ under the RMPP model and uses these inserted edges to update the input $k$-subnetwork $G_k$. After reaching a termination condition, these *inserted* edges will be treated as *native* edges of $G_k$ in the next iteration, which will help $s$ to influence the nodes that are many hops away. In what follows, we will introduce how the method EdgeInsertionSketch works and then propose a much more efficient yet practical edge insertion strategy (Section 4.1). Afterwards, we will further optimize this subnetwork augmentation sketch (Section 4.2).

## 4.1 Edge Insertion

Given a candidate network $G_c$ that contains candidate edges to be inserted into the current $k$-subnetwork $G_k$, the most straightforward way to maximize the influence of $s$ is to greedily insert an edge with the maximum marginal gain into $G_k$ without breaking

**Algorithm 2:** EdgeInsertionSketch

**Input** : The input network $G = (V, E)$, the current $k$-subnetwork $G_k = (V_k, E_k)$, the seed user $s$ and an integer $k$.
**Output**: The updated $k$-subnetwork $G_k$.

$C \leftarrow$ candidate set initialized as $\emptyset$;
**foreach** *node* $u \in G_k$ **do**
  **if** $|N^{G_k}_{out}(u)| < min(k, |N^{G}_{out}(u)|)$ **then**
    **foreach** $(u, v) \in E$ **do**
      **if** $(u, v) \notin E_k$ *and* $v \neq s$ **then**
        Add $(u, v)$ into $C$
$G_c = \{V_c, E_c\} \leftarrow$ construct candidate graph based on $C$;
Compute the probabilities of SMPPs from $s$ to all nodes in $G_k$ ;
**foreach** *node* $v \in V_c$ *with* $|N^{G_c}_{in}(v)| > 0$ **do**
  $u^*_v = \arg\max_{u \in N^{G_c}_{in}(v)} p^{SP}_{(s,u)} \cdot p_{(u,v)}$;
**while** $G_c$ *has nodes with incoming neighbors* **do**
  $v^* = \arg\max_{v \in V_c, |N^{G_c}_{in}(v)|>0} \delta^{\triangle}_{G_k \cup \{(u^*_v, v)\}}(s) - \delta^{\triangle}_{G_k}(s)$;
  Remove all incoming edges to $v^*$ from $G_c$;
  Insert $(u^*_{v^*}, v^*)$ into $G_k$;
  **if** $|N^{G_k}_{out}(u^*_{v^*})| = min(k, |N^{G}_{out}(u^*_{v^*})|)$ **then**
    **foreach** $v \in N^{G_c}_{out}(u^*_{v^*})$ **do**
      **if** $u^*_v = u^*_{v^*}$ **then** Update $u^*_v$ ;
      Remove $(u^*_{v^*}, v)$ from $G_c$;
Mark all inserted edges as native;
**return** $G_k$;

the constraint of $k$-subnetwork. This approach suffers from serious scalability issues since it needs to update the marginal gain of all candidate edges in each iteration and the marginal gain computation of each edge incurs $O((V_k + E_k) \log V_k)$ time complexity with a variant of Dijkstra algorithm as in [23]. To address this issue, we first present an edge insertion sketch from the theoretical point of view, followed by a practical insertion method.

*4.1.1 Edge Insertion Sketch.* Later in Theorem 2, we prove that for all candidate edges that share the same destination node, we can carefully insert the *critical* edge (which will be defined in Definition 8) such that inserting other edges after this critical edge will not increase the influence spread of $s$ under the RMPP model. With this observation, we only need to compare the marginal gain of critical edges with different destination nodes in each iteration. Hence, the cost of marginal gain computation can be greatly reduced. To facilitate the presentation of our method, we use $\delta^{\triangle}()$ as the influence under the RMPP model, and introduce a concept, namely *Strict Maximum Probability Path (SMPP).*

Definition 7 (Strict Maximum Probability Path (SMPP)). *Given a seed node $s$ and a $k$-subnetwork $G_k$ where edges are either native or inserted, the strict maximum probability path SMPP$(s,u)$ is a directed path from $s$ to $u$ whose probability $p^{SP}_{(s,u)}$ is the maximum among all the paths from $s$ to $u$ containing native edges only (i.e., a special case of RMPP). Ties are broken arbitrarily.*

Example 3. *As shown in Figure 3, SMPP$(s,c)$ consists of two native edges, namely $(s,b)$ and $(b,c)$. SMPP$(s,c)$ remains unchanged even after we insert edge $(s,c)$, since the inserted edge is not native.*

The algorithm to be introduced is based on Theorem 2 which is in turn established upon Lemma 1 and Lemma 2 below.

Lemma 1. *Given a $k$-subnetwork $G_k$, a seed node $s$, and two candidate edges $(u, v)$ and $(u', v)$ which are in the candidate network $G_c$ and share the same endpoint $v$, and $p^{RP}_{(s,x)}(u, v)$ which denotes the probability of $RMPP(s, x)$ after inserting the edge $(u, v)$ into $G_k$, if $p^{SP}_{(s,u)} \cdot p_{(u,v)} \geq p^{SP}_{(s,u')} \cdot p_{(u',v)}$, then for any node $x$ which can be reached by $v$ via directed paths, $p^{RP}_{(s,x)}(u, v) \geq p^{RP}_{(s,x)}(u', v)$.*

Proof. If the $RMPP(s, x)$ from $s$ to $x$ remains the same after inserting $(u, v)$ and $(u', v)$, the influence probability of $x$ will remain unchanged, i.e., $p^{RP}_{(s,x)}(u, v) = p^{RP}_{(s,x)}(u', v)$.

If the $RMPP(s, x)$ from $s$ to $x$ is changed after inserting $(u, v)$ or $(u', v)$, the $RMPP$ must contain $(u, v)$ or $(u', v)$. Let $RMPP_{(u,v)}$ and $RMPP_{(u',v)}$ denote the *updated RMPP* from $s$ to $x$ by inserting edge $(u, v)$ and $(u', v)$ respectively. Each of these two $RMPP$s consist of two sub-paths. That is, $P_1(RMPP_{(u,v)}) = [s, ..., u, v]$, $P_2(RMPP_{(u,v)}) = [v, ..., x]$, $P_1(RMPP_{(u',v)}) = [s, ..., u', v]$ and $P_2(RMPP_{(u',v)}) = [v, ..., x]$. Based on Definition 5, the second paths of both $RMPP$s do not contain inserted edges. Thus, $P_2(RMPP_{(u,v)})$ and $P_2(RMPP_{(u',v)})$ have the same probability. Since $p^{SP}_{(s,u)} \cdot p_{(u,v)}$ and $p^{SP}_{(s,u')} \cdot p_{(u',v)}$ refer to the probability of the first sub-paths of these two RMPPs respectively and the former one is not smaller than the latter, we have $p^{RP}_{(s,x)}(u, v) \geq p^{RP}_{(s,x)}(u', v)$. □

Lemma 2. *Given a $k$-subnetwork $G_k$, a seed node $s$, and two candidate edges $(u, v)$ and $(u', v)$ which are in the candidate network $G_c$ and share the same endpoint $v$, if $p^{SP}_{(s,u)} p_{(u,v)} \geq p^{SP}_{(s,u')} p_{(u',v)}$, $\delta^{\triangle}_{G_k \cup \{(u,v)\}}(s) = \delta^{\triangle}_{G_k \cup \{(u,v),(u',v)\}}(s)$.*

Proof. Based on Lemma 1, for any node $x$ that can be reached by $v$, the influence probability $p^{RP}_{(s,x)}$ of $RMPP(s, x)$ under $G_k \cup \{(u, v)\}$ is never smaller than that under $G_k \cup \{(u', v)\}$. Thus, introducing $\{(u', v)\}$ after inserting $(u, v)$ does not contribute to the influence increment of $s$. Thus, $\delta^{\triangle}_{G_k \cup \{(u,v)\}}(s) = \delta^{\triangle}_{G_k \cup \{(u,v),(u',v)\}}(s)$. □

To better describe Theorem 2 later, we have the definition below.

Definition 8 (critical neighbor and edge). *Given a $k$-subnetwork $G_k$, the seed node $s$, the candidate network $G_c$ and a node $v$, if $|N^{G_c}_{in}(v)| > 0$, the critical neighbor $u^*_v$ of $v$ in $G_c$ is the neighbor whose SMPP path probability $p^{SP}_{(s,u^*_v)}$ times the influence probability $p_{(u^*_v,v)}$ of the edge $(u^*_v, v)$ is the greatest among all the incoming neighbors of $v$ in $G_c$. That is, $u^*_v = \arg\max_{u \in N^{G_c}_{in}(v)} p^{SP}_{(s,u)} \cdot p_{(u,v)}$. Correspondingly, the incoming edge $(u^*_v, v)$ is the critical edge of $v$.*

Theorem 2. *Given a $k$-subnetwork $G_k$, the seed node $s$, a node $v$, a candidate set $E' = \{(u, v) | u \in N^{G_c}_{in}(v)\}$ for insertion and the critical edge $(u^*_v, v)$, we have: $\delta^{\triangle}_{G_k \cup E'}(s) = \delta^{\triangle}_{G_k \cup \{(u^*_v,v)\}}(s)$.*

Proof. Since $(u^*_v, v)$ is the critical edge, $p^{SP}_{(s,u^*_v)} \cdot p_{(u^*_v,v)}$ is the greatest among all the incoming neighbors of $v$ in $G_c$. Based on Lemma 2, we can know that, after inserting $(u^*_v, v)$, introducing edges sharing the same destination node as $(u^*_v, v)$ will not increase the RMPP-based influence. Thus, this theorem is deduced. □

Algorithm 2 describes the process of edge insertion sketch by leveraging Theorem 2. Lines 1-7 construct the candidate graph $G_c$ containing the candidate edges which can be inserted into $G_k$. $G_c$ only contains the edges between nodes in $G_k$ and their neighbors in $G$, since nodes in $G_k$ cannot reach their two-hop neighbors in $G$ with only one inserted edge. Lines 8-10 compute the critical incoming neighbor for each node in $G_c$, and Lines 11-18 iteratively insert the critical edge with the maximum marginal gain into $G_k$ until all the critical edges have been inserted.

**Algorithm 3:** PracticalEdgeInsertion (PEI)

**Input :** The input network $G = (V, E)$, the current $k$-subnetwork $G_k = (V_k, E_k)$, the seed user $s$ and an integer $k$.
**Output :** The updated $k$-subnetwork $G_k$.

1 $G_c = \{V_c, E_c\} \leftarrow$ constructs the candidate graph as in Algorithm 2;
2 $L \leftarrow$ nodes in $G_k$ excluding $s$, ranked by their subtree sizes of in the SMPP-based tree in non-increasing order;
3 $L' \leftarrow$ nodes, that are in $G_c$ but not in $G_k$, ranked by their out-degrees in non-increasing order;
4 $L \leftarrow$ Append $L'$ to the end of $L$;
5 **foreach** $v$ *in* $L$ **do**
6  **if** $|N^{G_c}_{in}(v)| = 0$ **then** Continue;
7  $u^*_v = \arg\max_{u \in N^{G_c}_{in}(v)} p^{SP}_{(s,u)} \cdot p_{(u,v)}$;
8  Add $(u^*_v, v)$ into $G_k$, and remove $(u^*_v, v)$ from $G_c$;
9  **if** $|N^{G_k}_{out}(u^*_v)| = min(k, |N^{G}_{out}(u^*_v)|)$ **then**
10  Delete all outgoing edges of $u^*_v$ from $G_c$.
11  Mark all inserted edges as native;
12 **return** $G_k$;

**Theoretical Guarantee**. Maximizing the influence spread of $s$ in the given $G_k$ under the RMPP model is already a submodular set function maximization problem with partition matroids constraint. That is, we are only allowed to pick a single edge from edges with the same destination/end node. Thus, the greedy approach (i.e., Algorithm 2) naturally leads to a 1/2 approximation ratio [35].

**Time Complexity**. There are at most $O(|V|)$ iterations and each iteration takes $O(|V|(|V| + |E|) \log |V|)$ to find the critical edge with the maximum marginal gain via a variant of Dijkstra algorithm [23]. Thus, Algorithm 2 takes $O(|V|^2(|V| + |E|) \log |V|)$ time.

*4.1.2 Practical Edge Insertion.* Algorithm 2 suffers from very high time complexity and thus is infeasible in real-world large-scale social networks. In each iteration, Algorithm 2 tries to insert a critical incoming edge to a node with the maximum marginal gain w.r.t. the influence, and correspondingly update the candidate network which can result in updates of critical edges of the remaining nodes and the marginal gain of each critical edge (Lines 11-18). We observe that the nodes considered across all iterations actually form an order. If the order is known in advance, we can directly follow this order to insert critical edges *without* marginal gain computation and comparison, and hence the whole process can be notably accelerated. Unfortunately, computing this order is the main efficiency bottleneck of Algorithm 2.

We find that in the special case below (i.e., Theorem 3), there is no need to know this order because, no matter what the order is, the critical edge of any remaining node will not be updated in the iterative insertion process.

Theorem 3. *Given the original network $G$, a $k$-subnetwork $G_k$, the initial candidate network $G_c = \{V_c, E_c\}$ (i.e., Line 7 in Algorithm 2),*

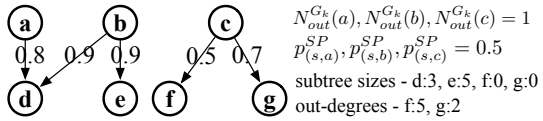



**Figure 4: An example of a candidate network $G_c$ with $k = 2$.**

*the seed node $s$, and $Cr(x) = \{v | \forall v \in V_c, x = u^*_v\}$ which denotes the set of nodes whose critical neighbor is $x$. If $\forall x \in V_c, |Cr(x)| \leq min(k, N^G_{out}(x)) - N^{G_k}_{out}(x)$, then directly inserting all critical edges leads to the optimal solution with the maximum influence.*

Proof. Since $min(k, N^G_{out}(x))$ describes the maximum number of outgoing edges from $x$ allowed in $G_k$, $N^{G_k}_{out}(x)$ describes how many of them have already been inserted into $G_k$, and $\forall x \in V_c, |Cr(x)| \leq min(k, N^G_{out}(x)) - N^{G_k}_{out}(x)$, inserting all critical edges in $Cr(x)$ will not result in the critical edge/neighbor update of the rest nodes (i.e., Line 17 in Algorithm 2). Based on Theorem 2, directly inserting critical edges of all nodes leads to the maximum influence spread since replacing any critical edges with the remaining edge candidates or inserting more edges will not increase influence. □

In the special case above, directly inserting all critical edges produces the optimal solution as Algorithm 2 which does not need to update the critical edges (Line 17) in each iteration. Although this special case is rather rare in practice, it inspires us to find an order which can be easily obtained in prior and follow this order to insert an incoming edge $(u, v)$ for each node $v$ whose $p^{SP}_{(s,u)} \cdot p_{(u,v)}$ is as large as possible. Ideally, this strategy is expected to effectively approximate Algorithm 2. We make the following observation to guide us to find such an order.

**Observation 1**. If the insertion of an edge $(u, v)$ introduces an $RMPP(s, v)$ with an influence probability greater than that of the $SMPP(s, v)$, for any node $x$ which $s$ needs to reach via $v$, the insertion of $(u, v)$ could also introduce $RMPP(s, x)$ with a greater influence probability than $SMPP(s, x)$.

Based on this observation, we should prioritize inserting incoming edges for nodes which *appear* in a large number of SMPPs from $s$ to different nodes since such insertions could potentially increase the influence probabilities of the *RMPPs* from $s$ to a large number of nodes. Based on Definition 7, the graph formed by all SMPPs is a tree rooted at $s$, namely *SMPP-based tree*, because any two nodes are connected by exactly one path. Thus, we can compute the appearance frequency by estimating the subtree size rooted at each node in the SMPP-based tree, which can be realized by a depth-first search and dynamic programming in $O(|E_k|)$ time. To help the influence spread reach a large number of users, we should also consider nodes in $G_c$ which have not been included in the $k$-subnetwork. For these nodes, their subtree sizes are set as 0 and ranked based on their out-degrees in the input network $G$ since nodes with greater out-degrees have larger potentials to expand the $k$-subnetwork. Algorithm 3 describes the overall process of practical edge insertion.

Example 4. *Figure 4 shows a candidate network $G_c$ with $k = 2$. The information needed by Algorithm 3 is listed at the right side. Nodes are ordered by their subtree sizes and then out-degrees. Thus, the node ranking we follow to insert incoming edges is $e, d, f$ and $g$.*

**Algorithm 4:** PracticalSubnetworkAugmentation (PSNA)

**Input** : The input network $G$, the seed user $s$, an integer $k$ and an error ratio $\epsilon$.
**Output**: The $k$-subnetwork $G_k$.

```
1  G_k = (V_k, E_k) ← k-subnetwork containing the seed s only;
2  G_c = (V_c, E_c) ← all outgoing edges of s in G;
3  L' ← all outgoing neighbors of s in G;
4  δ^△_pre = 1;
5  if G_c has edges then // expansion stage
6      G_k = PEI(G, G_k, G_c, s, k, L')
7      if (δ^△_{G_k}(s) − δ^△_pre)/δ^△_pre ≤ ε then Break ;
8      else δ^△_pre = δ^△_{G_k}(s);
9      G_c, L' = UpdateCandidateGraph(G, G_k, G_c, s, L');
10 G_k = FUR(G, G_k, G_c, s)// filling stage
11 Return G_k;
```

**Algorithm 5:** UpdateCandidateGraph

**Input** : The input network $G$, the $k$-subnetwork $G_k$, the candidate graph $G_c$ and the seed user $s$, a set $L'$ of candidate nodes to be included into $G_k$.
**Output**: The set of candidate nodes for $G_k$ expansion.

```
1  L ← ∅;
2  L' ← Remove all nodes not in G_k from L' ;
3  foreach u ∈ L' do
4      foreach v ∈ N^G_out(u) do
5          if (u, v) ∉ E_k and v ≠ s then
6              if v ∉ V_k then Add v into L;
7              Add (u, v) into G_c;
8  Return L ;
```

*When we consider node $e$, edge $(b, e)$ will be inserted into $G_k$ and edge $(b, d)$ will be deleted from $G_c$ since $N^{G_k}_{out}(b)$ will become $2 = k$. When we consider $d$, edge $(a, d)$ will be inserted. Similarly, edge $(c, f)$ will be inserted but edge $(c, g)$ will be deleted when we consider node $f$ and thus there will not be incoming edge candidates for $g$.*

**Time Complexity**. In Algorithm 3, Line 1 constructs the candidate graph in $O(|V| + |E|)$ time, Lines 2-4 rank nodes based on their subtree sizes in the SMPP-based tree or degrees in $O((|V| + |E|) \log |V|)$ time, and Lines 5-10 iteratively insert incoming edges for ranked nodes in $O(|V| + |E|)$ time since the probabilities of all SMPPs from $s$ have already been computed when we construct the SMPP-based tree in Line 2. Thus, the total time complexity is $O((|V| + |E|) \log |V|)$.

## 4.2 Subnetwork Augmentation

The subnetwork augmentation sketch (i.e., Algorithm 1) suffers from two issues which incur notable computation costs.

- Issue 1: the candidate graph in each iteration is constructed from scratch without leveraging the candidate graphs generated in previous iterations.
- Issue 2: the number of edge insertions in each iteration is very limited (i.e., at most one incoming edge for each node) such that it may take considerable iterations to fill up the recommendations.

**Observation 2**. Regarding Issue 1, we observe that the remaining candidate graph $G^{re}_c$, at the *end* of PEI (i.e., Algorithm 3) in the *last* iteration of Subnetwork Augementation Sketch (i.e., Algorithm 1), is a subgraph of the *initial* candidate graph $G^{init}_c$ of PEI in the

**Algorithm 6:** FillUpRecommendation (FUR)

**Input :** The input network $G$, the $k$-subnetwork $G_k$, the candidate graph $G_c$ and the seed user $s$.
**Output :** The updated $G_k$.

1 **foreach** $v \in V_k$ *where* $|N_{in}^{G_k}(v)| > 0$ **do**
2  $L^{in}[v] \leftarrow$ a list of incoming neighbors of $v$ in $G_c$ where each neighbor $u$ is sorted by $p_{(s,u)}^{SP} \cdot p_{(u,v)}$ in descending order;
3  $I[v] = 1$;
4 $L \leftarrow$ a list of nodes in $V_k$ ordered by their subtree sizes in the graph formed by SMPPs from $s$ in descending order;
5 **while** $L$ *is not empty* **do**
6  $L' \leftarrow$ an empty list;
7  **foreach** $v$ *in* $L$ **do**
8  **foreach** $i$ *from* $I[v]$ *to* $|L^{in}[v]|$ **do**
9  $u = L^{in}[v][i]$;
10  **if** $(u, v) \notin E_c$ **then** Continue;
11  **if** $i \neq |L^{in}[v]|$ **then** Add $v$ into $L'$ ;
12  $I[v] = i + 1$;
13  Add $(u, v)$ into $G_k$; Remove $(u, v)$ from $G_c$;
14  **if** $|N_{out}^{G_k}(u)| = min(k, |N_{out}^{G}(u)|)$ **then**
15  Delete all outgoing edges of $u$ from $G_c$.
16  Break;
17  $L = L'$;
18 Return $G_k$;

*current* iteration. Specifically, the extra edges of $G_c^{init}$ compared to $G_c^{re}$ are outgoing edges from the nodes that are newly introduced into the $k$-subnetwork $G^k$ by PEI in the last iteration. Thus, we propose a method called UpdateCandidateGraph (Algorithm 5) to build the candidate graph based on the previous one by only introducing these edges. With this update method, the process of constructing the candidate graph from scratch in PEI (i.e., Line 1 in Algorithm 3) can be replaced.

**Observation 3**. Regarding Issue 2, we observe that the influence spread of $s$ will converge after a few iterations (e.g., 6) of the subnetwork augmentation and the $k$-subnetwork $G_k$ barely includes more nodes from $G$ after convergence (as shown in Figure 11 in experiments). Considering that we treat all inserted edges as native ones at the end of each iteration, the influence spread of $s$ is actually based on SMPP paths. Therefore, the converged influence also indicates that the SMPPs from $s$ to the rest of nodes barely change, and accordingly the ranking of the nodes based on their subtree sizes in the SMPP-based tree is quite stable.

As a result, after the convergence, we can leverage the same and converged node ranking for edge insertions in the rest of iterations of the subnetwork augmentation. Furthermore, the candidate graph at the end of the last iteration naturally becomes the initial candidate graph in the current iteration since no more new nodes are considered. With these properties, we propose a method called FillUpRecommendation (FUR) (Algorithm 6) which simulates the process of PEI in the rest of all iterations of the subnetwork augmentation. Specifically, each while loop from Line 6 to 17 corresponds to one iteration where we greedily insert the critical edge for each node in $L$ constructed based on the converged ranking. In each loop, $L'$ is used to store nodes which still have incoming neighbors and thus will be considered in the next iteration, and $I[v]$ is used

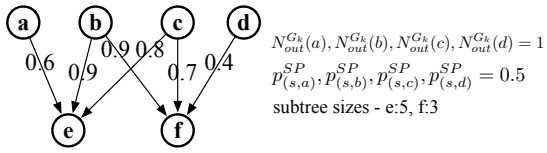



**Figure 5: An example of a candidate network $G_c$ with $k = 2$.**

to efficiently retrieve the critical neighbor by pruning out neighbors which either were considered in previous iterations or cannot be connected to $v$ anymore due to the update of $G_c$ in previous iterations (Lines 14-15). Since each incoming neighbor has been sorted in Line 2, simulating each iteration is very efficient with aforementioned data structures.

Example 5. *Figure 5 shows a candidate network $G_c$ with $k = 2$. Suppose that the influence spread in $G_k$ has converged at iteration $i$ and the converged node ranking is $e$ and $f$. Algorithm 3 will be executed two more iterations to reach the termination condition (i.e., no edges exist in $G_c$). At iteration $i$+1, edges $(b, e)$, $(b, f)$, $(c, f)$ and $(c, e)$ will be inserted, deleted, inserted and deleted in sequence respectively. The remaining candidate graph becomes the initial candidate graph at iteration $i + 2$ where edges $(a, e)$ and $(d, f)$ will be inserted. The method FUR (Algorithm 6) is proposed to simulate the iterative process after the convergence and the while loop in FUR will be executed twice.*

**Practical Subnetwork Augmentation**. By incorporating the measures for tackling the two issues above, we propose a framework called PracticalSubnetworkAugmentation (PSNA) (Algorithm 4) where the predefined $\epsilon$ (e.g., $10^{-4}$) controls the convergence point (Line 7) and FUR, based on the converged subtree-size-based node ranking (Line 10), repeats the process of Lines 5-6 in PSNA until no edges exist in $G_c$. Note that $L'$ records the new candidate nodes to be incorporated into $G_k$ for graph expansion in the last iteration and is updated by UpdateCandidateGraph in Line 9. For ease of illustration in our experiments later, we regard the process before convergence as the *expansion stage* (i.e., Lines 5-9) and the process afterwards as the *filling stage* (i.e., Line 10).

**Time Complexity**. Suppose Algorithm 4 takes $I$ iterations to converge. Lines 5-9 take $O(I(|V|+|E|)\log|V|)$ time which is acceptable in practice since $I$ is empirically small (e.g., 6). Lines 1-3 in Algorithm 6 take $O(|V|d_m \log d_m)$ where $d_m$ refers to the maximum in-degree in $G$, and Lines 5-17 take $O(|E|)$ time as each incoming neighbor of each node is visited only once. Thus, the total time complexity of Algorithm 4 is $O(I(|V| + |E|)\log|V| + |V|d_m \log d_m)$.

## 5 SOLVING IMCSN FOR MULTIPLE SEEDS

When there is a set $S$ of independent seed nodes, directly adopting the previous idea of inserting at most one incoming edge for each node is not feasible, since the SMPPs from seed nodes to the same node can be different and hence the critical edge for each node is seed-specific. The most straightfoward approach to handle the case is to iteratively insert the edge, which brings the maximum marginal gain to the sum of the influence increment of all seed nodes, until no edges can be inserted into the $k$-subnetwork. However, this approach is impractical due to expensive marginal gain computation and a huge number of iterations needed to converge.

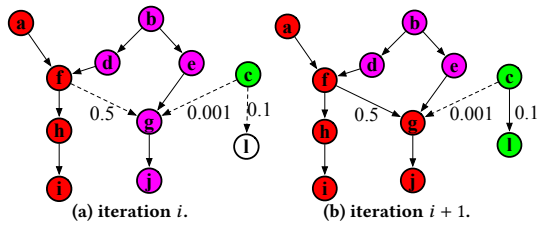



**Figure 6: SMPP-based subgraphs in two iterations.**

**Observation 4**. We observe that an edge $(u, v)$ insertion can have different levels of impact on increasing the influence spread of different seed nodes. For example, if $u$ is a neighbor of the seed node $s_1$ but ten hops away from the seed node $s_2$, inserting $(u, v)$ is more likely to bring more influence to $s_1$ than $s_2$. Furthermore, as the iteration goes, the $k$-subnetwork will be expanded with more topological information during the augmentation process. Thus, we should not 'waste' many unnecessary recommendation opportunities in the current stage of the $k$-subnetwork since they can be saved for better decision making in later iterations.

Thus, to make the best use of candidate recommendations, we should focus only on the recommendation which will bring large influence increment to the relevant seed node. Specifically, we define $r(u) = \arg\max_{s \in S} p^{SP}_{(s,u)}$ as the *relevant seed node* of $u$ since inserting outgoing edges from $u$ would be more likely to increase the influence of the seed node $r(u)$. For all incoming neighbors of node $v$, we should connect the critical neighbor $u^*_v$ to $v$ which forms the RMPP with the maximum probability from the corresponding seed node $r(u^*_v)$, i.e., $u^*_v = \arg\max_{u \in N^{G_c}_{in}(v)} p^{SP}_{(r(u),u)} \cdot p_{(u,v)}$.

Nodes with the same relevant seed node $s$ and SMPPs from $s$ to them naturally form an SMPP-based tree. We can construct the SMPP-based trees based on each seed node and these trees are disjoint (i.e., no node overlap). With these trees, we can compute the subtree size of each node in the corresponding tree and prioritize inserting the nodes with large subtree sizes, because increasing the probability to influence the node $u$ will also increase the probability to influence the descendants of $u$ in the SMPP-based trees.

Example 6. *Figure 6 shows an example of the SMPP-based trees of a 2-subnetwork in two consecutive iterations of the PSNA. Here, a, b and c are the seed nodes, all solid edges have a weight of 0.1, nodes with the same color belong to the same SMPP-based trees, and white nodes are candidates for the 2-subnetwork expansion. In iteration i, node g is processed before l because g has a larger subtree size (i.e., 2). We have two candidate incoming edges to node g, and $(f, g)$ will be inserted since $f = \arg\max_{u \in \{f,c\}} p^{SP}_{(r(u),u)} \cdot p_{(u,g)}$ where $r(f) = a$ and $r(c) = c$. In the input 2-subnetwork of iteration $i+1$, the relevant seed nodes of g and j are updated to node a since inserting outgoing edges from them is more likely to increase the influence of a.*

The premise of the aforementioned idea requires computing the SMPP between each node $u$ and $r(u)$. A straightforward approach is to enumerate SMPPs from every seed node to $u$ and get the one with the greatest probability. However, this is very expensive in real-world large-scale social networks, especially when $|S|$ is large. To mitigate this issue, we can simply introduce edges with the same influence probability of 1 from a *virtual* node $x$ to every seed node. Afterwards, we only need to adopt a variant of Dijkstra algorithm [23] to compute the SMPPs from $x$ to the rest of the nodes; the second node in each SMPP must be a seed node, and be the relevant seed node of the end node of this path. With the virtual node $x$, we simplify the case of multiple seed nodes into the case of a single seed node $x$ since edge insertion can be guided by the probability of SMPP from $x$ instead of relevant seed nodes, i.e.,

$$u^*_v = \arg\max_{u \in N^{G_c}_{in}(v)} 1 \cdot p^{SP}_{(r(u),u)} \cdot p_{(u,v)} = \arg\max_{u \in N^{G_c}_{in}(v)} p^{SP}_{(x,u)} \cdot p_{(u,v)}$$

Since we transform the IMCSN problem for multiple seed nodes into the one for a single virtual node $x$, we can directly adopt PSNA for the single seed node with minor adjustments: (1) introduce $O(|S|)$ edges with the same influence probability 1 from a virtual node $x$ to every seed node in the initial $k$-subnetwork; (2) remove the virtual node and edges between it and seed nodes from the output $k$-subnetwork of PSNA.

**Time Complexity**. Considering that we only change the topology of the $k$-subnetwork, the previous time complexity analysis for the single case still applies. Thus, the total time complexity is $O(I(|V| + |E| + |S|) \log |V| + |V| d_m \log d_m)$.

## 6 EXPERIMENT

In this section, we will conduct experiments on three problems to demonstrate the robustness and effectiveness of our methods:

- The first problem is maximizing the influence in *open* social networks via edge insertions, where existing baselines including the state-of-the-art [23] are compared (Section 6.1).
- The second is our proposed problem, IMCSN, where the extensions of the methods in the first problem are compared (Section 6.2). Note that the core difference between these two problems has been illustrated and please refer to Section 2 for details.
- The third problem is maximizing users' Click-trough Rate in an activity of an online Tencent application, where we deploy our method and evaluate how it helps improve user retentions and interactions (Section 6.3).

**Datasets**. Table 1 presents all the real-world undirected social networks used. In particular, MOBA and MOBAX correspond to two friendship networks of Tencent multiplayer online battle arena games, RPG corresponds to a friendship network of a role-playing game, and the other datasets are available in [30]. Note that each edge is represented twice since the influence propagation is directed. To obtain experimental results in a reasonable time under different problem settings, the first four datasets, the next four datasets and the last dataset will be used for the first, the second and the third problem, respectively.

**Environments**. We conduct all experiments on a Linux server with Intel Xeon E5 (2.60 GHz) CPUs and 512 GB RAM. All algorithms are implemented in Python and our code is available at [12].

### 6.1 Experiment with the Open Sharing Model

In this section, we study maximizing the influence in open social networks. Here, we insert a set $X'$ of edges from the candidate set $X$ into the original social network such that (1) at most $k$ edges in $X'$ share the same source node and (2) the influence sum of the seed nodes is maximized. The purpose of this section is twofold:

**Table 1: Dataset Statistics**

| Dataset | \|V\| | \|E\| | Avg. Degree |
|---|---|---|---|
| EU-Core | 1,005 | 51,142 | 51 |
| Hamster | 2,426 | 33,261 | 14 |
| CiaoDVD | 4,658 | 80,266 | 17 |
| Brightkite | 58,228 | 428,156 | 7 |
| Catster | 149,700 | 10,898,550 | 73 |
| MOBA | 503,029 | 9,372,022 | 19 |
| RPG | 2,331,047 | 88,227,562 | 38 |
| Orkut | 3,072,441 | 234,369,798 | 76 |
| MOBAX | 36,201,207 | 3,281,207,036 | 90 |

- Our proposed method PSNA is able to effectively maximize the influence of multiple seed nodes, as compared to the state-of-the-art called continuous greedy CG [23] on this problem.
- Our proposed method PSNA is robust to different variations of influence maximization via edge recommendations.

**Baselines**. Baselines are listed below where Degree and FoF (i.e., Friend of Friend) were also compared by CG [25]. For ease of description, we assume out-degrees of nodes in the original network to be greater than $k$.

- Degree based method where, for candidates sharing the same source node, top $k$ edges with the highest degrees of end nodes are selected.
- FoF based method where, for candidates sharing the same source node, top $k$ edges whose source and end nodes share the highest number of common friends are selected.
- Random based method where $k$ edges are randomly selected from candidates sharing the same source node. Its performance is reported as the average over five independent runs.
- CG [23] where edges are inserted based on a continuous greedy method followed by randomized rounding. We set all parameters as recommended by the original work.

**Edge Weight Settings**. We follow the previous work [23] to adopt the constant model where each edge in the original network and candidate edge for insertion is assigned with a constant influence probability and we set it as 0.1. Note that the constant model and the influence probability setting are also commonly used by existing studies in the classical influence maximization problem [14, 36, 46].

**Parameter Settings**. Considering that CG is not scalable to a large number of seed nodes and edge candidates, we conduct experiments on small-scale datasets. Specifically, we randomly select five nodes from the top 1% nodes with the highest degrees in the original network as the seed nodes. Afterwards, we generate 50 outgoing edges from each node within two hops away from the seed nodes as candidates, and these candidates are generated randomly by following the setup of [23] where an edge candidate can be added between any pair of nodes without group information. We set $k$ as 20, the error ratio $\epsilon$ of PSNA as $10^{-4}$, and follow the common standard [14] to estimate the influence of any solution under the IC model via 10,000 Monte Carlo simulations.

**Experimental Results**. Table 2 compares the influence spread achieved by different methods under the RMPP model and the IC model, respectively. It is obvious that there is a positive correlation between influence under the RMPP model and the one under the IC model. That is, greater influence under the former one also indicates larger influence under the latter. Hence, the idea of optimizing the RMPP-based influence spread to approximate the IC-based influence is effective. PSNA significantly outperforms Degree, FoF and Random, and achieves very competitive results with CG. Furthermore, PSNA achieves up to five-orders-of-magnitude speedup over CG as shown in Table 3. These results demonstrate the efficiency, effectiveness and robustness of PSNA, in terms of maximizing the influence of multiple seed nodes via edge recommendations/insertions.

**Table 2: Comparison of influence achieved by different methods under the RMPP model (RP) and the IC model.**

| Method | EU-Core | | Hamster | | CiaoDVD | | Brightkite | |
|---|---|---|---|---|---|---|---|---|
| | RP | IC | RP | IC | RP | IC | RP | IC |
| Degree | 65 | 2801 | 65 | 2513 | 92 | 5891 | 80 | 8912 |
| FoF | 68 | 3012 | 67 | 2789 | 95 | 6013 | 75 | 8134 |
| Random | 70 | 3189 | 64 | 2445 | 100 | 6231 | 73 | 7986 |
| PSNA | 76 | 3684 | 72 | 3142 | 109 | 6618 | 88 | 12034 |
| CG | 78 | 3769 | 74 | 3221 | 109 | 6680 | 89 | 12851 |

**Table 3: Running time comparison between CG and PSNA.**

| Method | EU-Core | Hamster | CiaoDVD | Brightkite |
|---|---|---|---|---|
| PSNA | 5.8E-1 | 1.7E-1 | 1.5E0 | 1.7E1 |
| CG | 3.56E4 | 3.0E4 | 5.5E4 | 4.6E4 |

**Table 4: Effectiveness comparison with $k = 30$ and $|S| = 50$.**

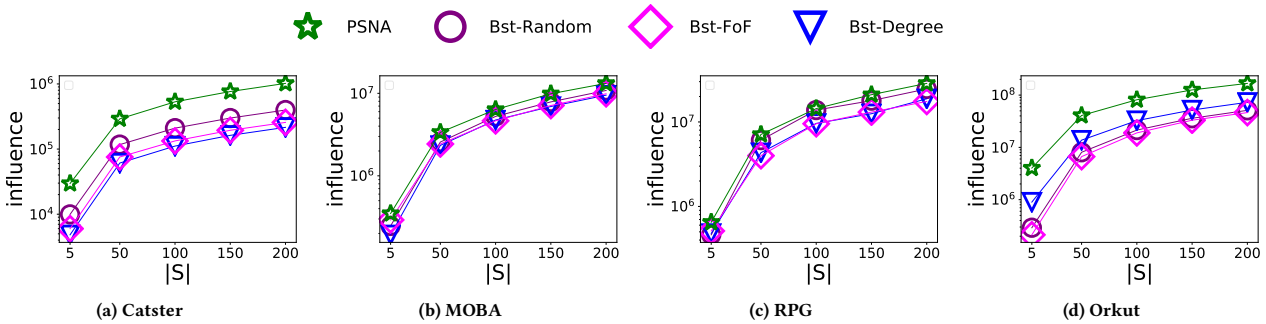
| Dataset | Degree | | FoF | | Random | | PSNA |
|---|---|---|---|---|---|---|---|
| | Ori | Bst | Ori | Bst | Ori | Bst | |
| Catster | 3.1E2 | 4.2E5 | 3.8E2 | 4.8E5 | 3.6E3 | 6.3E5 | **8.6E5** |
| MOBA | 5.0E6 | 7.4E6 | 6.5E6 | 7.7E6 | 7.0E6 | 8.0E6 | **9.8E6** |
| RPG | 1.1E6 | 2.1E7 | 2.4E6 | 2.2E7 | 1.9E7 | 2.4E7 | **3.2E7** |
| Orkut | 7.0E2 | 5.4E7 | 6.8E2 | 5.6E7 | 9.6E5 | 5.9E7 | **8.2E7** |

**Table 5: Running time (s) comparison with $k = 30$ and $|S| = 50$ where the numbers in brackets under PSNA refers to the time cost in the graph expansion stage which is also involved in boosted baselines. Note that the number of common friends is pre-computed in Bst-FoF.**

| Dataset | Bst-Degree | Bst-FoF | Bst-Random | PSNA |
|---|---|---|---|---|
| Catster | 1.1E2 | 1.0E2 | 9.8E1 | 1.2E2 (9.1E1) |
| MOBA | 1.4E2 | 1.3E2 | 1.2E2 | 1.8E2 (1.3E2) |
| RPG | 9.6E2 | 9.1E2 | 8.8E2 | 9.8E2 (7.3E2) |
| Orkut | 4.4E3 | 4.2E3 | 4.1E3 | 5.1E3 (3.6E3) |

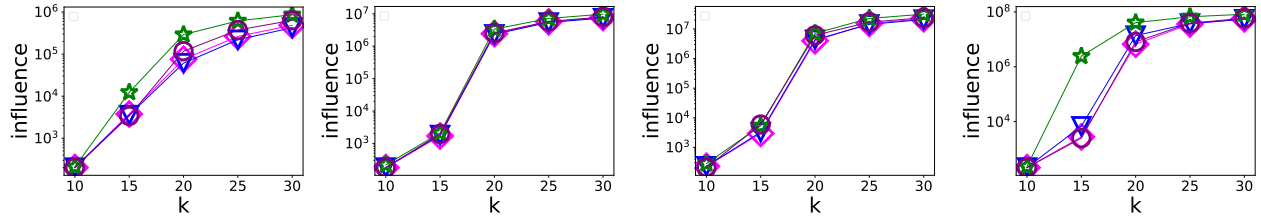
## 6.2 Experiment on the IMCSN problem

We conduct seven experiments to demonstrate that: (1) the extension of existing baselines on open social networks cannot well address our problem but their performance can be significantly boosted with the help of our method PSNA (Exp1); (2) PSNA consistently and significantly outperforms all the boosted baselines




(a) Catster (b) MOBA (c) RPG (d) Orkut

**Figure 7: Performance Comparison with different $|S|$.**




(a) Catster (b) MOBA (c) RPG (d) Orkut

**Figure 8: Performance Comparison with different $k$.**

with different $|S|$ and $k$ (Exp2-3); (3) how $k$ and $|S|$ impact the convergence of PSNA (Exp4); (4) PSNA is highly scalable to handle large-scale datasets (Exp5); (5) the influence convergence is also a good indicator of the node size convergence in the diffusion network (Exp6); (6) the influence of seeds in the diffusion network, produced by PSNA with a small edge size, is very competitive with the influnce of the seeds in the original network (Exp7).

**Baselines**. We compare six baselines which include the aforementioned methods, Degree, FoF and Random, as well as their respective boosted versions. Since each baseline has two versions, original and boosted, we use the prefixes 'Ori' and 'Bst' to distinguish them. The boosted versions are built upon the diffusion network generated by the *expansion* stage of PSNA (i.e., Lines 5-9 in Algorithm 4) and work similar to the original ones. The only difference is that, in the boosted version, some important edges have been inserted by PSNA and these baselines only 'fill up' recommendations for those nodes in the generated diffusion network. Hence, if we assume that the out-degrees of nodes in the original network are greater than $k$ and our method has already inserted $h < k$ outgoing edges from $u$, these boosted versions just need to select $k - h$ outgoing edges for $u$. For the reason why CG [23] cannot be extended, please refer to Section 2 and its scalability issue in Section 6.1 for details.

**Edge Weight Settings**. Considering that the influence probability of edges are different in practice, we adopt the commonly used trivalency model [24] for Catster and Orkut without edge attributes. It randomly assigns a weight for each edge, from $\{10^{-1}, 10^{-2}, 10^{-3}\}$.

For MOBA and RPG, we assign edge weights based on the intimacy of friendship. In MOBA and RPG, each pair of friends has different levels of intimacy which describes the number of interactions (e.g., the number of games they play together, the number of gifts sent from one to another). Since the pair-wise intimacy in both MOBA and RPG is represented as integers, here we transform it into an influence probability within [0,1] for deployment in the IC model. According to the Susceptible-Infected-Recovered (SIR) model [13] and heterogeneous mean-filed theory [21, 29, 54, 62], the lowest influence probability should not be smaller than a constant $\lambda$ times $\beta_c = \sum_{v \in V} |N_{out}(v)| / (\sum_{v \in V} |N_{out}(v)|^2 - \sum_{v \in V} |N_{out}(v)|)$, where $\beta_c$ is the epidemic threshold in the SIR model and calculated as 0.024 and 0.001 on MOBA and RPG, respectively. If $\lambda\beta_c$ is too small, the influence of the seeds will be quite limited. If $\lambda\beta_c$ is too large, the influence spread can cover a large percentage of nodes, irrespective of where it originated, and the methods' performance cannot be well compared. By following existing studies [51, 56], we determine $\lambda$ by simulation on real networks. Specifically, we determine $\lambda_1$ and $\lambda_2$ and control the edge weights within the range $[\lambda_1\beta_c, \lambda_2\beta_c]$. Each influence probability $p_{(u,v)} = ((u,v)_I - \min_{e \in E} e_I)/(\max_{e \in E} e_I - \min_{e \in E} e_I)(\lambda_2 - \lambda_1)\beta_c + \lambda_1\beta_c$ where $(u,v)_I$ denotes the intimacy between $u$ and $v$. Note that we have checked several settings of $\lambda_1$ and $\lambda_2$ and these different settings will not affect the conclusion (i.e., performance ranking of methods). To test these methods' robustness to the influence probability distributions, in both MOBA and RPG, we set the range $[\lambda_1\beta_c, \lambda_2\beta_c]$ as $[0.007, 0.01]$ which has a dramatically different distribution from the trivalency model deployed for other datasets.

**Parameter Settings**. We randomly select $|S|$ (50 by default) nodes from the top 1% nodes with the highest degrees in the original network as the seed nodes, set $k = 20$ and the error ratio $\epsilon = 10^{-4}$ by default. We evaluate solution quality based on the IC-based influence spread via 10,000 Monte Carlo simulations.

***Exp1 - Case study on the two versions of baselines.*** Table 4 and Table 5 compare the effectiveness and efficiency with $k = 30$ and $|S| = 50$. We have four main observations:

(1) Org-Random is more effective than the original versions of other baselines. The reason is that nodes with high degrees will

'attract' much more incoming edges in other strategies (i.e., Org-Degree and Org-FoF) which 'waste' a lot of recommendation opportunities to repeatedly influence/activate these nodes.

(2) The boosted versions can achieve about five-orders-of-magnitude larger influence than their original counterpart but can still be notably outperformed by PSNA. Specifically, PSNA can still outperform the second best performer by at least 23%-39% on different datasets respectively, which demonstrates the effectiveness of both two stages (i.e., the expansion and filling stages).

(3) We also compute the constitution of the edges recommended by PSNA in the boosted baselines, and the result shows that PSNA only contributes 17%-27% of total edges in the diffusion network. The significant performance improvement of baselines with limited involvement of PSNA further demonstrates the effectiveness of PSNA in terms of identifying important edges and nodes for increasing the seeds' influence.

(4) PSNA is very competitive with other boosted baselines in terms of running time, because all methods involve the *expansion* stage of PSNA (i.e., Lines 5-9 in Algorithm 4) which dominates the total computational cost.

Due to the poor performance of the original baselines, we only use the boosted versions for comparison in the rest experiments.

***Exp2 - Effectiveness comparison with different*** $|S|$. Figure 7 compares the performance with different $|S|$. Our method PSNA consistently outperforms other methods across all instances while the performance of other methods is not stable, e.g., Bst-Random outperforms Bst-Degree on Catster but the latter is more effective on Orkut. Another interesting observation is that the performance ranking of methods on the same dataset is consistent across different $|S|$. We think that a good edge recommendation for a non-seed user $u$ under an instance with a small $|S|$ can also be an effective recommendation for $u$ as the seed user under an instance with a great $|S|$. Thus, the diffusion networks generated by the expansion stage of PSNA under different $S$ can be similar such that edges chosen by a specific baseline upon these networks have large overlaps, which explains this observation and reflects the superiority of PSNA.

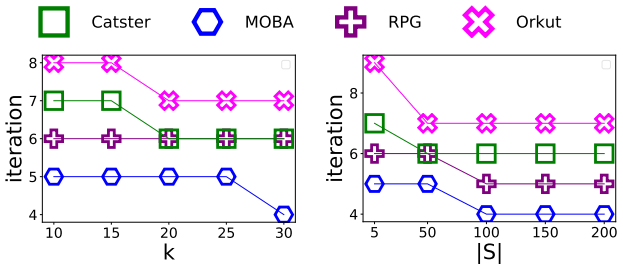
***Exp3 - Effectiveness comparison with different*** $k$. Figure 8 compares the performance at different $k$ across all datasets. We have four main observations:

(1) PSNA outperforms the boosted baselines and can achieve up to two-orders-of-magnitude larger influence (e.g., on Orkut).

(2) The performance gap becomes smaller when $k$ is larger. The reason is that all methods make almost the same recommendation strategy to low-degree nodes since their out-degrees are close to or even smaller than $k$. In this case, the number of combinations of $k$ outgoing edges of these nodes is very limited. This observation also explains why the performance gap is larger on datasets with greater average degrees.

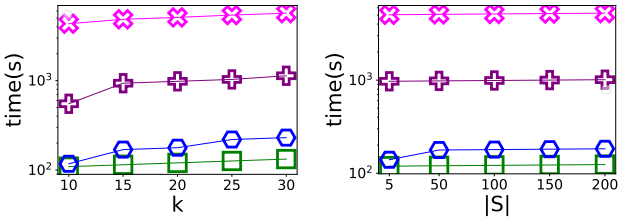
(3) When $k$ is small (e.g., 10), all methods achieve similar influence since it is barely possible for any method to achieve large influence with very limited edges.

(4) The performance gap on MOBA and RPG is not as significant as that on other datasets. That is because the average edge weight on MOBA and RPG are notably larger. Thus, regardless of how the diffusion network is generated, seeds can easily influence many nodes with sufficient influence probabilities.




(a) varing $k$ (b) varing $|S|$

**Figure 9: The number of iterations in the expansion stage with different $k$ and $|S|$.**




(a) varing $k$ (b) varing $|S|$

**Figure 10: Running time of PSNA with different $k$ and $|S|$.**

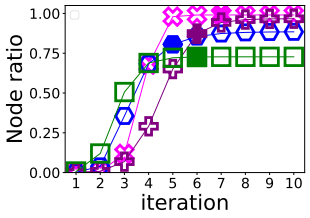
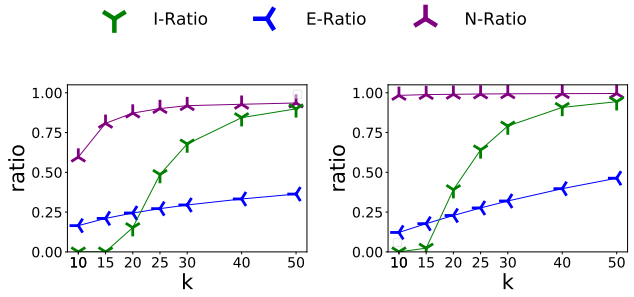



**Figure 11: The percentage of the nodes in the diffusion network over the one in the original network with different iterations in the expansion stage of PSNA, where the iterations with solid fill refer to the convergence points under different datasets repsectively.**

Note that the performance difference may not be easily distinguished visually due to the log scale of y axis, but we have shown the notable performance difference with $k = 20$ and $k = 30$ in Figure 7 and Table 4 at a finer granularity already.

***Exp4 - Impact of*** $k$ ***and*** $|S|$ ***on PSNA convergence***. Figures 9 (a) and (b) show the number of iterations in the expansion stage of PSNA when varing $k$ and $|S|$ respectively. Interestingly, when $k$ or $|S|$ increases, the number of iterations for convergence is non-increasing. This is because PSNA will include more nodes into the current $k$-subnetwork in each iteration such that it may take fewer iterations to identify all important nodes for influence spread.

***Exp5 - Scalability study on PSNA***. Figures 10 (a) and (b) show the runtime of our (full-stage) PSNA when varing $k$ and $|S|$. As $k$ grows, more nodes tend to be included in the diffusion network and more edges tend to be 'filled up' in the filling stage, hence costing more time. On the other hand, increasing $|S|$ does not notably incur




(a) RPG (b) Orkut

**Figure 12: The influence ratio (I-Ratio), edge ratio (E-Ratio) and node ratio (N-Ratio) in the diffusion network produced by PSNA with different $k$.**

more computation costs, for two reasons: 1) we transform the case of multiple seeds into the one of a single virtual node by introducing only $|S|$ extra edges into the diffusion network; 2) the size of the diffusion networks generated under instances of different $|S|$ can be similar as discussed earlier. It is obvious to see that PSNA is highly scalable and efficient, since it can produce good recommendation on datasets with millions of nodes and hundred millions of edges within two hours (e.g., Orkut). This performance is promising since such recommendation is usually deployed for a long-term usage.

***Exp6 - Impact of iterations on the diffusion network node size.*** Figure 11 shows how the node size of the diffusion network in the expansion stage of PSNA grows iteratively. Here, the iterations with solid fill refer to the convergence points. Note that the expansion stage is terminated after convergence for other experiments and here we force it to run 10 iterations. It is obvious that, the node size in the diffusion network converges as the influence spread, which indicates the effectiveness of the convergence criteria.

***Exp7 - Case study with large*** $k$. Here, we conduct a case study on the two largest datasets, RPG and Orkut. Our goal is to show how the statistics of the diffusion network produced by PSNA and influence spread of seed nodes in this network will change when we further increase $k$ up to 50. Specifically, we use the *influence ratio* (I-Ratio) to denote the percentage of the influence of seeds in the diffusion network over their full influence in the original network, *edge ratio* (E-Ratio) and *node ratio* (N-Ratio) to denote the percentage of the edge and node size of the diffusion network over the ones of the original network, respectively. As shown in Figure 12, as $k$ increases, PSNA is able to achieve an I-Ratio of 90% with only an E-Ratio of 36%, and an I-Ratio of 94% with only an E-Ratio of 46% on RPG and Orkut respectively. These again demonstrate the superiority of PSNA. Furthermore, when the N-Ratio converges, the I-Ratio can still notably increase as $k$ grows, and even the converged N-Ratio can be notably smaller than 1 (e.g., on RPG). These indicate that PSNA can identify important nodes for spreading the influence.

## 6.3 Deployment

In order to demonstrate the practical effectiveness of our IMCSN problem, we deployed our solution into an activity in a multi-player online battle arena game of Tencent of which the friendship network is MOBAX in Table 1.

**Summary of the activity**. Users join this activity by interacting (e.g., sending gifts and gaming invitation) with their friends recommended by the system and will obtain rewards via such interactions. This form of interaction could trigger domino effects such that the interactions can stimulate the message receiver to further interact with other users. The aim of this activity is to improve user retention and interactions. Note that, in this activity, we do not specifically designate seeds and every user could be a potential and spontaneous seed. Specifically, users who log in the game will see this activity and may perform three possible behaviors: (1) proactively start interactions as a seed, (2) start interactions as a message receiver (i.e., a non-seed) and (3) ignore this activity.

**Edge weights and the diffusion model**. We adopt the same settings as we use for generating edge weights on MOBA and RPG, and generate the diffusion network based on the IC model. Note that the IC model is only used for generating the diffusion network but not used for evaluation since it is very likely that the real-world influence will not spread in a way specified by the IC model or any other classical diffusion models (e.g., Linear Threshold model [46]).

**Methods for comparison**. Four methods, namely Random, Degree, FoF and PSNA, are deployed. For evaluation, we use an online A/B testing. Specifically, it randomly assigns each online user to one of these four methods which recommend a list of friends for each assigned user. The list size $k$ is set as 20 for all methods because, in this application, only 20 friends are shown due to several reasons (e.g., the screen size and resolution settings of most mobile devices). Based on our observations made in Section 6.2, the k-subnetwork generated based on a given seed set could also be effective for different seed sets. Thus, for PSNA, we first generate the $k$-subnetwork with top 10% of assigned users with the highest degrees as the seeds, and then use this generated $k$-subnetwork for this activity where every user could be a potential seed or non-seed.

**Experimental results**. The number of users who interact with recommended friends directly indicates the quality of the underlying recommendation system and can effectively reflect the potential size of users who are impacted by others to performance interactions. Thus, we adopt the Click-through Rate (CTR) to evaluate the performance. The CTR denotes the number of users who interacted with recommended friends divided by the total number of users who log in the game during this activity. The CTR achieved by Random, Degree, FoF and PSNA are 81.87%, 82.73%, 82.81% and **88.14%** respectively, which again demonstrates the effectiveness of PSNA. Despite that CTR is a very popular evaluation metric for recommendation systems in industry, there still exist some other evaluation metrics (e.g., the actual influence spread of seeds via knowing the propagation paths and who are the seeds) worthy of explorations. However, due to the privacy restriction of this activity, we have no access to the detailed information in user logs to compute the performance under other potential metrics. In future, we would like to see how our proposed method will perform in other activities under various evaluation metrics.

## 7 CONCLUSION

In this paper, we study the problem of Influence Maximization in Closed Social Networks which aims to recommend a limited number of edges for users to propagate information, such that the seeds' influence via the selected edges is maximized. This problem is shown to be very useful in many industrial applications and we prove the NP-hardness of this problem. Moreover, we further propose a scalable and effective method to augment the diffusion network of seed users. We conduct extensive experiments to demonstrate that our method is very efficient and effective in our problem, a variant of our problem in open social networks and a real-world application. As a future direction, we will explore this problem under other diffusion models, and analyze theoretical properties and extension of our solutions.

## REFERENCES


[1] 2014. https://business.sohu.com/20140624/n401244299.shtml.
[2] 2018. https://www.postbeyond.com/blog/millennials-genz-social-media/.
[3] 2018. https://medium.com/@lorenabarquin/are-closed-social-media-platforms-the-future-of-social-3a5b0cbea025.
[4] 2018. https://www.warc.com/newsandopinion/news/the_new_facebooks_the_trend_towards_a_closed_social_media/40929.
[5] 2018. https://www.quora.com/Why-are-some-people-not-interested-in-exposing-themselves-on-social-media.
[6] 2021. https://www.tailwindapp.com/blog/private-on-pinterest.
[7] 2021. https://zhuanlan.zhihu.com/p/82896779.
[8] 2021. https://cfm.qq.com/gicp/news/186/15185249.html.
[9] 2022. https://www.facebook.com/help/233739099984085.
[10] 2022. https://help.twitter.com/en/safety-and-security/public-and-protected-tweets.
[11] 2022. https://techalignment.com/closed-versus-open-social-networks/.
[12] 2022. https://github.com/rmitbggroup/IMCSN.
[13] Roy M Anderson and Robert M May. 1992. *Infectious diseases of humans: dynamics and control*.
[14] Akhil Arora, Sainyam Galhotra, and Sayan Ranu. 2017. Debunking the myths of influence maximization: An in-depth benchmarking study. In *SIGMOD*. 651–666.
[15] Cigdem Aslay, Laks VS Lakshmanan, Wei Lu, and Xiaokui Xiao. 2018. Influence maximization in online social networks. In *WSDM*. 775–776.
[16] Suman Banerjee, Mamata Jenamani, and Dilip Kumar Pratihar. 2019. ComBIM: A community-based solution approach for the Budgeted Influence Maximization Problem. *Expert Systems with Applications* 125 (2019), 1–13.
[17] Glenn S Bevilacqua and Laks VS Lakshmanan. 2021. A fractional memory-efficient approach for online continuous-time influence maximization. *The VLDB Journal* (2021), 1–27.
[18] Song Bian, Qintian Guo, Sibo Wang, and Jeffrey Xu Yu. 2020. Efficient algorithms for budgeted influence maximization on massive social networks. *VLDB* 13, 9 (2020), 1498–1510.
[19] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *SODA*. 946–957.
[20] Taotao Cai, Jianxin Li, Ajmal S Mian, Timos Sellis, Jeffrey Xu Yu, et al. 2020. Target-aware holistic influence maximization in spatial social networks. *TKDE* (2020).
[21] Claudio Castellano and Romualdo Pastor-Satorras. 2010. Thresholds for epidemic spreading in networks. *Physical review letters* 105, 21 (2010), 218701.
[22] Bogdan Cautis, Silviu Maniu, and Nikolaos Tziortziotis. 2019. Adaptive influence maximization. In *SIGKDD*. 3185–3186.
[23] Vineet Chaoji, Sayan Ranu, Rajeev Rastogi, and Rushi Bhatt. 2012. Recommendations to boost content spread in social networks. In *WWW*. 529–538.
[24] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *SIGKDD*. 1029–1038.
[25] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *SIGKDD*. 199–208.
[26] Wei Chen, Yifei Yuan, and Li Zhang. 2010. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*. 88–97.
[27] Suqi Cheng, Huawei Shen, Junming Huang, Guoqing Zhang, and Xueqi Cheng. 2013. Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In *CIKM*. 509–518.
[28] Boreum Choi and Inseong Lee. 2017. Trust in open versus closed social media: The relative influence of user-and marketer-generated content in social network services on customer trust. *Telematics and Informatics* 34, 5 (2017), 550–559.
[29] Reuven Cohen, Keren Erez, Shlomo Havlinl, Mark Newman, Albert-László Barabási, Duncan J Watts, et al. 2011. Resilience of the internet to random breakdowns. In *The Structure and Dynamics of Networks*. 507–509.
[30] The Koblenz Network Collection. 2017. http://konect.uni-koblenz.de.
[31] Federico Coró, Gianlorenzo DâĂŹangelo, and Yllka Velaj. 2021. Link Recommendation for Social Influence Maximization. *TKDD* 15, 6 (2021), 1–23.
[32] Gianlorenzo D'Angelo, Lorenzo Severini, and Yllka Velaj. 2019. Recommending links through influence maximization. *Theor. Comput. Sci.* 764 (2019), 30–41.
[33] Sainyam Galhotra, Akhil Arora, and Shourya Roy. 2016. Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In *SIGMOD*. 1077–1088.
[34] Jacob Goldenberg, Barak Libai, and Eitan Muller. 2001. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review* 9, 3 (2001), 1–18.
[35] Pranava R Goundan and Andreas S Schulz. 2007. Revisiting the greedy approach to submodular set function maximization. *Optimization online* (2007), 1–25.
[36] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. 2011. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *WWW*. 47–48.
[37] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. 2011. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *ICDM*. 211–220.
[38] Mark Granovetter. 1978. Threshold models of collective behavior. *American journal of sociology* 83, 6 (1978), 1420–1443.
[39] Kai Han, Keke Huang, Xiaokui Xiao, Jing Tang, Aixin Sun, and Xueyan Tang. 2018. Efficient algorithms for adaptive influence maximization. *VLDB* 11, 9 (2018), 1029–1040.
[40] Qiang He, Xingwei Wang, Zhencheng Lei, Min Huang, Yuliang Cai, and Lianbo Ma. 2019. TIFIM: A two-stage iterative framework for influence maximization in social networks. *Appl. Math. Comput.* 354 (2019), 338–352.
[41] Huimin Huang, Hong Shen, Zaiqiao Meng, Huajian Chang, and Huaiwen He. 2019. Community-based influence maximization for viral marketing. *Applied Intelligence* 49, 6 (2019), 2137–2150.
[42] Keke Huang, Jing Tang, Kai Han, Xiaokui Xiao, Wei Chen, Aixin Sun, Xueyan Tang, and Andrew Lim. 2020. Efficient approximation algorithms for adaptive influence maximization. *The VLDB Journal* 29, 6 (2020), 1385–1406.
[43] Shixun Huang. 2021. *Capturing and leveraging collective behavior for large-scale social networks analysis*. Ph.D. Dissertation. RMIT University.
[44] Shixun Huang, Zhifeng Bao, J Shane Culpepper, and Bang Zhang. 2019. Finding temporal influential users over evolving social networks. In *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 398–409.
[45] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. Irie: Scalable and robust influence maximization in social networks. In *ICDM*. 918–923.
[46] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *SIGKDD*. 137–146.
[47] Elias Boutros Khalil, Bistra Dilkina, and Le Song. 2014. Scalable diffusion-aware optimization of network topology. In *SIGKDD*. 1226–1235.
[48] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *SIGKDD*. 420–429.
[49] Xiang Li, J David Smith, Thang N Dinh, and My T Thai. 2019. Tiptop:(almost) exact solutions for influence maximization in billion-scale networks. *IEEE/ACM Transactions on Networking* 27, 2 (2019), 649–661.
[50] Wei Liu, Xin Chen, Byeungwoo Jeon, Ling Chen, and Bolun Chen. 2019. Influence maximization on signed networks under independent cascade model. *Applied Intelligence* 49, 3 (2019), 912–928.
[51] Linyuan Lü, Tao Zhou, Qian-Ming Zhang, and H Eugene Stanley. 2016. The H-index of a network node and its relation to degree and coreness. *Nature communications* 7, 1 (2016), 1–7.
[52] Marco Minutoli, Mahantesh Halappanavar, Ananth Kalyanaraman, Arun Sathanur, Ryan Mcclure, and Jason McDermott. 2019. Fast and scalable implementations of influence maximization algorithms. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. 1–12.
[53] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions. *Mathematical programming* 14, 1 (1978), 265–294.
[54] Mark EJ Newman. 2002. Spread of epidemic disease on networks. *Physical review E* 66, 1 (2002), 016128.
[55] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2014. Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations. In *AAAI*. 138–144.
[56] Panpan Shu, Wei Wang, Ming Tang, and Younghae Do. 2015. Numerical identification of epidemic thresholds for susceptible-infected-recovered model on finite-size networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 25, 6 (2015), 063104.

[57] Lichao Sun, Weiran Huang, Philip S Yu, and Wei Chen. 2018. Multi-round influence maximization. In *SIGKDD*. 2249–2258.
[58] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*. 1539–1554.
[59] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*. 75–86.
[60] Yanhao Wang, Qi Fan, Yuchen Li, and Kian-Lee Tan. 2017. Real-time influence maximization on dynamic social streams. *PVLDB* 10, 7 (2017), 805–816.
[61] Hao-Hsiang Wu and Simge Küçükyavuz. 2018. A two-stage stochastic programming approach for influence maximization in social networks. *Computational Optimization and Applications* 69, 3 (2018), 563–595.
[62] Jiarong Xie, Fanhui Meng, Jiachen Sun, Xiao Ma, Gang Yan, and Yanqing Hu. 2021. Detecting and modelling real percolation and phase transitions of information on social media. *Nature Human Behaviour* (2021), 1–8.
[63] Wenguo Yang, Shengminjie Chen, Suixiang Gao, and Ruidong Yan. 2020. Boosting node activity by recommendations in social networks. *Journal of Combinatorial Optimization* 40 (2020), 825–847.
[64] Wenguo Yang, Jianmin Ma, Yi Li, Ruidong Yan, Jing Yuan, Weili Wu, and Deying Li. 2019. Marginal gains to maximize content spread in social networks. *IEEE Transactions on Computational Social Systems* 6, 3 (2019), 479–490.
[65] Kaichen Zhang, Jingbo Zhou, Donglai Tao, Panagiotis Karras, Qing Li, and Hui Xiong. 2020. Geodemographic influence maximization. In *SIGKDD*. 2764–2774.