A Query Approach for Influence Maximization on Specific Users in Social Networks

Jong-Ryul Lee and Chin-Wan Chung

Abstract—Influence maximization is introduced to maximize the profit of viral marketing in social networks. The weakness of influence maximization is that it does not distinguish specific users from others, even if some items can be only useful for the specific users. For such items, it is a better strategy to focus on maximizing the influence on the specific users. In this paper, we formulate an influence maximization problem as query processing to distinguish specific users from others. We show that the query processing problem is NP-hard and its objective function is submodular. We propose an expectation model for the value of the objective function and a fast greedy-based approximation method using the expectation model. For the expectation model, we investigate a relationship of paths between users. For the greedy method, we work out an efficient incremental updating of the marginal gain to our objective function. We conduct experiments to evaluate the proposed method with real-life datasets, and compare the results with those of existing methods that are adapted to the problem. From our experimental results, the proposed method is at least an order of magnitude faster than the existing methods in most cases while achieving high accuracy.

Index Terms—Graph algorithms, influence maximization, independent cascade model, social networks

1 Introduction

Recently, the amount of propagation of information is steadily increased in online social networks such as Facebook and Twitter. To use online social networks as a marketing platform, there are lots of research on how to use the propagation of influence for viral marketing. One of the research problems is influence maximization (IMAX), which aims to find k seed users to maximize the spread of influence among users in social networks. It is proved to be an NP-hard problem by Kempe et al. [1]. Since they proposed a greedy algorithm for the problem, many researchers have proposed various heuristic methods.

Viral marketing is one of the key applications of influence maximization. In viral marketing, an item that a marketer wants to promote is diffused into social networks by "word-of-mouth" communication. From the perspective of marketing, influence maximization provides how to get the maximum profit from all the users in a social network through viral marketing. However, influence maximization is not always the most effective strategy for viral marketing, because there can be some items that are useful to only specific users. These specific users can be a few people with a common interest in a given item, some or all people in a community, or some or all users in a class. There is no limit for being specific users. For example, consider a marketer that is asked to promote a cosmetic product for women through viral marketing. For the cosmetic product, the specific users are female users who are likely to use it and male users who wish to purchase it as a gift for female users. In this case, the marketer does not need to be concerned about the other users because the cosmetic product is not useful to them. Instead, it is a better strategy to focus on maximizing the number of influenced specific users, but influence maximization has the weakness that it cannot distinguish them from the other users. The only way of handling such targets with influence maximization is making a homogeneous graph with the targets and executing influence maximization on the graph. However, the result of this approach should be inaccurate, because there can be some users who are not targets but can strongly influence the targets.

Based on the motivation for target-aware viral marketing, there is an earlier study which focuses on specific targets in influence maximization [2]. In [2], each user has several predefined labels before query processing, a query contains some labels to specify targets whom a marketer wants to influence. However, it is not flexible to predefine labels to each user before query processing, since a query for targets who do not share any existing label cannot be formulated. In this case, we should add a new label including those targets if we want to formulate the query. In addition, if we use a preprocessed structure to compute results quickly, we should update the structure when adding a new label, however the cost for updating is likely to be high. There is another research which can be applied to influence a specific part of a social network. Lu and Lakshmanan [3] devise a variation of influence maximization which separates being influenced and adopting an item for profit maximization. In their problem, if a user is influenced for an item, then the user adopts it with some probability. Thus, by setting the probability for a user who is not a target to adopt an item to 0, their problem can handle maximizing influence on specific targets. However, it requires to check all users one hundred times when we have one hundred items associated with different sets of targets. It is apparently inefficient to check all users when we have many such items. As these two problems, there is no

• *The authors are with the Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea. E-mail: jrlee@islab.kaist.ac.kr, chungcw@kaist.edu.*

Manuscript received 5 Aug. 2013; revised 10 May 2014; accepted 18 May 2014. Date of publication 12 June 2014; date of current version 23 Dec. 2014. Recommended for acceptance by A. Gionis.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2014.2330833

novel problem which processes maximizing influence on specific targets and has the flexibility to handle multiple items without additional costs.

To overcome the weakness of influence maximization and to provide the flexibility, we formulate an influence maximization problem as query processing without predefined labels and call this *IMAX* query processing. In IMAX query processing, a social network is represented by a graph where a node represents an individual and an edge represents a relationship between two individuals such as the friendship. Each edge (u, v) has a probability that u influences v. With such probabilities on edges, the propagation of information is modeled by the independent cascade (IC) model. In the IC model, user u has one-time chance to influence an uninfluenced neighbor v at time $t+1$ when u is influenced at time t. If u fails to influence v, there is no second chance for u to influence v. However, v can be influenced by another user w when there is an edge from w to v and w is influenced. In addition, if u is influenced at time t, u will not be influenced again after time t. Under the IC model, an IMAX query consists of seed set size k and target node set T, and it asks k seed users to maximize the number of influenced users among the users specified in the query. We suppose that the number of the targets is much larger than k. The number of influenced users can be measured by the expected number of influenced users.

The IMAX query problem is worth receiving attention of researchers from two aspects. One is the suitability of IMAX query processing for target-aware viral marketing. As we explained, since the influence maximization problem cannot distinguish targets from the other users, it is not suitable for target-aware viral marketing. However, in the IMAX query problem, we can specify targets explicitly using a set and focus on maximizing influence on those targets. The formulation of the IMAX query problem is sufficient for modeling target-aware viral marketing in general purposes.

Next, the other is efficiency. In the real world, there are lots of users who want to promote many items for various purposes using online social networks. Since IMAX query processing can be a breakthrough to promote an item effectively for those users, the number of potential users of IMAX query processing can be very large. It means that efficiency is a very critical issue for IMAX query processing. However, IMAX query processing is NP-hard like the influence maximization problem. Since the submodularity in the influence maximization problem is preserved in IMAX query processing, several techniques utilized for influence maximization can still be used for IMAX query processing. However, they are inefficient to process the IMAX query. In contrast to influence maximization, we know target nodes that we want to influence when an IMAX query is given. It means that an efficient method for an IMAX query should identify quickly the nodes that strongly influence the targets of the query with preprocessed data. Since existing methods for the influence maximization problem do not utilize the nature of query processing, we need to give attention to query processing to develop a new efficient method for IMAX query processing.

In this paper, we propose a new efficient expectation model for the influence spread of a seed set based on independent maximum influence paths (IMIP) among users. We also show that the new objective function of the new expectation model is submodular. Based on the new expectation model, we present a method to efficiently process an IMAX query. The method consists of identifying local regions containing nodes that influence the target nodes of a query and approximating optimal seeds from the local regions as the result of the query. Identifying such local regions helps to reduce the processing time, when the number of targets in an IMAX query is small compared to the number of all nodes. To approximate optimal seeds, we use a greedy method based on the marginal gain to the new objective function. In addition, we present a method to incrementally update the marginal gain of each user to accelerate the greedy method.

Our contributions. This paper makes the following contributions:

- We identify the limitations of existing researches related to maximizing influence on specific targets. We formulate an influence maximization problem as query processing without predefined labels to address the limitations. We prove that the problem is NP-hard and that the objective function of the IMAX query problem is submodular. Based on the submodularity of the objective function, we present a greedy algorithm for IMAX query processing and show that it has a $(1 - 1/e)$ approximation ratio.
- We propose a new efficient expectation model for influence spread of a seed set. We show that the new objective function of the expectation model is submodular. Based on the new expectation model, we propose a greedy-based approximation method to process an IMAX query with efficient incremental updating of the marginal gain of each user. We also propose an effective method to reduce the number of candidates for optimal seeds by identifying users who strongly influence targets from preprocessed data.
- We experimentally demonstrate that our identifying local influencing regions technique is very powerful and the proposed method is at least an order of magnitude faster than the comparison methods in most cases with high accuracy. Identifying local influencing regions makes the basic greedy algorithm about 6 times faster in the experiments.

The rest of this paper is organized as follows. In Section 2, we review related works. We formulate the IMAX query problem under the IC model in Section 3, and show the NP-hardness and the submodularity of its objective function. In Section 4.1, since the exact computation of influence spread is so expensive, we develop a new expectation model for the influence spread. Then, we devise an efficient algorithm based on the expectation model to process the IMAX query in Section 4.2. We demonstrate the effectiveness and the efficiency of the proposed method through various experiments in Section 5. We make conclusions and outline future works in Section 6.

2 Related Works

IMAX query processing originates from influence maximization. Domingos and Richardson [4] first study influence

maximization as an algorithmic problem based on a Markov random field. Influence maximization is formulated by Kempe et al. under basic diffusion models [1]. Since influence maximization is NP-hard, Kempe et al. propose a greedy method and show that its accuracy is higher than those of other naive methods. Leskovec et al. improve the greedy method with the Cost-Effective Lazy Forward (CELF) selection [5]. Goyal et al. improve the CELF greedy method by exploiting submodularity [6]. Wang et al. [7] propose a community-based greedy method based on identifying influence spreads in communities. Chen et al. [8], [9] focus on reducing the cost for calculating the influence spread. They propose a greedy method based on randomly generated graphs and a degree-based method wherein the largest effective degree nodes are selected as influential seeds. They also propose prefix excluding maximum influence arborescence (PMIA) heuristics where seed nodes influence the other nodes along the maximum influence path from a seed node to each node [9]. In the PMIA heuristics, if the maximum influence path from seed node s to node v includes another seed node s' in their greedy-based algorithm, then their algorithm calculates the next maximum influence path from s to v which does not include s'. However, since calculating it in query processing time is expensive, the PMIA heuristics are inefficient for IMAX query processing. As the PMIA heuristics, the proposed method in this paper also uses such maximum influence paths, but it is more efficient than the PMIA heuristics based on keeping multiple alternative paths on a novel preprocessed structure. Jiang et al. [10] present simulated annealing-based methods that are used to escape the confinement problem of the greedy approach. Jung et al. [11] propose a new method for influence ranking using a system of linear equations, and introduce a way of utilizing their ranking method for influence maximization. These existing methods are not applicable to IMAX query processing, because they cannot be directly used and are not efficient to process an IMAX query.

There are many variations of the influence maximization problem like IMAX query processing. One is competitive influence maximization which considers multiple competing innovations within a social network [12], [13], [14]. Bharathi et al. [12] formulate a new variation of influence maximization to model the case when multiple innovations are competing within a social network. Carnes et al. [13] focus on another case that a new product is introduced into a market, in which a competing product is already being diffused. Irfan and Ortiz [14] introduce a new approach for influence maximization based on non-cooperative game theory and formulate a new class of graphical game modeling the behavior of each individual in social networks.

Another interesting problem is influence maximization in continuous-time diffusion networks [15], [16]. Gomez-Rodriguez and Scholkopf [15] formulate influence maximization on the fully continuous time model of diffusion and propose a method for solving it by exploiting the temporal dynamics of diffusion networks. Du et al. [16] improve the work of Gomez-Rodriguez and Scholkopf in terms of scalability through graphical model inference and neighborhood size estimation.

One noteworthy part of the work of Du et al. is that they use real historical data (i.e., users' action logs) for evaluating a method [16]. There is an earlier study exploiting real historical data for influence maximization [17]. In [17], Goyal et al. propose a method for influence maximization based on real historical data and evaluate it with respect to the actual spread of information. In this work, we also conduct experiments based on the actual spread of information like [17], but in a different way.

Some research has introduced new variations based on several possible constraints in viral marketing. Singer [18] formulates a variation of influence maximization reflecting the cost for being seeds in viral marketing with a budget limit. Goyal et al. [19] formulate one problem to find the minimum size seed set achieving a threshold for the extent of diffusion and the other problem to minimize the time when the threshold is achieved. As we mentioned in Section 1, there is existing research which can set targets for influence maximization. Li et al. [2] focus on maximizing influence on targets whose label is included in a query. In [2], since Li et al. assume that the probability that a user influences another is uniformly distributed, their algorithm is not compared in the experiments. Lu et al. propose another variation of influence maximization which is mentioned in Section 1 [3]. However, they only consider the linear threshold (LT) model which is different from the IC model. Thus, the algorithm proposed in [3] is not compared either. To the best of our knowledge, there is no variation of influence maximization which can handle IMAX query processing without any modification.

In the perspective of targeting a part of a social network, IMAX query processing is related to researches for topic-level social influence analysis [20], [21], [22]. The main tasks of these researches are how to effectively construct a new influence propagation model and estimate social influence between users based on topic-level historical data. The ranking method of [22] based on topic-level influence can be modified to find top-k influencers on specific targets, but it needs topic-level historical data. The two tasks based on topic-level historical data are beyond the scope of this paper.

3 Problem Definition

In this paper, a social network is represented as a directed graph $G = (V, E)$ where V is a set of nodes that represent users and E is a set of directed edges that represent relationships between users. For every edge $(u, v) \in E$, (u, v) has a weight, denoted as $p(u, v)$, that is the probability that u influences v directly. We denote an empty set as $\emptyset$.

Influence diffusion model. We assume that influence is propagated from seed nodes according to the IC model. Let $S \subseteq V$ be a set of seeds such that, for every $s \in S$, s is influenced initially at time 0. Let $S_t \subseteq V$ be a set of nodes each of which is influenced at time t by a node in S_{t-1}. Let $n_{out}(u)$ be a set of out-edge neighbors of u. Then, node $u \in S_t$ has one-time chance to independently influence an uninfluenced neighbor $v \in n_{out}(u)$ with $p(u, v)$ at time $t + 1$. If v is influenced at time $t + 1$, we put v into S_{t+1}. From the initial time 0 with $S_0 = S$, this diffusion process runs iteratively until $S_{t'} = \emptyset$ where $t' \geq 0$. Given a set of targets $T \subseteq V$, the influence spread of seed set S on targets in T, which is

denoted as $\sigma_T(S)$, is measured by the expected number of nodes in T which are influenced by one of the nodes in S.

When $T = V$, $\sigma_T(S)$ becomes the objective function of influence maximization. As proved in [9], computing $\sigma_V(S)$ is #P-hard. For every $T \subseteq V$ such that $T \neq \emptyset$, computing $\sigma_T(S)$ is also #P-hard, since $\sigma_V(S) = \sigma_T(S) + \sigma_{V-T}(S)$. To approximate $\sigma_T(S)$, Monte-Carlo simulations are used in the experiments. After the simulations, $\sigma_T(S)$ is approximated as the average number of influenced users over simulations

IC model and target-aware viral marketing. Given a certain item to be promoted, three kinds of users can exist in target-aware viral marketing. First, there are target users who have an interest in the item. Second, there are non-target users who can be influenced for the item to introduce it to their friends. Finally, there are non-target users who are immune to being influenced for the item, because they do not want to introduce it to their friends.

It is easy to see that the IC model can handle the first case and the second case. However, the IC model cannot handle the third case, because it does not distinguish such immune nodes from the others. Nevertheless, we can easily modify the IC model to support the third case by adding one condition to it. The modified IC model says that user u has one-time chance to influence an uninfluenced neighbor v, *which is not immune*, at time $t+1$ when u is influenced at time t. Fortunately, this modification only marginally affects the proposed method, since immune nodes can be handled like seed nodes except that they do not influence other nodes and are not counted for influence spread. This is because seed nodes cannot be either influenced by another node like immune nodes. Thus, for simplicity, we stick to the original IC model to explain the proposed method in the rest of the paper. Instead, we will explain how to minimally modify the proposed method to handle the immune nodes in Section 4.2.

Propagation probability. For every pair $i, j \in V \times V$ such that there is at least one path from i to j, let the influence from i to j be the probability that i influences j. This is same as the probability that j is influenced when i is the only seed. Recall that for every edge $(u, v) \in E$, $p(u, v)$ is the probability that u influences v through edge (u, v). We call this the direct influence from u to v. $p(u, v)$ does not involve any indirect influence on another path from u to v. Because a path consists of several edges, the indirect influence of a path can be considered as a series of the direct influences of edges on the path. For every path P in G, the influence on path P, denoted $p(P)$, is calculated as,

$$p(P) = \prod_{(u,v)\in P} p(u, v). \quad (1)$$

It is beyond IMAX query processing and influence maximization to determine the direct influence on each edge. We assume that direct influences are given.

Definition 3.1 (Influence Maximization query). *Under the IC model, given a directed graph $G = (V, E)$, an IMAX query asks k-seed set S such that $S \subseteq V$ and S maximizes $\sigma_T(S)$ where T is a set of targets.*

The IMAX query problem is NP-hard and its objective function, σ_T, is submodular.

Theorem 1. *Given a directed graph $G = (V, E)$, IMAX query processing is NP-hard.*

TABLE 1
Frequently Used Notations in This Paper

$\sigma^*_T(S)$	the influence spread of seed set S under the IMIP model
$P^t(i, j)$	the tth IMIP
$\pi^h(i, j)$	the IMIPS from node i to node j
$p_v(S)$	the influence probability of node v given seed set S under the IMIP model
T_v	the influence tree of node v
$\lambda(u)$	the set of the local influencers of node u
$\theta(u)$	the set of the locally influenced targets of node u

Proof. See Section 2.1 in the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2014.2330833. □

Theorem 2. *Given a directed graph $G = (V, E)$ and a set of targets T, $\sigma_T : 2^V \rightarrow \mathbb{R}$, where 2^V is the power set of V, is submodular.*

Proof. See Section 2.2 in the supplemental material, available online. □

One can easily see that σ_T is monotonically increasing. Since σ_T is submodular and monotonic, the greedy method which is described in Algorithm 1 provides a $(1 - 1/\ e)$-approximation. It picks k nodes maximizing the marginal gain to the objective function at each iteration in lines 3-5.

Algorithm 1. Greedy Algorithm ($G = (V, E), k, T$)

input: G: An input graph, k:size of a seed set, T:a set of targets
output: S : Output seed set
1: **begin**
2: $S = \emptyset$;
3: **for** $i = 1$ *to* k **do**
4: $s = \arg\max_{v \in V}(\sigma_T(S \cup \{v\}) - \sigma_T(S))$;
5: $S = S \cup \{s\}$;
6: return S;

Since σ_T is submodular, existing techniques for influence maximization can be applied to IMAX query processing. However, they are not designed to utilize the nature of query processing. For processing an IMAX query efficiently with high accuracy, we need a novel preprocessed structure requiring a reasonable space based on a concrete and effective expectation model for influence spread.

4 Algorithms

4.1 Independent Maximum Influence Paths-Based Expectation Model

Table 1 summarizes frequently used notations in Section 4.

Since calculating the influence spread of a seed set is #P-hard, existing studies usually use the Monte-Carlo simulations to approximate the influence spread. However, the

simulations are still very expensive, so we need a new expectation model to approximate the influence spread.

The hardness of calculating the influence spread lies in that a node can influence another node through various paths and the paths are complicatedly entangled. Thus, the new expectation model starts from simplifying the paths with an important property, called the independence between paths. For every two paths that share the destination and may share the source, if they do not share any node except the destination and the source, the two paths are defined to be independent. There is an interesting observation in the independence between paths. Suppose two paths P, Q are independent and they do not share the source. If the source of P is a seed, the other nodes in P can be influenced by the seed but nodes in Q cannot be influenced by the seed. This observation leads to (2).

Let the influence probability of a node $v \in V$ be the probability that v is influenced by a node of a given seed set S and it is denoted as $p(S, v)$. For every node $v \in V$, if all paths which start from a seed and have v as the destination are independent of each other, by the IC model, the influence probability of v is computed to be,

$$p(S, v) = 1 - \prod_{P \in \pi(S,v)} (1 - p(P)), \tag{2}$$

where S is the set of the seeds and $\pi(S, v)$ is the set of all the paths from a seed in S to v.

Next, to simplify various paths between nodes effectively, we focus on finding a path which represents the influence between two nodes. For every pair $(i, j) \in V \times V$, we denote the set of all paths from i to j as $\pi(i, j)$. Let us define the maximum influence path $P^1(i, j)$ from i to j as $P^1(i, j) = \arg\max_{P \in \pi(i,j)} p(P)$. From the maximum influence path, we derive a more general concept representing the influence between two nodes using the independence between paths, which is the independent maximum influence path set (IMIPS). For any path P and set of paths π, when path P is independent from all paths in set π, we denote this as $P \perp\!\!\!\perp \pi$.

Definition 4.1 (Independent Maximum Influence Path Set). *For every pair $(i, j) \in V \times V$, the independent maximum influence path set from i to j with an integer parameter h, denoted as $\pi^h(i, j)$, is,*

$$\pi^h(i, j) = \begin{cases} \emptyset & \text{for } h = 0 \\ \bigcup_{t=1}^{h} \{P^t(i, j)\} & \text{for } h \geq 1, \end{cases} \tag{3}$$

where $P^t(i, j) = argmax_{P \in \{P | P \in \pi(i,j) \wedge P \perp\!\!\!\perp \pi^{t-1}(i,j)\}} p(P)$. We call $P^t(i, j)$ the tth independent maximum influence path from i to j.

Given two nodes i and j, IMIPS $\pi^h(i, j)$ is computed as follows. Initially, $P^1(i, j)$ is computed by running the Dijkstra's algorithm. Then, for $2 \leq t \leq h$, $P^t(i, j)$ is iteratively computed through the Dijkstra's algorithm after excluding nodes on $t - 1$ already computed IMIPs from G except i and j. In this way, we can construct the IMIPS from a node to another.

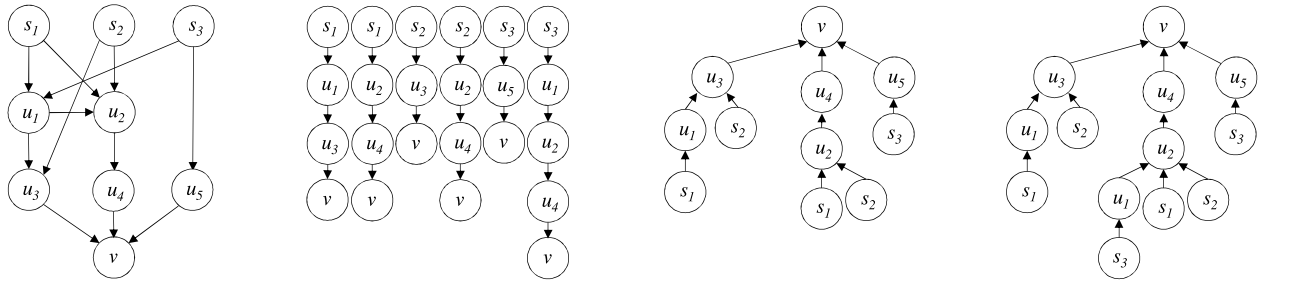
A new expectation model. To approximate the influence spread efficiently, we propose a new model, which is the independent maximum influence path-based expectation model (IMIP model). The IMIP model says that a node influences another node through one of paths in their IMIPS.

The intuition of the IMIP model is as follows. Consider the situation that a node becomes a new seed in the greedy algorithm. As we mentioned, when the new seed is on the maximum influence path from node v to another node u, the PMIA heuristics in [9] find the alternative maximum influence path from v to u, since the seed blocks v on the maximum influence path. However, it is quite expensive to compute it in the query processing time. In the IMIP model, even if the new seed is on one of IMIPs from v to u, we can efficiently estimate the influence from v to u using the independence among the other IMIPs and (2). That is the intuition of the IMIP model.

Error analysis. Since the IMIP model covers only a constant number of independent paths from a node to another, there must be an error of the IMIP model. However, we claim that the error of the IMIP model is usually small in online social networks. This claim is based on two observations. First, in online social networks, information is usually diffused from a seed within a very small number of hops [23], [24], [25]. For example, the portion of sampled retweets within two-hops is more than 95.8 percent of the total sampled retweets [25]. It means that the lengths of strong influence paths are mostly one hop or two hops. Second, one-hop and two-hop paths, which share the same destination, are always independent of each other by definition. Based on these observations, since the IMIP model covers such strong influence paths, the claim makes sense. In addition, we experimentally verify the claim in the supplemental material, available online.

Under the IMIP model, when a node $s \in V$ is the only seed, the influence from s to a non-seed node $v \in V$ is easily calculated to be $1 - \prod_{P \in \pi^h(s,v)} (1 - p(P))$ by (2). However, when there are multiple seeds in a seed set S, calculating the influence probability of non-seed node v is not trivial, because there is no guarantee that all IMIPs starting from different seeds are independent of each other.

Influence tree. Let us introduce an efficient way of handling the issue of multiple seeds. Since a node is influenced only through the IMIPs from the seeds to the node under the IMIP model, we consider a structure consisting of the IMIPs to compute the influence probability of the node as follows. Let us consider a seed set $S \subseteq V$ and a non-seed node $v \in V$. There are at most $|S|h$ IMIPs in $\bigcup_{s \in S} \pi^h(s, v)$. To compute the influence probability of v given seed set S under the IMIP model, we use a directed tree in which the root is a node labeled as v. We call it the influence tree of v, denote it as T_v, and build it as follows. T_v initially has only one node v, the root. For each IMIP P in $\bigcup_{s \in S} \pi^h(s, v)$, first we find the common part of P and T_v in a sequence starting from v in the reverse direction of P. Then, we copy nodes and edges (with weights) in the remaining part of P to the position before the first node of the common part in T_v. For example, Fig. 1 shows an example of IMIPs and an influence tree. The original graph is shown Fig. 1a and the IMIPs from the seeds (s_1, s_2, s_3) to node v are shown in Fig. 1b. Consider that we look at each IMIP from left to right in Fig. 1b to build the influence tree of v. Then, the last IMIP is $< s_3, u_1, u_2, u_4, v >$. In addition, Fig. 1c shows the situation



(a) A part of graph G (b) The IMIPs from s_1, s_2, and s_3 to v (c) Before processing the last IMIP to build the influence tree of v (d) The influence tree of v

Fig. 1. An example of IMIPs and an influence tree.

that we are looking at the last IMIP to build the influence tree of v. To process the last IMIP, we find the common part $< u_2, u_4, v >$ in T_v. Then, we copy u_1, $s_3, (s_3, u_1)$, and (u_1, u_2) to the position before u_2 of the common part. After processing the last IMIP, the influence tree of v is built as described in Fig. 1d.

Computing the influence probability. Now, we can compute the influence probability of node v given seed set S under the IMIP model. Let $p_v(S)$ denote the influence probability of v given seed set S under the IMIP model. Recall that node v is influenced only through the IMIPs from the seeds to v under the IMIP model. In addition, all IMIPs from the seeds to v are included in T_v. Thus, the influence probability of v in G under the IMIP model is the same as the influence probability of the root in T_v where all leaves are seeds in T_v. Note that an influence tree consists of copied nodes and the influence probability of a copied node u is different from the influence probability of u's original node in G, except the root. Then, we can recursively compute the influence probability of node v in G by calling $influ(v, root(v))$, where $root(v)$ is the root in T_v (in case of v, there is only one v in the influence tree of v, and $root(v) = v$), as described in Algorithm 2. We denote the immediate predecessors of a copied node i as $IN(i)$. In lines 2-3 of Algorithm 2, if i is a leaf, then the algorithm returns 1, since a leaf corresponds to a seed. Otherwise, in lines 5-9, the algorithm computes the influence probability of i according to the IC model, and then returns it. Thus, $influ(v, root(v))$ returns the influence probability of v.

Algorithm 2. $influ(v, i)$

input v: a node in V, i: a copied node in T_v
output the influence probability of i when S is a seed set under the IMIP model

1: **begin**
2: **if** i *is a leaf* **then**
3: return 1;
4: **else**
5: $p = 1$;
6: **for** $n \in IN(i)$ **do**
7: $p = p(1 - p(n, i)influ(v, n))$;
8: $p = 1 - p$;
9: return p;

One might worry about the case that two copied nodes in an influence tree, which correspond to the same node in G, have different influence probabilities. In Fig. 1, there are two nodes in T_v which correspond to u_1, but those nodes can have different probabilities in T_v. In fact, this case is still consistent to the IMIP model. Assume that the two nodes share the same influence probability in T_v. It means that the influence on the path from s_1 to u_1 is considered when computing the influence on the path from s_3 to u_2. However, this situation obviously violates the IMIP model, since the path $< s_1, u_1, u_2, u_4, v >$ is considered for the influence probability of v, even if it is not an IMIP. That is why we maintain different influence probabilities in T_v for a node in G.

Submodularity. Based on Algorithm 2, given a target set T and a seed set S, we define the influence spread of seed set S under the IMIP model as $\sigma^*_T(S) = \sum_{v \in T} p_v(S)$. We prove that function $\sigma^*_T : 2^V \rightarrow \mathbb{R}$ is submodular.

Lemma 1 (Incremental Update Lemma). *Given a node $v \in V$, a seed $s \in S$, and a path $P \in \pi^h(s, v)$, consider that P is being added into T_v. Then, for any edge (i, j) in T_v, which is corresponded to an edge on P, if $p(i)$ increases, $p(j)$ increases as,*

$$p(j) = 1 - \frac{(1 - \hat{p}(j))(1 - p(i, j)p(i))}{(1 - p(i, j)\hat{p}(i))}, \tag{4}$$

where for node $a \in T_v$, $p(a)$ is a's new influence probability in T_v, $\hat{p}(a)$ is a's old influence probability in T_v, and $p(i, j)\hat{p}(i) \neq 1$.

Proof. See Section 2.3 in the supplemental material, available online. □

Theorem 3. *Given a directed graph $G = (V, E)$ and a set of targets T, function $\sigma^*_T : 2^V \rightarrow \mathbb{R}$ is submodular.*

Proof. See Section 2.4 in the supplemental material, available online. □

From influence spread function σ^*_T under the IMIP model, we will propose an efficient greedy-based method to approximate IMAX query processing. In fact, since Algorithm 2 causes still expensive cost, we focus on efficiently calculating the marginal gain to σ^*_T in Section 4.2.2.

4.2 Query Processing Using Local Region

To process an IMAX query efficiently, first we identify which nodes strongly influence targets, and consider such

nodes as candidates for optimal seeds. Then, we approximate the optimal seeds with the candidates.

4.2.1 Identifying Local Influencing Regions

Since the solution space of a given IMAX query can be very large, it is significant to consider how to efficiently identify the candidates, for optimal seeds, which strongly influence targets in query processing time. Thus, we devise an efficient way of identifying those candidates. The basic idea is that for any node $v \in V$, storing nodes which influence v more than some threshold and retrieving them when v is a target.

Recall that direct influences are usually low in social networks as described in [23], [24], [25]. Thus, nodes that can strongly influence targets are likely to be near from the targets. Based on this observation, for any node $v \in V$, we define a local influencer of v as a node which can influence v with the probability larger than or equal to the influence threshold α such that $0 < \alpha \leq 1$, under the IMIP model. Then, we define a local influencing region of $v \in V$ as the set of the local influencers of v. We denote it as $\lambda(v)$. Similarly, we define a locally influenced target of $v \in V$ as a node in T which has v as a local influencer of it. Let us define $\theta(v)$ as the set of the locally influenced targets of v. For every $v \in V$, $\lambda(v)$ can be computed in preprocessing time, while $\theta(v)$ should be computed in query processing time. By storing the local influencers of each node in V and retrieving them in query processing time, we can efficiently identify candidates for optimal seeds which strongly influence targets.

To find candidates for optimal seeds, the proposed method finds all local influencing regions of given targets as shown in Algorithm 3. In lines 3-10, for each target $t \in T$, $\theta(i)$ of local influencer i in $\lambda(t)$ is computed and $\sigma^*_T(\{i\})$ is computed incrementally. Then, for each $m \in M$, m is inserted into C whenever $\sigma^*_T(\{m\}) \geq \beta$ in lines 11-13. The nodes of C are our candidates for optimal seeds given an IMAX query. The minimum value of β is 1, because for any $t \in T$, $\sigma^*_T(\{t\}) \geq 1$. In addition, if k is the size of the seed set, $|T| > k$, and $\beta = 1$, then this filtering step does not affect the accuracy because picking $t \in T$ is always better than picking i such that $\sigma^*_T(\{i\}) < 1$. Therefore, under the IMIP model, there is no accuracy drop in this filtering step with $\beta = 1$. On the other hand, in Algorithm 3, we only consider all the local influencing regions of targets and the targets themselves as candidates. It might reduce the effectiveness of the proposed method, because for any target $t \in T$, there can exist some nodes w such that $p_t(\{w\}) < \alpha$ and $\sigma^*_T(\{w\}) > \beta$. Thus, we analyze the approximation guarantee given by removing non-local influencers in Theorem 4.

Theorem 4. *Given a set of targets T and seed set size k, under the IMIP model, when $\beta = 1$, removing non-local influencers gives an approximation guarantee as follows:*

$$\begin{cases} \frac{k}{|T|(1-(1-(\alpha-\epsilon)^k)} & \text{if } |T|(1-(1-(\alpha-\epsilon))^k) > k \\ \text{no approximation} & \text{otherwise,} \end{cases}$$

where $0 < \epsilon < \alpha$.

Proof. See Section 2.5 in the supplemental material, available online. □

Since α is usually very small value, we expect that removing non-local influencers does not much reduce effectiveness. We will show experimental results related to this analysis in Section 5.

Algorithm 3 requires only $O(|T|n_\lambda)$ time where $n_\lambda = \max_{v \in V} |\lambda(v)|$, since we can get the influence from a local influencer to a node under the IMIP model in constant time using preprocessed data. The preprocessed data are computed as follows.

Algorithm 3. $findLR(G, T)$

input: $G = (V, E)$: an input graph, T: a set of targets
output: C: a set of candidates

```
1: begin
2:     M = ∅, C = ∅;
3:     for t ∈ T do
4:         for i ∈ λ(t) do
5:             if i is not in M then
6:                 σ*_T({i}) = 0;
7:                 insert i into M;
8:             σ*_T({i}) = σ*_T({i}) + p_t({i});
9:             insert t into θ(i);
10:        insert t into C;
11:    for m ∈ M do
12:        if σ*_T({m}) ≥ β then
13:            insert m into C;
14:    return C;
```

To efficiently identify local influencing regions for targets in query processing time under the IMIP model, we need to calculate the local influencers of all the nodes of graph G and all IMIPs from the local influencers to the nodes in preprocessing time. For efficient preprocessing, Theorem 5 is used for limiting the search region of finding IMIPs between two nodes.

Lemma 2. *Given two nodes $u, v \in V$, under the IMIP model, if u is a local influencer of v, $p(P^1(u, v)) \geq 1 - \sqrt[h]{1-\alpha}$ where h is the maximum number of the IMIPS from u to v.*

Proof. See Section 2.6 in the supplemental material, available online. □

Theorem 5. *Given two nodes $u, v \in V$, under the IMIP model, for any integer t such that $2 \leq t \leq h$, if u is a local influencer of v, $p(P^t(u, v)) \geq 1 - \sqrt[h-(t-1)]{\frac{1-\alpha}{F}}$ where $F = \prod_{t'=1}^{t-1}(1 - p(P^{t'}(u, v)))$, and h is the maximum number of the IMIPS from u to v.*

Proof. See Section 2.7 in the supplemental material, available online. □

Algorithm 4 shows how to efficiently find the local influencers of all nodes in V and all related IMIPs under the IMIP model. In lines 3-4, we find candidates for the local influencers of v by computing the maximum influence paths going to v with the Dijsktra's algorithm by taking logarithms and negating each logarithm value. By Lemma 2, we limit the search region of the Dijkstra's algorithm using our lower bound of $p(P^1(u, v))$ for u to be a local influencer of v. In lines 9-21, the rest of the IMIPs from u to v are found. In line 13, we find $P^t(u, v)$ with the Dijkstra's algorithm also. By Theorem 5, we limit the search region of the Dijkstra's

algorithm using δ or our lower bound of $p(P^t(u,v))$ for u to be a local influencer of v. δ is a parameter used to limit the search region more effectively. By definition, our lower bound of $p(P^t(u,v))$ can be extremely small, and then we cannot limit the search region effectively in that case. δ is used to prevent the search region from being a very large part of G, and it should be a small value, lower than $1-\sqrt[h]{1-\alpha}$, to reduce the loss of accuracy. In line 14, we test whether $P^t(u,v)$ is empty or not. If $P^t(u,v)$ is empty and $1-F<\alpha$, v cannot be a local influencer of v by definition. In addition, If $P^t(u,v)$ is not empty and $F(1-P^t(u,v))=0$, we do not need to proceed to the next iteration, because now we know that $p_v(\{u\})=1$. In line 7 and line 21, V' is used to guarantee the independence between IMIPs which are computed here. If a node is included in one of the IMIPs, then the node is excluded from the next iteration. By Algorithm 4, we have the local influencing regions of every node in G and all IMIPSs related to them. We use them to approximate optimal seeds efficiently. In Algorithm 4, all nodes in a graph are checked and we find IMIPs based on the Dijkstra's algorithm. Algorithm 4 requires $O(nn_{\lambda'}c_{di}h)$ time where n is $|V|$, $n_{\lambda'}=\max_{v\in V}|\lambda'(v)|$ ($\lambda'(v)$ is defined in line 4), and c_{di} is the maximum cost for running the Dijkstra's algorithm. It also requires $O(nn_\lambda n_P h)$ space where $n_P=\max_{s,t\in V}|P^i(s,t)|$ for $1\le i\le h$.

Reducing the search space of the Dijkstra's algorithm. To speed up Algorithm 4, we exploit the result of the Dijkstra's algorithm in line 3 to reduce the search space of the Dijkstra's algorithm in line 13. After line 3, we know $\lambda'(v)$ and $p(P^1(u,v))$ for any node $u\in\lambda'(v)$. We claim that when the Dijkstra's algorithm in line 13 is executed, we do not need to visit a node w such that $w\notin\lambda'(v)$. The reason is that if w is not in $\lambda'(v)$, then $p(P^1(w,v))<1-\sqrt[h]{1-\alpha}$. In addition, consider a current node x in the loop for the relaxation step of the Dijkstra's algorithm in line 13. We claim that we do not need to visit any neighbor x' of x such that $p(P^1(u,x))p(x,x')p(P^1(x',\ v))<bound$ where $bound$ is defined in line 12. Note that $p(P^1(x',v))$ is computed in line 3 if x' is included in $\lambda'(v)$, and we do not need to consider x' such that $x'\notin\lambda'(v)$ by the first claim. Based on these two claims, we efficiently implement the Dijkstra's algorithm in line 13.

4.2.2 Approximating Optimal Seeds

From Section 4.2.1, we identify candidates each of which has the sum of influences on targets is larger than or equal to β. In this section, we propose an efficient greedy-based method to approximate optimal seeds from the candidates.

Recall that σ_T^* is submodular. In addition, it is obviously monotonic. Thus, if we modify the greedy algorithm to use σ_T^* instead of σ_T as an objective function, it has a $(1-1/e)$ approximation ratio under the IMIP model. However, Algorithm 2 requires too much cost for looking at the entire part of an influence tree. Thus, we devise an efficient way of calculating the marginal gain to σ_T^*.

Preprocessed structure. We construct a novel data structure for efficiently calculating the influence spread under the IMIP model. Let us consider p nodes which are the local influencers of node v. We define the local influence tree of v as the influence tree of v when the p local influencers are seeds. In preprocessing time, for each node v, we build the local influence tree of v with information computed by Algorithm 4. As we mentioned in Section 4.1, v's local influence tree can be easily computed by traversing all IMIPs from v's local influencers to v once. The cost of building the local influence trees of all nodes is much smaller than that of Algorithm 4.

Incremental updating. Using the local influence tree of a node, we can incrementally update the influence probability of the node when a new seed is added in the seed set. Recall that the greedy algorithm picks k nodes as new seeds iteratively. To make the greedy algorithm much faster under the IMIP model, given a current seed set S, we should efficiently calculate the marginal gain of $x\in V\setminus S$ with respect to the influence probability of any node $u\in V\setminus S$, which is $p_u(S\cup\{x\})-p_u(S)$. Algorithm 5 shows how to efficiently calculate the marginal gain using the local influence tree. It is also used to update the influence probability of a node when a new seed is added into the current seed set.

Algorithm 4. $storingLR(G,h,\delta)$

input: $G=(V,E)$: an input graph, h: the maximum number of an IMIPS, δ: a parameter in (0,$1-\sqrt[h]{1-\alpha}$)

1: **begin**
2: **for** $v\in V$ **do**
3: compute $P^1(u,v)$ s.t. $u\in V\wedge p(P^1(u,v))>1-\sqrt[h]{1-\alpha}$;
4: $\lambda'(v)=\{u|u\in V, p(P^1(u,v))>1-\sqrt[h]{1-\alpha}\}$;
5: **for** $u\in\lambda'(v)$ **do**
6: insert $P^1(u,v)$ into $\pi^h(u,v)$;
7: $V'=\bigcup_{(i,j)\in P^1(u,v)}\{j\}-\{v\}$;
8: $flag=false$, $h'=0$;
9: **for** $t=2$ to h **do**
10: $h'=t$;
11: $F=\prod_{t'=1}^{t-1}(1-p(P^{t'}(u,v)))$;
12: $bound=min\left\{1-\sqrt[h-(t-1)]{\frac{1-\alpha}{F}},\delta\right\}$;
13: compute $P^t(u,v)$ with $V-V'$ s.t. $p(P^t(u,v))>bound$;
14: **if** $P^t(u,v)$ *is empty* **then**
15: **if** $1-F<\alpha$ **then**
16: $flag=true$;
17: break;
18: **else if** $F(1-p(P^t(u,v)))=0$ **then**
19: break;
20: insert $P^t(u,v)$ into $\pi^h(u,v)$;
21: $V'=V'\cup\bigcup_{(i,j)\in P^t(u,v)}\{j\}-\{v\}$;
22: **if** $flag\neq true$ **then**
23: $p_v(\{u\})=0$;
24: **for** $t=1$ *to* h' **do**
25: $p_v(\{u\})=1-(1-p_v(\{u\}))(1-p(P^t(u,v)))$;
26: insert u into $\lambda(v)$

Algorithm 5 is used to handle two tasks: calculating the marginal gain and updating the influence probability. In Algorithm 5 and Algorithm 6, for any copied node u, we

denote the new influence probability of u when s is a new seed as $p(u)$, and the old influence probability of u as $\hat{p}(u)$. Consider that $p(u)$ and $\hat{p}(u)$ are global variables. In addition, we denote the immediate successor of any copied node u as $SUCC(u)$. Note that a copied node can have only one immediate successor in the influence tree where it participates. When node t in the original graph is copied to the local influence tree of t, t becomes the root of the local influence tree T_t. In this case, to distinguish t in the original graph and t in T_t, the copied node t in T_t is denoted by $root(t)$. In line 3 of Algorithm 5, $leaves(s,t)$ denotes the set of nodes, each of which is copied from new seed s as a leaf node in T_t. For each copied node $s' \in leaves(s,t)$, Algorithm 5 calls Algorithm 6 after setting $p(s')$ to 1.

Algorithm 5. $traverseLIT(s,t,update)$

input: s: a new seed, t: a node in $\theta(s)$, $update$: a flag for updating
output: mg: the marginal gain of s with respect to the influence probability of t

1: **begin**
2: $\hat{p}(t) = p(t)$
3: **for** $s' \in leaves(s,t)$ **do**
4: $\hat{p}(s') = p(s')$;
5: $p(s') = 1$;
6: $next(SUCC(s'), root(t), s')$;
7: $mg = p(t) - \hat{p}(t)$;
8: **if** $update = false$ **then**
9: $p(v) = \hat{p}(v)$ such that $v \in T_t$ and $p(v) \neq \hat{p}(v)$;
10: return mg;

In Algorithm 6, the new influence probability of copied node v is computed in line 3 and line 6. line 3 and line 6 come from (4) of Lemma 1. Algorithm 6 recursively calls itself to reflect the change of the influence probability of v to the influence probability of $SUCC(v)$. If $p(SUCC(v))) = 1$, the influence probability of v cannot affect that of $SUCC(v)$ by (4). When $v = t$, Algorithm 6 returns and we are back to line 7 of Algorithm 5. In line 7 of Algorithm 5, we compute the marginal gain of node s with respect to the influence probability of t. In lines 8-9, if $update$ is false, we restore $p(v)$ if the influence probability of v is changed. This restoration can be done efficiently by keeping visited nodes in lines 3-6. In line 10, Algorithm 5 returns the marginal gain.

Algorithm 6. $next(v,t,v')$

input: v: a current copied node, t: the root node, v': the immediate predecessor of v

1: **begin**
2: **if** $v = t$ **then**
3: $p(v) = 1 - \frac{(1-p(v))(1-p(v')p(v',v))}{(1-\hat{p}(v')p(v',a))}$;
4: return;
5: $\hat{p}(v) = p(v)$;
6: $p(v) = 1 - \frac{(1-p(v))(1-p(v')p(v',v))}{(1-\hat{p}(v')p(v',a))}$;
7: **if** $p(SUCC(v)) \neq 1$ **then**
8: $next(SUCC(v), t, v)$;
9: return;

Algorithm 5 requires $O(n_P h)$ time in the worst case since it is equivalent to traversing IMIPs from a node to another.

Algorithm 7. IMIP-based IMAX query(G,T,k)

input: $G = (V,E)$: an input graph, T: a set of targets, k: the size of the output seed set
output: S: a seed set

1: **begin**
2: $S = \emptyset$;
3: // find candidates and compute θ function
4: $C = findLR(G,T)$;
5: **for** $c \in C$ **do**
6: $\Delta\sigma_T^S(c) = \sigma_T^*(\{c\})$;
7: **for** $i = 1$ *to* k **do**
8: $s = \arg\max_{c \in C-S} \Delta\sigma_T^S(c)$;
9: $S = S \cup \{s\}$;
10: **for** $c \in copied(s)$ **do**
11: $p(c) = 1$;
12: **for** $t \in \theta(s)$ **do**
13: // update the influence probability of t
14: $traverseLIT(s,t,true)$;
15: **if** s *in* T **then**
16: **for** $s' \in \lambda(s)$ **do**
17: **if** $s' \in C$ **then**
18: // reduce $\Delta\sigma_T^S(s')$ since s is a seed
19: $\Delta\sigma_T^S(s') = \Delta\sigma_T^S(s') - \Delta\sigma_T^S(s',s)$;
20: $\Delta\sigma_T^S(s',s) = 0$;
21: **for** $t \in \theta^*(s)$ **do**
22: **if** $t \in S$ *or* $t \notin T$ **then**
23: continue;
24: **for** $l \in \lambda(t)$ **do**
25: $temp = \Delta\sigma_T^S(l,t)$;
26: $\Delta\sigma_T^S(l,t) = traverseLIT(l,t,false)$;
27: $\Delta\sigma_T^S(l) = \Delta\sigma_T^S(l) - temp + \Delta\sigma_T^S(l,t)$;

IMIP-based IMAX query algorithm. Let us explain the proposed method for IMAX query processing. Given a directed graph $G = (V,E)$, a set of targets $T \subseteq V$ and a set of seeds $S \subseteq V$, for any two nodes $u, v \in V$, let $\Delta\sigma_T^S(u,v)$ denote the marginal gain to the influence probability of v when u becomes a new seed under the IMIP model. By definition, $\Delta\sigma_T^S(u,v) = p_v(S \cup \{u\}) - p_v(S)$. Then, the marginal gain to the new objective function when u becomes a new seed, denoted as $\Delta\sigma_T^S(u)$, is,

$$\Delta\sigma_T^S(u) = \sum_{v \in T-S} \Delta\sigma_T^S(u,v). \quad (5)$$

In addition, for any node $v \in V$, let $\theta^*(v)$ denote the set of nodes which have a copied node of v in their influence tree. Let $copied(v)$ denote the set of the copied nodes of v. Algorithm 7 is a greedy approach which picks node $s \in V$, maximizing $\Delta\sigma_T^S(s)$ per iteration. Our preprocessed structure helps to quickly calculate $\Delta\sigma_T^S(s)$ with incremental updating. In line 4, candidates for optimal seeds and $\theta(i)$ for each $i \in C$ are computed. The marginal gains of the candidates with the empty seed set are computed in lines 5-6. In lines 7-27, the output k-seed set is computed. The influence

probabilities of the copied nodes of s are updated to 1 in lines 10-11. The influence probabilities of the nodes in $\theta(s)$ are updated in lines 12-14. If s is one of the targets in T, the marginal gain of each node in $\lambda(s)$ is decreased in lines 15-20. Since s becomes a new seed, the copied nodes of s decrease the marginal gains of all the local influencers of nodes in $\theta^*(s)$ in lines 21-27. After the main loop in lines 7-27 is completed, S becomes the output k-seed set.

Algorithm 7 requires $O(|T|n_\lambda)$ time for identifying candidates with $findLR(G,T)$ in line 4. In line 8, selecting s in V that maximizes $\Delta\sigma_T^S(s)$ can be implemented with a priority queue. Then, selecting a new seed requires $O(1)$ time and the update of the queue requires $O(\log(|T|n_\lambda))$ time. In lines 10-11, the influence probability $p(c)$ of each node c in $copied(s)$ is updated in $O(|copied(s)|)$ time. Recall that IMIPs from a node to another do not share any intermediate node. It leads to the fact that in a local influence tree, the number of the copies of a node is always lower than the number of local influencers. Thus, $O(copied(s)) = O(|T|n_\lambda)$. Next, let $n_\theta = \max_{v\in V}\theta(v)$. Then, since Algorithm 5 takes $O(n_P h)$ time, updating the influence probabilities of nodes in $\theta(s)$ takes $O(n_\theta n_P h)$ time in lines 12-14. In lines 15-20, the marginal gain of each candidate that has s as a locally influenced target is updated in $O(n_\lambda \log(|T|n_\lambda))$ time. $\log(|T|n_\lambda)$ is for the priority queue update. In lines 21-27, the cost for updating the marginal gains of all the local influencers of nodes in $\theta^*(s)$ is $O(|T|n_\lambda(n_P h + \log(|T|n_\lambda)))$, because $|\theta^*(s)| = O(|T|)$. Therefore, the total time complexity of Algorithm 7 is $O(|T|n_\lambda + k(|T|n_\lambda(n_P h + \log(|T|n_\lambda)))) = O(k(|T|n_\lambda(n_P h + \log(|T|n_\lambda))))$.

Handling immune nodes. Recall that immune nodes cannot be influenced by any other node. We can handle such immune nodes by modifying the proposed method as follows. In Algorithm 7, the marginal gain of each candidate with the empty seed set is computed in lines 5-6. From the initial marginal gain, the marginal gain of each candidate is incrementally updated in lines 7-27. By definition, when a node i is set to an immune node, the influence probability of another node, whose local influence tree contains a copy of i, can be changed. If the influence probability of a node is changed, then the marginal gain of each local influencer of the node is also changed. Thus, between lines 6 and 7 in Algorithm 7, for each immune node i and each node $t \in \theta^*(i)$, we update the initial marginal gain of every candidate which is a local influencer of t. This update can be done using the same procedure as lines 21-27 in Algorithm 7 except that s is replaced with immune node i. Next, we add a condition that $SUCC(v)$ is not immune to line 7 in Algorithm 6 with the AND operation. This is because the influence probability of a node is incrementally computed according to Algorithm 6 to compute the marginal gains of candidates in Algorithm 7. With these two minor modifications, the proposed method can handle the immune nodes.

5 Experiments

We conduct various experiments with several comparison methods and real data sets. In these experiments, we focus on testing the efficiency of the proposed method based on the IMIP model and the incremental updating. We run the experiments on an Intel(R) i7-990X 3.46 GHz CPU machine with 48 GB RAM.

5.1 Experimental Environment

Comparison methods. In the experiments, we use the following six comparison methods.

- IMIP is the proposed method in this paper.
- CELF++ is an improved greedy algorithm exploiting submodularity in [6].
- CELF++LR is the CELF++ method with identifying local influencing regions.
- PMIA is a greedy-based algorithm based on maximum influence paths between nodes [9]. In PMIA, parameter θ is used to prune out maximum influence paths having low influence. We set $\theta = 1 - \sqrt[h]{1-\alpha}$, because $1 - \sqrt[h]{1-\alpha}$ is used for the same purpose as θ in this paper.
- IRIE is one of recent algorithms for influence maximization [11]. For IRIE, we set $\alpha = 0.7$ which is a damping factor, and it is not the same as ours. As PMIA and IMIP, IRIE also uses the maximum influence path from a seed to a node with the same parameter θ. For IRIE, we use the same value of θ in PMIA with the same reason.
- CD is the greedy method using the CD model in [19]. The CD model is a probabilistic model based on users' historical action logs. We use this method only for the experiment related to the actual influence spread.

Direct influence model. To model direct influence, which is the probability that a user influences a neighbor, we use following two models. The Bernoulli (BN) model says that for any edge $(u,v) \in E$, it is considered as a Bernoulli trial that u influences v. Then, as the maximum likelihood estimation, the direct influence on (u,v) can be estimated to be $n_{u\to v}/n_u$, where $n_{u\to v}$ is the number of actions diffused from u to v, and n_u is the number of actions conducted by u. Kempe et al. introduce the weighted cascade (WC) model in [1]. The WC model says that direct influences from the neighbors of node v to v are equal to $1/din_v$ where din_v is the in-degree of v.

*Data sets.*For experiments, we use eight real data sets. Wiki-Vote, Epinions, Slashdot, Amazon, Pokec, and Gowalla are published online by Jure Leskovec.[1] The Flixster data set is used in [19], and published online by Mohsen Jamali.[2] The Digg data set was introduced in [26]. Wiki-Vote is based on the elections for promoting adminship, and there is an edge from u to v when user u votes on user v. Epinions is a who-trust-whom online social network and Slashdot is a technology-related news website where there are friendships between users. Amazon is a co-purchasing network where there is an edge from u to v when u and v are co-purchased frequently. Gowalla is a location-based social network service where users can share their locations with friends. Digg is a social news site, and Flixster is a social networking site where a user can share movie reviews and ratings with friends. Pokec is the most popular online social network service in Slovakia. In addition to relationship

1. http://snap.stanford.edu/data/.
2. http://www.cs.sfu.ca/ sja25/personal/data sets/.

TABLE 2
Statistics of Our Data Sets

Data set	Wiki-Vote	Epinions	Slashdot	Amazon	Pokec
Node	7K	76K	77K	262K	1.6M
Edge	104K	509K	906K	1.2M	30.6K
Degree	14.6	6.7	11.7	4.7	18.8
Data set	Gowalla	Digg	Flixster		
Node	197K	279K	0.8M		
Edge	1.9M	1.7M	11.8M		
Degree	9.7	6.2	15		
Action	6.4M	3M	8.2M		

data, the Pokec data set contains profile data. We will use the profile data to specify real targets. Table 2 illustrates the statistics of the eight data sets. In Table 2, Degree denotes the average degree of nodes and Action denotes the number of action logs. A log consists of a user, an item, and the time when a user is influenced for the item. Gowalla, Digg, and Flixster contain actions logs as well as graph data. They are used for experiments about actual influence spread and the BN model. Note that if there are multiple action logs for a pair of a node and an item, we only use the earliest one, because an already influenced node cannot be influenced again for the same item. In addition, under the WC model, only the result from Flixster is shown among the three data sets, because of space limitation. The result from Gowalla and Digg shows a similar tendency to that from Flixster. To compare CELF++LR with CELF++, we use Wiki-Vote, which is a relatively small data set. To test scalability to the number of nodes, we use Flixster and Pokec.

Generating queries. For the experiments, we generate a syntactic query with three parameters as follows. First, we randomly select nodes as a part of total targets to be generated. Let p_1 denote a parameter for the proportion of the randomly selected targets. Next, for the remaining part, we select a node uniformly at random as a target and do breadth-first search starting from the node along in-bound edges. In the breadth-first search, for each visited node, we pick it as a target with probability p_2. We choose another node with probability p_3, and when we choose it, we apply uniform randomness, and do the same thing. We repeat this until the remaining part of the total targets is completed.

This method for generating syntactic queries can model various distributions of targets with the three parameters. p_1 and p_3 are used to control how much targets are connected, and p_2 is used to control how many targets exist in a connected subgraph. For example, when $p_1 = 1$ or $p_3 = 1$, all targets are randomly selected. When the values of p_1 and p_3 are low and the value of p_2 is high, targets are likely to be connected. In the experiments, we set $p_1 = 0.25$, $p_2 = 0.5$, and $p_3 = 0.02$. It means that 25 percent of targets are uniformly distributed and the others constitute several connected subgraphs. In the real world, targets can be uniformly distributed or constitute connected subgraphs depending on applications. We consider that our setting includes those aspects in the real world. Note that we performed experiments with other settings of p_1, p_2, and p_3, and their results have the same implications as those of the other settings in the next section.

TABLE 3
The Sensitivity Tests of the Parameters

$\alpha(\times 0.001)$	5	16.25	27.5	38.75	50
IS	1211.77	1202.65	1197.94	1195.98	1193.02
R(s)	0.1278	0.03385	0.0213	0.016	0.01375
β	0	1	1.5	2.0	2.5
IS	1211.75	1211.77	1211.75	1211.6	1211.68
R(s)	0.1646	0.1278	0.1139	0.105	0.09915
h	1	2	3	4	5
IS	1210.47	1211.91	1211.7	1211.68	1211.77
R(s)	0.107	0.12245	0.12635	0.1278	0.1278
$\delta(\times 0.0001)$	5	30	55	80	105
IS	1211.77	1211.12	1210.6	1208.32	1206.44
R(s)	0.1278	0.1107	0.099	0.06465	0.046

One might worry about how much this query generation method reflects the real world. Thus, we also extract real queries from the profile data of Pokec and perform experiments with the queries.

5.2 Experimental Results

Parameters. We conduct the sensitivity tests of α, β, h and δ, which are parameters for IMIP, under the WC model using Epinions. Given $\alpha = 0.005$, $\beta = 1.0$, $h = 5$, $\delta = 0.0005$, we vary each parameter and get a result. The results of the sensitivity tests are shown in Table 3 and we denote influence spread and running time as IS and R, respectively.

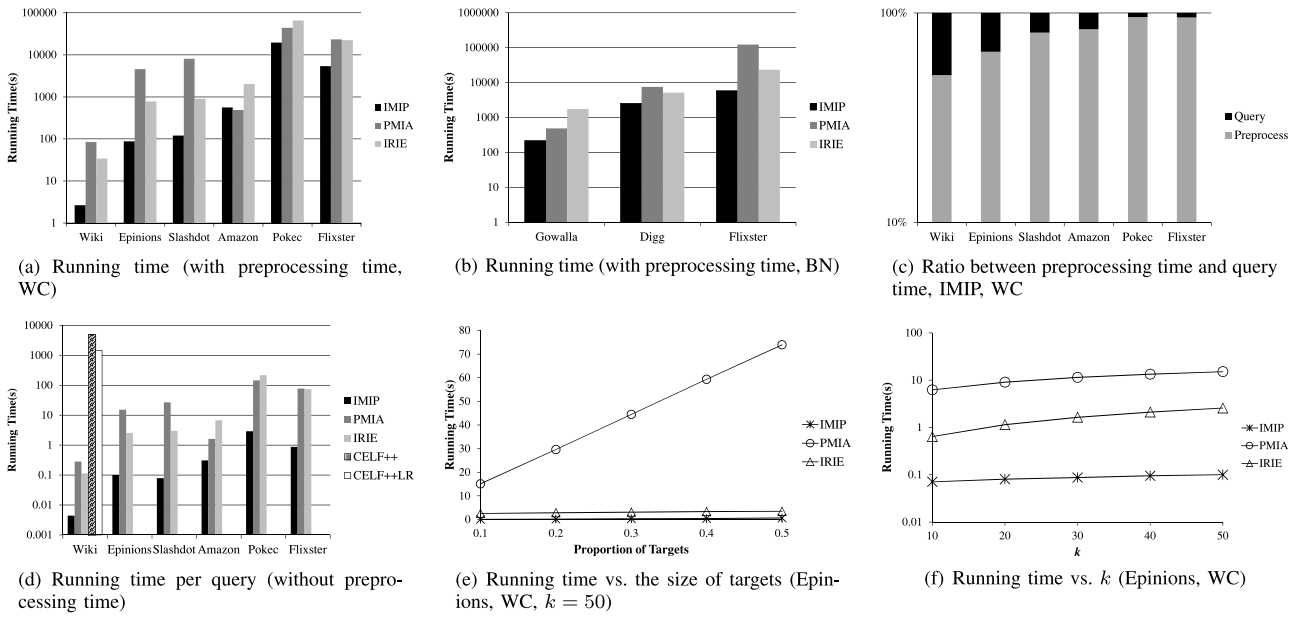
In the result of the sensitivity test for α, as α gets larger, the influence spread and the running time get smaller and shorter, respectively. The reason is that we do not consider any influence lower than α under the IMIP model. It causes more error for bigger α when computing the influence spread and makes the running time shorter.

Next, let us look at the result of the sensitivity test for β. In the result, even if β gets bigger, the influence spread is almost not changed and the running time gets shorter. This means that the technique for identifying local influencing regions in Algorithm 3 does not much miss influential nodes (i.e., true positives) on targets. In addition, a larger value of β lets the number of the result candidates of Algorithm 3 smaller and it affects the running time.

We also evaluate the proposed method with different maximum numbers of IMIPs. When $h = 1$, the influence spread is slightly low, but when $h > 1$, the influence spreads are high and similar to each other. Similarly, the running time gets longer, as h gets larger. Even if the influence spread when $h = 2$ is sufficiently high, we set $h = 5$ for stability in the rest of the experiments.

The last parameter of the proposed method is δ, which is used to avoid computing IMIPs whose influence is too small. In the result, when we set δ to a larger value, the influence spread becomes smaller and the running time becomes shorter. We set δ to a sufficiently smaller value than $1 - \sqrt[h]{1-\alpha}$ for the remaining experiments, since the influence spread is stable when $\delta < 1 - \sqrt[h]{1-\alpha}$.

For the rest of the experiments, our parameter settings are shown in the supplemental material, available online.



(a) Running time (with preprocessing time, WC)

(b) Running time (with preprocessing time, BN)

(c) Ratio between preprocessing time and query time, IMIP, WC

(d) Running time per query (without preprocessing time)

(e) Running time vs. the size of targets (Epinions, WC, $k = 50$)

(f) Running time vs. k (Epinions, WC)

Fig. 2. Running time analysis ($k = 50$).

We experimentally determine the value of each parameter considering efficiency and effectiveness.

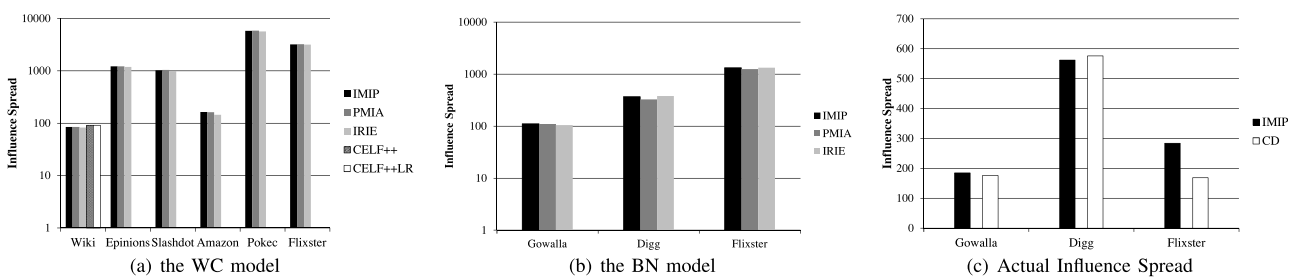
Running time. In the comparison of running time and influence spread, CELF++ and CELF++LR are evaluated only for Wiki-Vote under the WC model, because they are too slow to run in other environments.

To evaluate IMIP and its competitors with respect to efficiency, we generate 300 queries, each of which includes 10 percent of users, and run each method to process those queries. Preprocessing is done once before queries come and preprocessing is not necessary for each query processing. Figs. 2a and 2b illustrate the results from this experiment, and they include the preprocessing time. In the results, IMIP clearly outperforms its competitors in most cases. In Amazon, the running time of IMIP is longer than that of PMIA, because of the preprocessing time of IMIP. Fig. 2c shows the ratio between preprocessing time and query processing time in IMIP, and Fig. 2d shows the running time per query (not including preprocessing time). Even if IMIP is slower than PMIA in Amazon when we consider preprocessing time together in Fig. 2a, the running time per query of IMIP is much shorter than that of PMIA in Fig. 2d. As shown in Fig. 2c, most of the running time of IMIP is the preprocessing time. Thus, the performance gap between IMIP and each of the competitors with respect to running time per query is even bigger. IMIP is up to two orders of magnitude faster than PMIA and IRIE, and it is six orders of magnitude faster than CELF++. CELF++LR is 3.2 times faster than CELF++.

Effect of the size of targets. We test IMIP, PMIA and IRIE with different sizes of targets under the WC model. This experiment is related to the scalability of the number of targets. The result of this experiment is shown in Fig. 2f. In the result, IMIP is still faster than PMIA and IRIE when the half of all users are targets. The slope of PMIA is steeper than that of IMIP and IRIE. Since IRIE looks at all users per iteration to update the influence ranking regardless of the number of targets, the running time of IRIE is not affected by the number of targets as much as PMIA. The gentle slope of IMIP shows that IMIP is less affected by the number of targets than PMIA and IRIE.

Influence spread. For each data set, we evaluate the comparison methods in terms of influence spread with 50 syntactic queries, each of which includes 10 percent of users, as well as four queries extracted from real profile data. The result of evaluation with the syntactic queries is illustrated in Figs. 3a and 3b. In this result with various data sets, IMIP achieves influence spreads similar to those of PMIA and IRIE.

Based on the profile data of Pokec, we specify several sets of targets which can overlap and evaluate the proposed method with the sets. We use the WC model for this experiment. There are four sets of targets in this experiment: men, women, adults, and non-adults. As described in Table 4, IMIP is much faster than PMIA and IRIE over



(a) the WC model

(b) the BN model

(c) Actual Influence Spread

Fig. 3. Influence spread analysis ($k = 50$).

TABLE 4
Influence Spread and Running Time with Real Targets in Pokec, WC, $k = 50$

IS	Men	Women	Adults	Non-adults
IMIP	19678.1	22539.6	25411.9	17620.5
PMIA	20267.1	22931.7	25511.3	17782.9
IRIE	19580.9	22168.7	25418.3	17321.7
R(s)	Men	Women	Adults	Non-adults
IMIP	8.652	9.55	10.152	7.805
PMIA	473.69	539.636	613.081	466.488
IRIE	279.692	283.359	285.372	272.611

the sets of targets while achieving similar influence spread. Since a user is either male or female and either adult or non-adult, the set of men or the set of women has at least 50 percent of users as targets, and the same is true for the set of adults or the set of non-adults. Thus, this experiment also shows the scalability of the proposed method with respect to the number of real targets as well as the number of entire nodes.

Actual influence spread achieved. To demonstrate that the proposed method finds a seed set which can make large influence spread in reality, we compare the proposed method with the greedy method using the CD model in [19]. We set that the direct influence on an edge has the same value of the direct credit on the edge which is used in the CD model. In addition, to capture the actual influence spread achieved by a method, Goyal et al. take the seed set computed by the method and evaluate it with the influence spread predicted by their proposed CD model. The reason is that the CD model has least error in spread prediction compared to the other competitors in [19]. However, their evaluation can be unfair to the other competitors, because the result seed set of the greedy method using the CD model has of course bigger influence than the competitors. Therefore, in this work, we take another way to evaluate actual influence spread achieved by a method as follows.

Given a set of nodes A which we want to evaluate, let us consider an influenced node which is reachable from any influenced node in A with a path consisting of influenced nodes in the action logs of an item. We call it an actually influenced node of A and denote the number of times that a node u is an actually influenced node of A as $n(A, u)$ in the test set. Then, we consider that all nodes are targets and compute a seed set A with a method, which we want to evaluate, in the train set. In addition, we compute the actually influenced nodes of A for each item in the test set. Finally, we estimate the actual influence spread of A as the cardinality of the set of all actually influenced (and distinct) nodes u of A such that $n(A, u)$ is larger than a threshold. This threshold helps us to manipulate the level of confidence for this experiment. In this experiment, we set the threshold to 50 for Digg, 30 for Flixster, and 2 for Gowalla according to the sparseness of each data set. The train set consists of 60 percent of total action logs.

The result of comparing IMIP and CD with respect to the actual influence spread is shown in Fig. 3c. The differences between the actual influenced spreads achieved by IMIP and CD are not significant for the data sets. However, in [19], the seed sets, which the greedy algorithm using the IC model finds, have very poor performance with respect to the influence spread under the CD model. As Goyal et al. investigated, the EM algorithm sometimes determines an uninfluential node to be influential. That is why we use the direct credit of [19] for setting the direct influences instead of the EM method. Based on this experiment, we identify that IMIP finds a seed set which can make large influence spread in reality with an appropriate probability model for direct influences.

Overall, the proposed method, IMIP, is much more efficient than PMIA and IRIE while achieving similar accuracy. In addition, we identify that the proposed method can achieve an actual influence spread similar to that of CD.

6 Conclusions

In this paper, we formulate IMAX query processing to maximize the influence on specific users in social networks. Since IMAX query processing is NP-hard and calculating its objective function is #P-hard, we focus on how to approximate optimal seeds efficiently. To approximate the value of the objective function, we propose the IMIP model based on independence between paths. To process an IMAX query efficiently, extracting candidates for optimal seeds is proposed and the fast greedy-based approximation using the IMIP model.

We experimentally demonstrate that our identifying local influencing regions technique is effective and the proposed method is mostly at least an order of magnitude faster than PMIA and IRIE with similar accuracy In addition, the proposed method is mostly six orders of magnitude faster than CELF++ and the identifying local influencing regions technique makes CELF++ about 3.2 times faster while achieving high accuracy.

In the future, for IMAX query processing, we will consider more various distributions of targets such as users in the same community or the same university based on the static profiles of users. Next, we will apply IMAX query processing to the linear threshold model, and test whether the ideas in this paper are still applicable.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP)(No. NRF-2014R1A1A2002499).

References

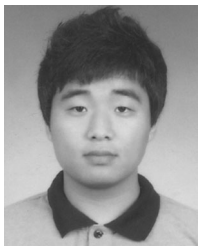
[1] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 137–146.

[2] F.-H. Li, C.-T. Li, and M.-K. Shan, "Labeled influence maximization in social networks for target marketing," in *Proc. IEEE 3rd Int. Conf. Privacy, Secur., Risk Trust, Int. Conf. Social Comput.*, 2011, pp. 560–563.

[3] W. Lu and L. Lakshmanan, "Profit maximization over social networks," in *Proc. IEEE 12th Int. Conf. Data Mining*, 2012, pp. 479–488.

[4] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 57–66.

[5] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 420–429.
[6] A. Goyal, W. Lu, and L. V. Lakshmanan, "CELF++: Optimizing the greedy algorithm for influence maximization in social networks," in *Proc. 20th Int. Conf. Companion World Wide Web*, 2011, pp. 47–48.
[7] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-k influential nodes in mobile social networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 1039–1048.
[8] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 199–208.
[9] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 1029–1038.
[10] Q. Jiang, G. Song, C. Gao, Y. Wang, W. Si, and K. Xie, "Simulated annealing based influence maximization in social networks," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 127–132.
[11] K. Jung, W. Heo, and W. Chen, "Irie: Scalable and robust influence maximization in social networks," in *Proc. IEEE 12th Int. Conf. Data Mining*, 2012, pp. 918–923.
[12] S. Bharathi, D. Kempe, and M. Salek, "Competitive influence maximization in social networks," in *Proc. 3rd Int. Conf. Internet Netw. Econ.*, 2007, pp. 306–311.
[13] T. Carnes, C. Nagarajan, S. M. Wild, and A. van Zuylen, "Maximizing influence in a competitive social network: A follower's perspective," in *Proc. Int. Conf. Electron. Commerce*, 2007, pp. 351–360.
[14] M. Irfan and L. Ortiz, "A game-theoretic approach to influence in networks," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 688–694.
[15] M. G. Rodriguez and B. Scholkopf, "Influence maximization in continuous time diffusion networks," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 313–320.
[16] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha, "Scalable influence estimation in continuous-time diffusion networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3147–3155.
[17] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "A data-based approach to social influence maximization," *Proc. VLDB* Endowment, vol. 5, no. 1, pp. 73–84, Sep. 2011.
[18] Y. Singer, "How to win friends and influence people, truthfully: Influence maximization mechanisms for social networks," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, 2012, pp. 733–742.
[19] A. Goyal, F. Bonchi, L. Lakshmanan, and S. Venkatasubramanian, "On minimizing budget and time in influence propagation over social networks," *Soc. Netw. Anal. Mining*, vol. 3, no. 2, pp. 179–192, 2013.
[20] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang, "Mining topic-level influence in heterogeneous networks," in *Proc. 19th ACM Int. Conf. Inform. Knowl. Manage.*, 2010, pp. 199–208.
[21] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: Finding topic-sensitive influential twitterers," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, 2010, pp. 261–270.
[22] N. Barbieri, F. Bonchi, and G. Manco, "Topic-aware social influence propagation models," in *Proc. IEEE 12th Int. Conf. Data Mining*, 2012, pp. 81–90.
[23] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the flickr social network," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 721–730.
[24] E. Sun, I. Rosenn, C. Marlow, and T. Lento, "Gesundheit! modeling contagion through facebook news feed," in *Proc. Int. AAAI Conf. Weblogs Soc. Media*, 2009, pp. 146–153.
[25] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 591–600.
[26] K. Lerman, and R. Ghosh, "Information contagion: An empirical study of the spread of news on digg and twitter social networks," in *Proc. Int. AAAI Conf. Weblogs Soc. Media*, 2010, pp. 90–97.



Jong-Ryul Lee received the BS degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST). He is currently working toward the PhD degree in computer science at KAIST. His research interests include data management, spatio-temporal data mining, and social network analysis.

Chin-Wan Chung received the BS degree in electrical engineering from Seoul National University, Korea, and the PhD degree in computer engineering from the University of Michigan, Ann Arbor. He is currently a professor in the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Korea. From 1983 to 1993, he was a senior research scientist and a staff research scientist in the Computer Science Department, General Motors Research Laboratories (GMR). While at GMR, he developed Dataplex, a heterogeneous distributed database management system integrating different types of databases. At KAIST, he developed a full-scale object-oriented spatial database management system called OMEGA, which supports ODMG standards. His current major project is about mobile social networks in Web 3.0. He was in the program committees of major international conferences including ACM SIGMOD, VLDB, IEEE ICDE, and WWW. He was an associate editor of *ACM Transactions on Internet Technology* and is an associate editor of *WWW Journal*. He was the general chair of WWW 2014. His current research interests include the Semantic Web, mobile web, social networks, spatio-temporal databases, and graph databases.