IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 1, 2019, accepted February 17, 2019, date of publication February 21, 2019, date of current version March 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900708

Location-Based Seeds Selection for Influence Blocking Maximization in Social Networks

WENLONG ZHU[1,2], WU YANG[1], SHICHANG XUAN[1], DAPENG MAN[1], WEI WANG[1], XIAOJIANG DU[3], (Senior Member, IEEE), AND MOHSEN GUIZANI[4], (Fellow, IEEE)

[1]Information Security Research Center, Harbin Engineering University, Harbin 165001, China
[2]College of Computer and Control Engineering, Qiqihar University, Qiqihar 165301, China
[3]Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
[4]College of Engineering, Qatar University, Doha 2713, Qatar

Corresponding author: Wu Yang (yangwu@hrbeu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572459 and Grant 61672180, in part by the Basic Scientific Research Project of Heilongjiang Education Department under Grant 135309469, and in part by the Teaching and Scientific Research Project of Qiqihar University under Grant 2016086 and Grant 201803.

ABSTRACT Influence blocking maximization (IBM) is a key problem for viral marketing in competitive social networks. Although the IBM problem has been extensively studied, existing works neglect the fact that the location information can play an important role in influence propagation. In this paper, we study the location-based seeds selection for IBM problem, which aims to find a positive seed set in a given query region to block the negative influence propagation in a given block region as much as possible. In order to overcome the low efficiency of the simulation-based greedy algorithm, we propose a heuristic algorithm IS-LSS and its improved version IS-LSS+, both of which are based on the maximum influence arborescence structure and Quadtree index, while IS-LSS+ further improves the efficiency of IS-LSS by using an upper bound method and Quadtree cell lists. The experimental results on real-world datasets demonstrate that our proposed algorithms are able to achieve matching blocking effect to the greedy algorithm as the increase in the number of positive seeds and often better than other heuristic algorithms, whereas they are four orders of magnitude faster than the greedy algorithm.

INDEX TERMS Influence blocking maximization, location-based, competitive social networks.

I. INTRODUCTION

Influence maximization (IM) is a fundamental social network problem which finds top-k optimal seeds to maximize the set of the influenced users in a given social network. It is first formulated as a discrete optimization problem by Kempe *et al.* [1]. In recent decades, the IM problem has been widely studied and applied in many applications, such as viral marketing [2], rumor control [3], and recommendation system [4]. However, Most of the studies only consider social influence for a single opinion or idea. In fact, it is often the case that different opinions and even opposite opinions are propagating in a social network simultaneously. A natural question is how to block the influence diffusion of competitors as much as possible when the two competitors are propagating influence of themselves simultaneously. We refer to this as the influence blocking maximization (IBM) problem. He *et al.* [5] formulate the IBM problem for the first time and some researchers also consider this topic from different aspects [3], [6]–[8].

Although the IBM problem meets many real-world applications such as product promotion, rumor control, etc., it neglects the fact that location information can play an important role in selecting the proper seeds. As an example which is shown in Fig. 1, the edges between nodes denote the direction of influence. Rectangle $\mathcal{R}^{Q1}$ or $\mathcal{R}^{Q2}$ which is represented by a dotted line denotes a geographical query region and rectangle $\mathcal{R}^{B1}$ or $\mathcal{R}^{B2}$ denotes a geographical block region. Suppose user N is the rumor starter and we want to select two seeds from a query region to start truth campaign for blocking rumor campaign in a block region. If there is no query region restriction, we will select P1 and P4 which prevent the most users from being influenced by the rumor N. However, if the query region is restricted to $\mathcal{R}^{Q1}$, P1 and P3 will be selected rather than P1 and P4. If the query region is restricted to $\mathcal{R}^{Q2}$ and the block region is restricted

The associate editor coordinating the review of this manuscript and approving it for publication was M. Shamim Hossain.

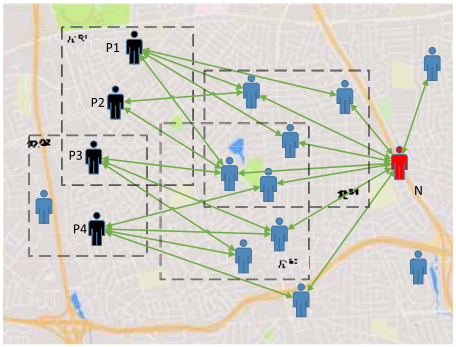


FIGURE 1. An example of location-based seeds selection for influence blocking maximization.

TABLE 1. Seeds selection under different region limitation in Fig. 1.

Region Limitation	Seeds Selection
No limitation	P1, P4
R^{Q1}	P1, P3
R^{Q2}	P3, P4
R^{B1}	P1, P2
R^{B2}	P3, P4
R^{Q1},R^{B1}	P1, P2
R^{Q1},R^{B2}	P1, P3
R^{Q2},R^{B1}	P3, P4
R^{Q2},R^{B2}	P3, P4

to $\mathcal{R}^{B2}$, P3 and P4 will be selected. We show the details of seeds selection under different region limitation in Table 1. As can be seen from the table, different regional queries leads to different results, so it is very important to consider location information for the optimal selection of seeds in competitive social networks.

In this paper, we focus on researching the problem of how to select the optimal seeds in different region information for IBM, or called location-based seeds selection for the influence blocking maximization (LSSIBM) problem. Given a location-aware social network, a competitive influence propagation model, a negative seed set S^N, a query region $\mathcal{R}^Q$ and a block region $\mathcal{R}^B$, the goal of LSSIBM is to identify the optimal positive seed set S^P of size at most k in $\mathcal{R}^Q$ such that the blocked negative influence spread in $\mathcal{R}^B$ caused by S^P is maximized. We prove that the LSSIBM problem is NP-hard and the influence function is monotone and submodular under the homogeneous competitive independent cascade model. Leveraging the monotonicity and submodularity of influence function, we can use the greedy algorithm [9] to solve the LSSIBM problem. However, the time complexity of the greedy algorithm is $O(knmr)$, where k is the number of positive seeds, n is the number of nodes in the query region, m is the edge set size in the social network, and r is the number of simulations.

To overcome the low efficiency of the Monte Carlo based greedy algorithm, we design a heuristic algorithm called IS-LSS which is based on influence set (IS) to solve the LSSIBM problem. In IS-LSS, we use the homogeneous competitive independent cascade model [10] to simulate influence propagation and run the algorithm in two steps. In the first step, we use a Quadtree index structure to store location information of nodes and a depth-first order search method to identify candidates located in a given query region. In the second step, we first use a max-heap to store candidates and their blocked negative influence. Then, we use the maximum influence arborescence (MIA) structure which is proposed by Chen *et al.* [11] to calculate the influence set of each candidate and run a dynamic programming algorithm [8] to calculate the blocked negative influence for all of the candidates. Further, we iteratively pop the candidate which has the maximum value in the max-heap as a positive seed.

The IS-LSS algorithm treats all nodes in the given query region equally and considers all nodes as candidates for seeds selection. In fact, many candidates are insignificant and will not be selected as seeds, so we want to avoid these unnecessary computations. To this end, we propose IS-LSS + to improve upon IS-LSS. IS-LSS + also utilizes a max-heap to store candidates and their blocked negative influence, but it employs an upper bound method and Quadtree cell lists to eliminate insignificant candidates.

Finally, we evaluate the performance of the proposed algorithms on three real-world datasets. The results show that both algorithms can achieve matching blocking effect to the greedy algorithm as the increase in the number of positive seeds, while run over four orders of magnitude faster than the greedy algorithm. Moreover, our algorithms are more effective than the other baseline algorithms. Meanwhile, compared with IS-LSS, IS-LSS + is more efficiency while has similar blocking effect.

Our main contributions in this paper are summarized as follows:

- We propose the LSSIBM problem for the first time and prove that it is NP-hard and the influence function is monotone and submodular.
- We devise IS-LSS and IS-LSS+ algorithms which are based on the MIA structure and Quadtree index to solve the LSSIBM problem.
- We demonstrate the effectiveness and efficiency of the proposed IS-LSS and IS-LSS+ algorithms by extensive experimental results on real-world datasets.

The remainder of this paper is organized as follows. Section II reviews related studies. Section III introduces some preliminaries. We formulate the LSSIBM problem in Section IV. Section V describes the IS-LSS algorithm and Section VI describes IS-LSS + algorithms. Experimental results are reported in Section VII. Finally, Section VIII summarizes the study and concludes the paper.

II. RELATED WORK

Many researchers have studied the network from different issues [12]–[14]. Influence maximization is one of the most

popular issue and has been investigated for many years. We review the literature related to this work from three areas, namely traditional influence maximization, location-aware influence maximization and influence blocking maximization.

A. TRADITIONAL INFLUENCE MAXIMIZATION

The IM problem is first studied as an algorithmic problem by Domingos and Richardson [15] and Richardson and Domingos [16]. Kempe *et al.* formulate it as a combinatorial optimization problem and present the formal definition for the first time. They prove that the IM problem is NP-hard under the classic independent cascade (IC) model and the linear threshold (LT) model, and propose a standard greedy algorithm with $(1-1/e-\varepsilon)$ approximation guarantee. Chen *et al.* [17] and Wang *et al.* [18] further prove that the IM problem is #P-hard under the LT model and the IC model separately. To improve the low efficiency of the Monte Carlo based greedy algorithm, Leskovec *et al.* [19] propose the cost-effective lazy forward (CELF) algorithm based on the diminishing return of marginal gain. Experimental results show that CELF achieves 700 times speedup compared to the greedy algorithm. Goyal *et al.* [20] propose the CELF ++ algorithm to further improve efficiency of CELF. Recently, Melo and Vignatti [21] propose a preselection algorithm to reduce simulation times of CELF under power law graphs.

Instead of using Monte Carlo simulations, many state-of-the-art heuristic algorithms [11], [17], [18], [22]–[25] are proposed to further improve the efficiency but sacrifice $(1 - 1/e - \varepsilon)$ approximation guarantee. Most of these algorithms are designed based on the maximum influence arborescence structure which is proposed by Chen *et al.* [11]. In this paper, we also use MIA to effectively approximate the blocked negative influence of nodes.

Another solution to IM is based on the reverse influence sampling (RIS) technology, which is proposed by Borgs *et al.* [26]. The main idea of RIS is not to estimate the influence from seed nodes, but to randomly sample nodes and run Monte Carlo simulations in opposite direction to search the nodes which can influence the sampled nodes. This motivates many studies to further improve the sampling technology [27]–[30] and memory consumption [31], [32].

B. LOCATION-AWARE INFLUENCE MAXIMIZATION

Recently, the location information plays an increasingly important role in social networks. Li *et al.* [33] are the first to consider the location information in the IM problem. The goal of their problem is to find a seed set to maximize the influence spread in the given query region. They use the PMIA [11] model to estimate influence propagation and further propose two efficient algorithms with $\varepsilon(1-1/e)$ approximation guarantee. Zhou *et al.* [34] devise a two-phase (TP) model and propose two heuristic algorithms to solve the location-aware IM problem under the Online to Offline (O2O) environment. Wang *et al.* [35] propose the distance-aware influence maximization (DAIM) model which combines two factors: influence spread and users' distance from the query region. Further, they devise an anchor point (AP) based algorithm to solve location-aware promotion problem. Li *et al.* [36] consider community-based seeds selection problem in location-aware social networks and propose a community-based seeds selection algorithm based on community detection and MIA. Su *et al.* [37] consider both topic and geographical preferences on IM problem. Zhu *et al.* [38] study the location-aware propagation probability problem by considering check-ins of users.

C. INFLUENCE BLOCKING MAXIMIZATION

There are also many research efforts on competitive influence maximization (CIM) problem [39]–[41], which consider the situation that some competitors propagate their influence in the social network simultaneously. The goal of CIM is to maximize the influence of one entity and the same time to block the influence of its competitors as much as possible. Different from the CIM problem, IBM aims to find a strategy for an entity to minimize the influence of its competitors or to maximize the blocking effect on its competitors.

The IBM problem is first formulated by He *et al.* [5]. They prove that the IBM problem is NP-hard and influence function is monotone and submodular under the CLT model. Further, they adopt the local directed acyclic graph (LDAG) structure [17] and propose an efficient heuristic algorithm for the IBM problem. Budak *et al.* [3] also consider the IBM problem under the competitive social networks, which they called the eventual influence limitation (EIL) problem. Wu and Pan [6] consider the IBM problem under two extended IC models and design two heuristic algorithms based on MIA. Lv *et al.* [42] devise a heuristic algorithm based on the community structure of the network for the IBM problem. Song *et al.* [7] consider IBM problem with time delays. Zhu *et al.* [8] propose two heuristic algorithms LIBM-H and LIBM-C to solve the location-aware IBM problem.

III. PRELIMINARIES

To clearly introduce the LSSIBM problem, we first introduce some preliminaries. Table 2 summarizes the frequently used notations throughout the paper.

A. PROPAGATION MODEL

A competitive social network can be modeled as a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. In the context of influence propagation, V can be viewed as the users of the social network and E can be viewed as the relationships of the users. A user u is a neighbor of a user v if and only if there is an edge $e_{u,v} \in E$. In addition, we assign two weights $pp_{u,v}$ and $pn_{u,v}$ to each edge $e_{u,v} \in E$ to model the positive and negative influence of the user u to the user v.

Although the IC model and the LT model are the most popular models in traditional IM problem, these models are not suitable for competitive influence propagation. Instead,

TABLE 2. Frequently used notations.

Notation	Meaning
$G = (V, E)$	a social network
u, v, w	a node in V
$\mathcal{R}^Q$	a query region for positive seeds selection
$\mathcal{R}^B$	a block region for influence blocking
$Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$	a query
S, S^N, S^P	seed set / negative seed set / positive seed set
$MIIA(u,\theta)$, $MIOA(u,\theta)$	maximum influence in/out arborescence of u, θ is a threshold
$I_r(u)$, $I_e(u)$	influencer / influencee set of u
$I_r(u,\mathcal{R})$, $I_e(u,\mathcal{R})$	region-constrained influencer/influencee set of u
$I_r(S,\mathcal{R})$, $I_e(S,\mathcal{R})$	region-constrained influencer/influencee sets of S
$DecInc(u,v)$	the blocked negative influence of u to v
$DecInc(u,\mathcal{R}^B)$, $DecInc(u)$	the blocked negative influence of u to nodes located in $\mathcal{R}^B$ in IS-LSS
$\mathcal{L}$, $\mathcal{L}_{c^0}$	cell index / not fully covered cell list
$\mathcal{L}_{c^1},...,\mathcal{L}_{c^r}$	fully covered cell lists
$U_{\max}$	the maximum blocked negative influence of visited candidates
$B_{\max}$	the upper bound of the blocked negative influence of unvisited candidates
$decInc(u,\mathcal{R})$	the blocked negative influence of candidate u to nodes located in $\mathcal{R}$ in IS-LSS+

the competitive independent cascade (CIC) model [10] and the competitive liner threshold (CLT) model [10] are well used for competitive social networks. In this paper, we use the homogeneous CIC [10] model which is an extension of the CIC model to simulate influence propagation under competitive social networks.

Given a competitive social network$G = (V, E)$, the initial negative seed set $S^N \subset V$ and positive seed set $S^P \subset V$, the CIC model works as follows. Let S_t^N and S_t^P be the set of nodes that are positively and negatively activated at step t respectively, with $t \geq 0$, $S_0^N = S^N$ and $S_0^P = S^P$. At step $t+1$, each node $u \in S_t^N$ has a single chance to negatively activate its inactive neighbor v with probability $pn_{u,v}$ and each node $u \in S_t^P$ has a single chance to positively activate its inactive neighbor v with probability $pp_{u,v}$. The process ends at a step t with $S_t^N = \emptyset$ and $S_t^P = \emptyset$. If in step t, v is activated by both positive propagation and negative propagation, then negative propagation dominates and v is negatively activated. The negative dominance rule matches the common sense that rumors are usually hard to fight with. We say the CIC model is homogeneous if each edge $e_{u,v} \in E, pn_{u,v} = pp_{u,v}$. Hereafter, when there is no ambiguity, we use $p_{u,v}$ to represent both positive and negative propagation probability of u to v.

B. THE MIA STRUCTURE

The MIA structure is first proposed by Chen *et al.* [11]. The basic idea of MIA is to use the maximum influence path (MIP) to estimate the influence from one node to another. For a path $P = < u = w_1, w_2, \ldots, w_m = v >$, the propagation probability of the path is defined as $pp(P) = \prod_{i=1}^{m-1} p_{w_i,w_{i+1}}$. For a node pair u and v, there are many paths from u to v. The maximum influence path from u to v is defined as $MIP_{u,v} = \arg\max pp(P)$. If we translate $p_{w_i,w_{i+1}}$ to a distance weight $-\log p_{w_i,w_{i+1}}$, then MIP is simply the shortest path from u to v, and can be computed by efficient algorithms such as Dijkstra algorithm. Further, Chen *et al.* [11] define maximum influence in-arborescence (MIIA) and maximum influence out-arborescence (MIOA). For a node $v \in V$, $MIIA(v,\theta)$ is the union of MIP from other nodes to v, $MIOA(v,\theta)$ is the union of MIP from v to other nodes and θ is an influence threshold to eliminate MIPs that have too small propagation probabilities. i.e.,

$$MIIA(v,\theta) = \bigcup_{u \in V, pp(MIP_{u,v}) \geq \theta} MIP_{u,v} \tag{1}$$

$$MIOA(v,\theta) = \bigcup_{u \in V, pp(MIP_{v,u}) \geq \theta} MIP_{v,u} \tag{2}$$

We can see that $MIIA(v,\theta)$ and $MIOA(v,\theta)$ give the local influence regions of v, and θ controls the size of these influence regions.

C. INFLUENCE BLOCKING MAXIMIZATION

He *et al.* [5] formulate the IBM problem for the first time. In this part, we first define the influence blocking set (IBS), then we define the blocked negative influence (BNI), finally, we define the IBM problem.

Definition 1 (Influence Blocking Set): Given a competitive social network$G = (V, E)$, a negative seed set S^N and a competitive propagation model, the influence blocking set of a positive seed set S^P, denoted as $IBS(S^P, S^N)$, is defined as the set of nodes that would be negatively activated if the positive seed set is empty, while are not negatively activated if the positive seed set is S^P.

Definition 2 (Blocked Negative Influence): Given a competitive social network $G = (V, E)$, a negative seed set S^N and a competitive propagation mode, the blocked negative influence of a positive seed set S^P, denoted as $\sigma(S^P, S^N)$, is defined as the expected size of $IBS(S^P, S^N)$. i.e.,

$$\sigma(S^P, S^N) = E(\left|IBS(S^P, S^N)\right|) \tag{3}$$

Definition 3 (Influence Blocking Maximization): Given a competitive social network $G = (V, E)$, a negative seed set S^N and a competitive propagation mode, the influence blocking maximization problem aims to find a positive seed set S^{P*} of size at most k to initiate positive propagation competing with negative propagation such that the blocked negative influence $\sigma(S^P, S^N)$ is maximized. i.e.,

$$S^{P*} = \arg\max_{S^P \subset V \setminus S^N, |S^P| \leq k} \sigma(S^P, S^N) \tag{4}$$

IV. LSSIBM PROBLEM AND THE GREEDY ALGORITHM

In this section, we first define region-constrained influence blocking set (RIBS) and region-constrained blocked negative influence (RBNI). Then, we define the LSSIBM problem. Finally, we prove that the LSSIBM problem is NP-hard and

the influence function has characteristics of monotonicity and submodularity under the homogeneous CIC model, and propose the greedy algorithm to solve the LSSIBM problem.

In a location-based competitive social network, each node $v \in V$ has a geographical location (x, y), where x is longitude and y is latitude. Given a query $Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$, $\mathcal{R}^Q$ is a query region for positive seeds selection, $\mathcal{R}^B$ is a block region for blocking the influence propagation of negative seeds and k is the size of positive seed set. We define the region-constrained influence blocking set as follows.

Definition 4 (Region-Constrained Influence Blocking Set): Given a competitive social network$G = (V, E)$, a negative seed set S^N, a competitive propagation model, a query region $\mathcal{R}^Q$ and a block region $\mathcal{R}^B$, the region-constrained influence blocking set of S^P, denoted as $RIBS(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)$, is defined as the set of nodes located in $\mathcal{R}^B$ that would be negatively activated if the positive seed set is empty, while are not negatively activated if the positive seed set is S^P, with the seeds of S^P located in $\mathcal{R}^Q$.

Definition 5 (Region-Constrained Blocked Negative Influence): Given a competitive social network $G = (V, E)$, a negative seed set S^N, a competitive propagation model, a query region $\mathcal{R}^Q$ and a block region $\mathcal{R}^B$, the region-constrained blocked negative influence of a positive seed set S^P, denoted as $\sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)$, is defined as the expected size of $RIBS(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)$. i.e.,

$$\sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B) = E(\left|RIBS(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)\right|) \quad (5)$$

Now we formally define the LSSIBM problem as follows.

Definition 6 (Location-Based Seeds Selection For Influence Blocking Maximization): Given a competitive social network $G = (V, E)$, a competitive propagation model, for a query $Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$, LSSIBM is the problem of finding a positive seed set S^{P*} of size at most k, with nodes in S^{P*} located in the query region $\mathcal{R}^Q$, such that the region-constrained blocked negative influence in the block region $\mathcal{R}^B$ is maximized. i.e.,

$$S^{P*} = \underset{S^P \subset \mathcal{R}^Q \setminus S^N, \mathcal{R}^Q \subset V, |S^P| \le k}{\arg\max} \sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B) \quad (6)$$

Example 1. Fig. 2 shows a location-based competitive social network with 27 nodes and 35 edges. The number on each directed edge $e_{u,v}$ is positive and negative influence propagation probability $p_{u,v}$. The value of $p_{u,v}$ is set to $1/d(v)$, where $d(v)$ is the in-degree of v. We randomly select 2 nodes as negative seeds, e.g., nodes 15 and 18. Given a query $Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$ with the dotted rectangle (e.g., $\mathcal{R}^Q = \mathcal{R}^{Q1}, \mathcal{R}^B = \mathcal{R}^{B1}$) and $k = 3$. The goal of LSSIBM is to find at most 3 positive seeds located in $\mathcal{R}^{Q1}$ to block the negative influence propagation of nodes 15 and 18 in the block region $\mathcal{R}^{B1}$ as much as possible.

Theorem 1: Under the homogeneous CIC model, the LSSIBM problem is NP-hard.

Proof: We prove the hardness of LSSIBM by considering a simple case that $S^P = \emptyset$ and both of $\mathcal{R}^Q$ and $\mathcal{R}^B$ cover all nodes in V. In this situation, the LSSIBM problem becomes a traditional IBM problem which is proved NP-hard by Wei *et al.* [10]. Therefore, the LSSIBM problem is NP-hard.

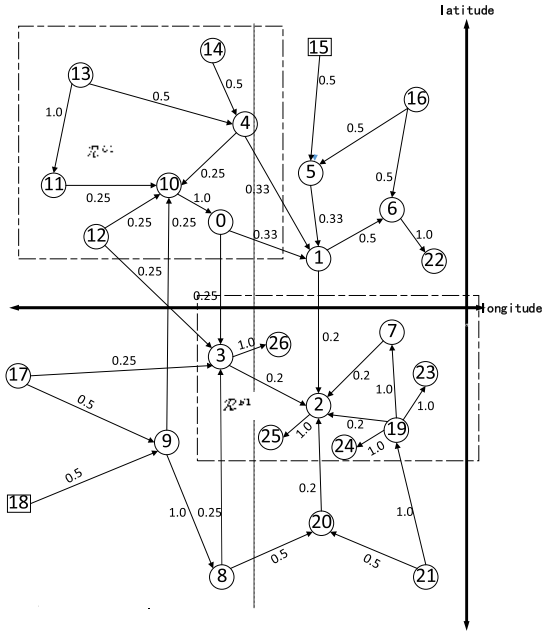


FIGURE 2. A running example.

To overcome the NP-hardness result of Theorem 1, we look for approximation algorithms. The Monte Carlo based greedy algorithm is well used in the IM problem. In order to use the greedy algorithm to solve the LSSIBM problem, the influence function $\sigma(\cdot)$ must satisfy the characteristics of monotonicity and submodularity. We say that an influence function $\sigma : 2^V \to \mathbb{R}$ is monotone if it satisfies $\sigma(S) \le \sigma(T)$ for any $S \subseteq T \subseteq V$. Moreover, we say that an influence function $\sigma : 2^V \to \mathbb{R}$ is submodular if it satisfies $\sigma(S \cup \{v\}) - \sigma(S) \ge \sigma(T \cup \{v\}) - \sigma(T)$ for any subset $S \subseteq T \subseteq V$ and any element $v \in V \setminus T$. Intuitively, submodularity means that $\sigma(\cdot)$ has a diminishing marginal return. On this basis, we have the following theorem.

Theorem 2: In the LSSIBM problem, the influence function $\sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)$ is monotone and submodular under the homogeneous CIC model.

Proof: It is easy to see that the influence function $\sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)$ is monotone. We prove the character of submodularity by a contradiction method. In [10], it is proved that the influence spread function $\sigma(S^P, S^N)$ is submodular under the homogeneous CIC model for the IBM problem. We can easily show that the influence function $\sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)$ of LSSIBM problem under the homogeneous CIC model is a reduction from the IBM problem by setting $\mathcal{R}^Q$ and $\mathcal{R}^B$ covers all of nodes in the given social network. If $\sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)$ is not submodular, then $\sigma(S^P, S^N)$ will not be submodular. So $\sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B)$ is submodular.

Based on the characteristics of monotonicity and submodularity, we propose the greedy algorithm. The pseudocode

of the greedy algorithm is shown as Algorithm 1. In each iteration, the greedy algorithm simulates influence propagation under the homogeneous CIC model and selects the seed which has the largest marginal value as the current seed until the positive seed set size is k. However, the greedy algorithm uses Monte Carlo simulations to select optimal seeds, even with the start-of-the-art optimization method such as CELF [19], the greedy algorithm takes more than twenty hours for the social network of four thousand nodes as we show in our experiments. We address this efficiency issue with our new algorithm IS-LSS in the next section.

Algorithm 1 Greedy

Input : $G = (V, E), S^N, Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$
Output : S^P
1: $S^P = \emptyset$;
2: **for** $i = 1$ to k **do**
3: $u = \arg\max_{v \in \mathcal{R}^Q \setminus (S^P \cup S^N)} (\sigma(S^P \cup \{v\}, S^N, \mathcal{R}^Q, \mathcal{R}^B) - \sigma(S^P, S^N, \mathcal{R}^Q, \mathcal{R}^B))$
4: $S^P = S^P \cup \{u\}$;
5: **end for**
6: **return** S^P;

V. IS-LSS ALGORITHM

In this section, we first introduce Quadtree and the influence set. Then, we propose the IS-LSS algorithm. Finally, we analyse the complexity of our algorithm.

A. THE QUADTREE STRUCTURE

Quadtree is a very useful index method for two-dimensional point data. In Quadtree, each Quadtree cell is formed as $< C_{id}, M, NSET, SE, SW, NE, NW >$. C_{id} is a Quadtree cell number. M is the minimum bounding rectangle (MBR) of location within this cell, e.g., $M = [(x, y), (xlim, ylim)]$, more precisely, x and y are the longitude and latitude of the bounding center of M, $xlim$ and $ylim$ are half the length and width of M. $NSET$ is the set of nodes located in M. SE,SW,NE,NW are pointers to sub cells. Moreover, Quadtree has a cell capacity c that limits the maximum number of nodes in each cell. When a cell contains more than c nodes after insertion, the cell splits into four cells based on the quadrants of its MBR.

In IS-LSS, to clearly identify the nodes in the query region and block region, we build a Quadtree according to the location information of nodes in the given social network and utilize the Quadtree to search the set of nodes in the given region. Given a region $\mathcal{R}$, we use $N(\mathcal{R})$ to represent the nodes located in $\mathcal{R}$. The searching process is described in Algorithm 2. Starting from the root cell, the algorithm recursively traverses all nodes overlapped with $\mathcal{R}$. When traversing a cell qt, if the MBR of it is not overlapped with $\mathcal{R}$, which means that nodes in qt and its sub cells are not located in $\mathcal{R}$, we ignore qt and do not check sub cells rooted at qt. Otherwise, we check each node in qt, if the node is located in $\mathcal{R}$, we add it to $N(\mathcal{R})$. We repeat this process recursively until we identify all cells in the Quadtree and take the final $N(\mathcal{R})$ as the return value. We provide an example of Quadtree structure and searching method in Example 2.

Algorithm 2 SearchNodeInRegion

Input : Quadtree cell qt, $\mathcal{R}$: *A region*.
Output : N($\mathcal{R}$)
1: **if** $\mathcal{R}$ is not overlapped with $qt.MBR$ **then**
2: return;
3: **end if**
4: **for** each $u \in qt.MBR$ **do**
5: **if** u is located in$\mathcal{R}$ **then**
6: add u to $N(\mathcal{R})$;
7: **end if**
8: **end for**
9: SearchNodeInRegion($qt.SE$, $\mathcal{R}$);
10: SearchNodeInRegion($qt.SW$, $\mathcal{R}$);
11: SearchNodeInRegion($qt.NE$, $\mathcal{R}$);
12: SearchNodeInRegion($qt.NW$, $\mathcal{R}$);
13: **return** $N(\mathcal{R})$;

Example 2. Fig. 3 shows the Quadtree structure of the location-based social network in Fig. 2 and we set cell capacity c = 3. The value in a circle represents C_{id}, e.g., C_0 means Quadtree cell 0, and the values in the rectangle represent nodes located in the corresponding MBR, e.g., nodes 24, 25, 26 are located in the cell C_{20}. We also list the location information of each cell under the corresponding rectangle. Next, given a region, we show how to search nodes in the Quadtree, e.g., $\mathcal{R}^{Q1} = [(-70, 45), (70, 40)]$. Starting from the root cell, we check whether the MBR of the root is overlapped with $\mathcal{R}^{Q1}$. Because the root is overlapped with $\mathcal{R}^{Q1}$, we check each node in the root and find node 0 is located in $\mathcal{R}^{Q1}$. Then, we check C_1 and ignore it and its sub cells for the reason that C_1 is not overlapped with $\mathcal{R}^{Q1}$. Then, we do the same operations on C_2 and C_3. We further check C_4, find nodes 4, 10 and 11, check C_6, find node 12, check C_7, C_8 and find nodes 14, 13. Finally, we get the set $N(\mathcal{R}^{Q1}) = \{0, 4, 10, 11, 12, 14, 13\}$ as the final result.

B. THE INFLUENCE SET

Social influence is reflected by the path of the connected nodes. Inspired by the MIA structure [11], we utilize the influence set in our algorithms. The influence set contains two parts, named the influencee set and the influencer set. The definitions are as follows.

Definition 7 (Influencee Set): The influencee set of a node u, denoted as $I_e(u)$, is the set of nodes influenced by u under the MIA structure. i.e.,

$$I_e(u) = \{v | v \in MIOA(u, \theta)\} \tag{7}$$

Definition 8 (Influencer Set): The influencer set of a node u, denoted as $I_r(u)$, is the set of nodes that can influence u under

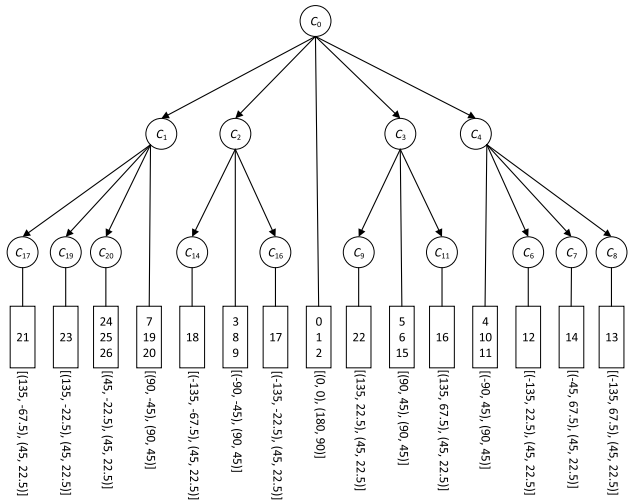


FIGURE 3. The Quadtree structure of Fig. 2.

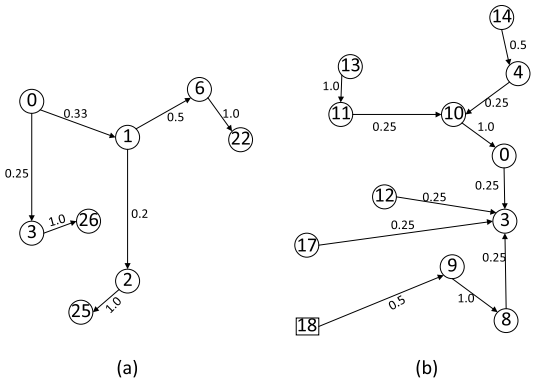


FIGURE 4. The MIA structure. (a) *MIOA* (0, 0.01). (b) *MIIA* (3, 0.01).

the MIA structure. i.e.,

$$I_r(u) = \{v | v \in MIIA(u, \theta)\} \tag{8}$$

The influencee set of u can be naturally extended to a set of users. Let S be a set of users, the influencee sets of S is defined as the union of the influencee set of each node in S, i.e., $I_e(S) = \cup_{u \in S} I_e(u)$. The concept of influencer sets of S can be defined similarly, i.e., $I_r(S) = \cup_{u \in S} I_r(u)$.

Further, we propose the region-constrained influencee set and region-constrained influencer set for LSSIBM. The formal definitions are as follows.

Definition 9 (Region-Constrained Influencee Set): Given a region $\mathcal{R} = [(x, y), (xlim, ylim)]$, the region-constrained influencee set, denoted as $I_e(u, \mathcal{R})$, is a subset of $I_e(u)$ consisting of the users located in the region $\mathcal{R}$. The concept of region-constrained influencee sets of a set of users can be defined similarly, i.e., $I_e(S, \mathcal{R}) = \cup_{u \in S} I_e(u, \mathcal{R})$.

Definition 10 (Region-Constrained Influencer Set): Given a region $\mathcal{R} = [(x, y), (xlim, ylim)]$, the region-constrained influencer set, denoted as $I_r(u, \mathcal{R})$, is a subset of $I_r(u)$ consisting of the users located in the region $\mathcal{R}$. The concept of region-constrained influencer sets of a set of users can be defined similarly, i.e., $I_r(S, \mathcal{R}) = \cup_{u \in S} I_r(u, \mathcal{R})$.

Example 3: Considering node 0 and node 3 in Fig. 2, we show how to compute $I_e(0, \mathcal{R}^{B1})$ and $I_r(3, \mathcal{R}^{Q1})$ with $\theta = 0.01$. For $I_e(0, \mathcal{R}^{B1})$, we should get $I_e(0)$ and take the set of nodes in $I_e(0)$ while located in $\mathcal{R}^{B1}$ as the final result. We achieve this by follows. First, we construct $MIOA(0, 0.01)$, as shown in Fig. 4 (a). After finishing the construction, we get $I_e(0) = \{1, 2, 3, 6, 22, 25, 26\}$. Then, we use Algorithm 2 to find the set of nodes located in $\mathcal{R}^{B1}$, denoted as $N(\mathcal{R}^{B1})$ and get $N(\mathcal{R}^{B1}) = \{2, 3, 7, 19, 24, 25, 26\}$. Finally, we check whether the nodes in $I_e(0)$ are in $N(\mathcal{R}^{B1})$ and get the final result set $I_e(0, \mathcal{R}^{B1}) = \{2, 3, 25, 26\}$. For $I_r(3, \mathcal{R}^{Q1})$, we first construct $MIIA(3, 0.01)$ and get $I_r(3)$. Then, we get $N(\mathcal{R}^{Q1})$ based on Example 2. Finally, we check whether the nodes in $I_r(3)$ are in $N(\mathcal{R}^{Q1})$ and get the final result set $I_r(3, \mathcal{R}^{Q1}) = \{0, 4, 10, 11, 12, 13, 14\}$.

C. INFLUENCE SET BASED ALGORITHM

We use the dynamic programming algorithm proposed in our previous work [8] to calculate the negative activation probability of a node u, denoted as $ap(u, S^P, S^N)$, under the negative seed set S^N and positive seed set S^P. The basic idea of the dynamic programming algorithm is to simulate the negative and positive influence propagation simultaneously under the MIA structure. One can see more details in [8, Algorithm 4].

For the LSSIBM problem, we want to select the seeds located in the query region $\mathcal{R}^Q$ to maximize the region-constrained blocked negative influence in the block region $\mathcal{R}^B$. To achieve this, for each node u located in $\mathcal{R}^Q$, we should first calculate the region-constrained blocked negative influence of u to nodes located in $\mathcal{R}^B$ after adding u to the positive seed set S^P. Then, we select the node with the maximum region-constrained blocked negative influence as the seed. In the absence of ambiguity, we use blocked negative influence (BNI) to represent region-constrained blocked negative influence (RBNI) for simplicity in the following work.

Accordingly, we use $DecInc(u, v)$ to represent the blocked negative influence of u to v after adding u to S^P. More precisely, $DecInc(u, v)$ is the negative influence reduction of v caused by adding u to the positive seed set S^P. i.e.,

$$DecInc(u, v) = ap(v, S^N, S^P) - ap(v, S^N, S^P \cup \{u\}) \tag{9}$$

Then, we use $DecInc(u, \mathcal{R}^B)$ to represent the blocked negative influence of u in $\mathcal{R}^B$. More precisely, $DecInc(u, \mathcal{R}^B)$ is the blocked negative influence of u to nodes located in $\mathcal{R}^B$ after adding u to the positive seed set S^P. i.e.,

$$DecInc(u, \mathcal{R}^B) = \sum\nolimits_{v \in \mathcal{R}^B \setminus (S^N \cup S^P)} DecInc(u, v) \tag{10}$$

Since the influence propagates through the MIA structure, only the nodes in the influencee set of u can cause negative influence reduction. i.e.,

$$DecInc(u, \mathcal{R}^B) = \sum\nolimits_{v \in I_e(u, \mathcal{R}^B) \setminus (S^N \cup S^P)} DecInc(u, v) \tag{11}$$

Example 4. Considering node 0 in Fig. 2, we show how to compute $DecInc(0, \mathcal{R}^{B1})$, with $S^P = \emptyset$. According to Example 3, we get $I_e(0, \mathcal{R}^{B1}) = \{2, 3, 25, 26\}$. Then, $DecInc(0, \mathcal{R}^{B1}) = DecInc(0, 2) + DecInc(0, 3) + DecInc(0, 25) + DecInc(0, 26)$. Based on [8, eq. (10) and Algorithm 4], we can get$DecInc(0, 2) = 0.0150$, $DecInc(0, 3) = 0.0312$, $DecInc(0, 25) = 0.0186$, $DecInc(0, 26) = 0.0546$. Finally, we can get $DecInc(0, \mathcal{R}^{B1}) = 0.1194$ as the final result.

Once we have computed $DecInc(u, \mathcal{R}^B)$ for each node u located in $\mathcal{R}^Q$. We can iteratively select the top-k seeds. For simplicity, we use $DecInc(u)$ to represent $DecInc(u, \mathcal{R}^B)$. The full pseudocode of IS-LSS is described in Algorithm 3.

The algorithm consists of two steps: offline step and online step. The goal of the offline step is to reduce the time-consuming of online step. In the offline step, for each node u, we preconstruct its MIA structure and get its influence set, then we precompute the initial negative activation probability of u based on [8, Algorithm 4 (lines 1–5)]. In the online step, given a query $Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$, we first use Algorithm 2 to get candidates located in $\mathcal{R}^Q$ (line 6). Then, we use a max-heap $\mathcal{H}$ to store the candidates and their blocked negative influence in $\mathcal{R}^B$. It is easy to see that the candidates are the nodes that can be selected as positive seeds. Next, we calculate the initial value of $DecInc(u, \mathcal{R}^B)$ for each candidate u based on (11 and 9) with $S^P = \emptyset$ and push $< u, DecInc(u) >$ into $\mathcal{H}$ (lines 7-13). In each iteration, we pop the top node in $\mathcal{H}$ as the next positive seed for the reason that the top node has the maximum blocked negative influence in all of the candidates (line 16). Once u is selected as a positive seed, only the candidates that have relations with u need to update their blocked negative influence. Based on the MIA structure, each node $w \in I_r(v, \mathcal{R}^Q)$ with $v \in I_e(u, \mathcal{R}^B)$ has old relation with u. So, we do the updating process by first minus the old blocked negative influence of w to each node $v \in I_e(u, \mathcal{R}^B)$, update $< w, DecInc(w) >$ in $\mathcal{H}$(lines 17-24). Then, we add the new seed u to S^P and update the MIA structures and influence sets of nodes which have relations with u(lines 25-28). Then, we add the new blocked negative influence of w to each node $v \in I_e(u, \mathcal{R}^B)$ and update $< w, DecInc(w) >$ in $\mathcal{H}$ (lines 27-34). Finally, we return S^P as the final positive seed set when all iterations are finished or there is no candidate in $\mathcal{H}$.

Algorithm 3 IS-LSS

Input : $G = (V, E), S^N, Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$
Output : S^P
//Offline step
1: **for** each node $u \in V$ **do**
2: construct $MIIA(u, \theta)$, $getI_r(u)$;
3: construct $MIOA(u, \theta)$, $getI_e(u)$;
4: $Inc(u) = ap(u, S^N, \emptyset)$, $DecInc(u) = 0$;
5: **end for**
//Online step, for each query$Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$
6: *get* $N(\mathcal{R}^Q)$ *base* on Algorithm 2, $S^P = \emptyset$, $\mathcal{H} = \emptyset$;
7: **for** each node $u \in N(\mathcal{R}^Q) \backslash S^N$ **do**
8: get $I_e(u, \mathcal{R}^B)$;
9: **for** each node $v \in I_e(u, \mathcal{R}^B) \backslash S^N$ **do**
10: $DecInc(u) += Inc(v) - ap(v, S^N, S^P \cup \{u\})$;
11: push $< u, DecInc(u) >$ into $\mathcal{H}$;
12: **end for**
13: **end for**
14: **for** $i = 1$ to k **do**
15: **if**$\mathcal{H}.top() = \emptyset$ **then return**S^P;
16: $u = \mathcal{H}.pop()$;
17: **for** each node $v \in I_e(u, \mathcal{R}^B) \backslash (S^P \cup S^N)$ **do**
18: get$I_r(v, \mathcal{R}^Q)$;
19: $Inc(v) = ap(v, S^N, S^P)$;
20: **for** each node $w \in I_r(v, \mathcal{R}^Q) \backslash (S^P \cup S^N)$ **do**
21: $DecInc(w) -= Inc(v) - ap(v, S^N, S^P \cup \{w\})$
22: update $< w, DecInc(w) >$ in $\mathcal{H}$;
23: **end for**
24: **end for**
25: $S^P = S^P \cup u$;
26: update $MIOA(u, \theta)$ and $I_e(u, \mathcal{R}^B)$;
27: **for** each node $v \in I_e(u, \mathcal{R}^B) \backslash (S^P \cup S^N)$ **do**
28: update $MIIA(v, \theta)$ and $I_r(v, \mathcal{R}^Q)$;
29: $Inc(v) = ap(v, S^N, S^P)$;
30: **for** each node $w \in I_r(v, \mathcal{R}^Q) \backslash (S^P \cup S^N)$ **do**
31: $DecInc(w) += Inc(v) - ap(v, S^N, S^P \cup \{w\})$
32: update $< w, DecInc(w) >$ in $\mathcal{H}$;
33: **end for**
34: **end for**
35: **end for**
36: **return**S^P;

D. COMPLEXITY ANALYSIS

Let $n = |V|, n_N = |S^N|, n_{\mathcal{Q}} = N(\mathcal{R}^Q), t_\theta$ be the maximum time of construction of$MIIA(v, \theta)$ or $MIOA(v, \theta)$, $m_\theta = \max_v\{|I_r(v)|, |I_e(v)|\}$. Since the construction of MIA has linear time complexity with its size, let $m_\theta = O(t_\theta)$ and the time complexity of $ap(u, S^P, S^N)$ is $O(m_\theta) = O(t_\theta)$[8]. We first analyze the time complexity of IS-LSS as follows.

In the offline step, for each node u, it takes $O(t_\theta)$ to construct the MIA structure and the influence set of u can be obtained during this time. Calculating $ap(u, S^P, S^N)$ takes $O(t_\theta)$. Overall, the offline step takes $O(nt_\theta)$. In the online step, Algorithm 2 takes $O(n)$. Initializing the max-heap takes $O(n_{\mathcal{Q}})$. So lines 7-13 take $O(n_{\mathcal{Q}}(m_\theta + m_\theta t_\theta))$ which can be simplified as $O(n_{\mathcal{Q}} t_\theta^2)$. For each iteration, updating the max-heap takes $O(\log n_{\mathcal{Q}})$. Lines 17-24 and lines 27-34 take $O(m_\theta(m_\theta + m_\theta + m_\theta(t_\theta + \log n_{\mathcal{Q}})))$ which can be simplified as $O(t_\theta^3 + t_\theta^2 \log n_{\mathcal{Q}})$. Line 26 takes $O(t_\theta)$. Thus, all iterations take $O(kt_\theta^3 + kt_\theta^2 \log n_{\mathcal{Q}})$. Thus, the online step takes $O(kt_\theta^3 + (n_{\mathcal{Q}} + k \log n_{\mathcal{Q}})t_\theta^2 + n)$. Overall, the total time complexity of IS-LSS is $O(kt_\theta^3 + (n_{\mathcal{Q}} + k \log n_{\mathcal{Q}})t_\theta^2 + nt_\theta)$.

For space complexity, if the capacity of each cell is c in a Quadtree, then the space complexity of the Quadtree is $O(n + 4n/c)$. A max-heap is used to store $DecInc(u)$ and it takes $O(n_{\mathcal{Q}})$. Constructing MIA structures takes $O(nm_\theta)$. The influence set can be obtained based on MIA structures, so it

does not take space. Overall, the total space complexity of IS-LSS is $O(nm_\theta)$.

VI. IS-LSS+ ALGORITHM

For each candidate located in the query region $\mathcal{R}^Q$, we need to calculate the blocked negative influence of the candidate in IS-LSS. In fact, most of the candidates are insignificant and should not be calculated during seeds selection. In this section, we propose IS-LSS+, an improved version of IS-LSS. We propose an upper bound method based on Quadtree cells and the MIA structure to prune insignificant candidates with small influence.

A. CELL INDEX AND CELL LISTS

Cell index $\mathcal{L}$. For each Quadtree cell C_i, we precompute the influencer set $I_r(v)$ for each node v in C_i, and take the union of the influencer set of nodes in C_i, denoted as $I_r(C_i)$. i.e.,

$$I_r(C_i) = \cup_{v \in C_i.NSET} I_r(v) \tag{12}$$

Then we use a list $\mathcal{L}_{C_i}$ to maintain each node u in $I_r(C_i)$ with its blocked negative influence to nodes located in C_i and sorted by descending order. i.e.,

$$\mathcal{L}_{C_i} = \{< u, decInc(u, C_i) > | u \in I_r(C_i)\} \tag{13}$$

More precisely, since the influence propagates in the MIA structure, $\mathcal{L}_{C_i}$ stores the nodes that have influence to nodes in C_i and their blocked negative influence. We use $decInc(u, C_i)$ to represent the blocked negative influence of node u to nodes in C_i. i.e.,

$$decInc(u, C_i) = \sum\nolimits_{v \in C_i.NSET \setminus (S^N \cup S^P)} DecInc(u, v) \tag{14}$$

Given a query $Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$, we first identify the set of Quadtree cells that are fully covered by $\mathcal{R}^B$, denoted as $C_F = \{C^1, C^2 \ldots, C^r\}$, and the set of nodes whose corresponding cells are not fully covered by $\mathcal{R}^B$, denoted as C^0. For each $C^j \in C_F$, we identify the corresponding cell list $\mathcal{L}_{C_i}$ from the cell index $\mathcal{L}$. It is worth mentioning that only the nodes located in $\mathcal{R}^Q$ are candidates, while many nodes in $\mathcal{L}_{C_i}$ are not located in $\mathcal{R}^Q$, so we combine these lists and only compute the blocked negative influence of candidates in C_F, denoted by $decInc(u, C_F)$. That is,

$$decInc(u, C_F) = \sum\nolimits_{1 \le j \le r} decInc(u, C^j) \tag{15}$$

Next, for C^0, we on the fly generate a sorted list of nodes based on their blocked negative influence in C^0, denoted as $\mathcal{L}_{C^0} = \{< u, decInc(u, C^0) >\}$. We get $\mathcal{L}_{C^0}$ by follows: First, we compute the region-constrained influencer sets of C^0, denoted as $I_r(C^0, \mathcal{R}^Q)$. Then, for each node u in $I_r(C^0, \mathcal{R}^Q)$, we get region-constrained influencee set of u in C^0, denoted as $I_e(u, C^0)$. Then, we compute the blocked negative influence of u based on $I_e(u, C^0)$ and (9). i.e.,

$$decInc(u, C^0) = \sum\nolimits_{v \in I_e(u, C^0) \setminus (S^N \cup S^P)} DecInc(u, v) \tag{16}$$

Finally, we add 2-tuples $< u, decInc(u, C^0) >$ to the cell list $\mathcal{L}_{C^0}$. It is easy to see that $\mathcal{L}_{C^0}$ stores the candidates that can block the negative influence to nodes located in C^0 and their blocked negative influence.

Now, we have a set of cell lists, denoted as $\mathcal{L}_{R_B} = \{\mathcal{L}_{C^0}, \mathcal{L}_{C^1}, ..., \mathcal{L}_{C^r}\}$. Using this set of cell lists, given a query $Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$, for each candidate u in $\mathcal{R}^Q$, we can compute its blocked negative influence in $\mathcal{R}^B$, denoted as $decInc(u, \mathcal{R}^B)$, as follows.

$$decInc(u, \mathcal{R}^B) = decInc(u, C^0) + decInc(u, C_F) \tag{17}$$

Based on (17) and corresponding (14-16, and 9), we can easily compute each candidate's blocked negative influence in $\mathcal{R}^B$.

Example 5: Also considering node 0 in Fig. 2, we show how to compute $decInc(0, \mathcal{R}^{B1})$ by using cell lists. Based on Fig. 3, we can get the fully covered set of cells $C_F = \{C_{19}, C_{20}\}$. For C_{19}, we get $decInc(0, C_{19}) = 0$ from $\mathcal{L}_{C_{19}}$. For C_{20}, we get $decInc(0, C_{20}) = 0.0732$ from $\mathcal{L}_{C_{20}}$. Thus, $decInc(0, C_F) = 0.0732$. Next, we can get the set of nodes in C^0 whose corresponding cells are not fully covered by $\mathcal{R}^{B1}$, $C^0 = \{2, 3, 7, 19\}$. Then, we can get $I_r(C^0, \mathcal{R}^Q) = \{0, 4, 10, 12, 13, 14\}$. Because node 0 is a candidate seed, based on (16) and $I_e(0, C^0) = \{2, 3\}$, we get $decInc(0, C^0) = decInc(0, 2) + decInc(0, 3) = 0.0150 + 0.0312 = 0.0462$. Finally, we can get $decInc(0, \mathcal{R}^{B1}) = decInc(0, C_F) + decInc(0, C^0) = 0.1194$ as the final result. We can see that this result is equal to the result in Example 4 in IS-LSS.

B. IMPROVED ALGORITHM

On this basis, we devise an improved algorithm IS-LSS+. The full pseudocode of IS-LSS+ is in Algorithm 4. In the offline step, comparing with IS-LSS, we precompute cell list $\mathcal{L}_{C_i}$ for each Quadtree cell C_i (lines 6). In the online step, given a query $Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$, we get corresponding cell lists $\mathcal{L}_{C^1}, \mathcal{L}_{C^2}, \ldots \mathcal{L}_{C^r}$ from $\mathcal{L}$ based on $N(\mathcal{R}^Q)$ (lines 7-8), and compute $\mathcal{L}_{C^0}$ for nodes whose corresponding cells are not fully covered by $\mathcal{R}^B$(line 9). In the iteration step, we also use a max-heap to identify positive seeds. Different from the IS-LSS algorithm, we do not insert all candidates into the max-heap, but add candidates as follows.

1) SELECTING THE FIRST SEED

As nodes in $\mathcal{L}_{C^0}, \mathcal{L}_{C^1}, \mathcal{L}_{C^2}, \ldots \mathcal{L}_{C^r}$ are sorted by their blocked negative influence in descending order, we first check the first node of each cell list. Notice that many of nodes in the fully covered cells are not candidates, we ignore them from the list until we get a candidate node (line 15). For the first candidate u_i in each cell $\mathcal{L}_{C^i}$, we compute $decInc(u_i, \mathcal{R}^B)$ based on (17) and add $< u_i, decInc(u_i, \mathcal{R}^B) >$ into the max-heap $\mathcal{H}$. Let $< u, U_{max} >$ be the first 2-tuples in $\mathcal{H}$, it is easy to see that U_{max} is the maximum blocked negative influence of visited candidates. Let $B_{\max} = \sum_{0 \le i \le r} decInc(u_i, C^i)$, where $decInc(u_i, C^i)$ is kept in the list $\mathcal{L}_{C^i}$. Obviously, B_{max} is an upper bound of the blocked negative influence of unvisited candidates. If $U_{max} > B_{max}$, the top node in $\mathcal{H}$ must be the seed and we terminate this iteration. Otherwise, we check the second candidate of each

Algorithm 4 IS-LSS+

Input : $G = (V, E)$, S^N, $Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$
Output : S^P
//Offline step
1: **for** each node $u \in V$ **do**
2: construct $MIIA(u, \theta)$, $getI_r(u)$;
3: construct $MIOA(u, \theta)$, $getI_e(u)$;
4: $Inc(u) = ap(u, S^N, \emptyset)$;
5: **end for**
6: Precompute cell list$\mathcal{L}_{C_i}$for each cellC_i;
//Online step, for each query$Q = (\mathcal{R}^Q, \mathcal{R}^B, k)$
7: *get* $N(\mathcal{R}^Q)$ *base*on Algorithm 2, $S^P = \emptyset$, $\mathcal{H} = \emptyset$;
8: identify $\mathcal{L}_{C^1}, \mathcal{L}_{C^2}, ..., \mathcal{L}_{C^r}$ from cell index $\mathcal{L}$
9: compute $\mathcal{L}_{C^0}$;
10: $U_{\max} = -1, B_{\max} = 0$;
11: **while** $U_{\max} < B_{\max}$ **do**
12: $B_{\max} = 0$;
13: **if** $\cup_{0 \le i \le r} \mathcal{L}_{C^i} = \emptyset$ **then break**;
14: **for** $i = 1$to r **do**
15: pop real candidate u_i from $\mathcal{L}_{C^i}$;
16: **if** *status* $(u_i) = 0$ **then**
17: compute $decInc(u_i, \mathcal{R}^B)$based on (17);
18: **else**
19: compute $decInc(u_i, \mathcal{R}^B)$based on (11);
20: **end if**
21: push $<u_i, decInc(u_i, \mathcal{R}^B)>$ into $\mathcal{H}$;
22: $B_{\max}$+=$decInc(u_i, \mathcal{R}^B)$
23: **end for**
24: get $<u, U_{\max}>$ from $\mathcal{H}$;
25: **if** $U_{\max} > B_{\max}$ **then**
26: **if** $State(u) = 0 || State(u) = 2$ **then**
27: set $State(w)$=1 for each $w \in Aff(u)$
28: pop $<u, U_{\max}>$ from $\mathcal{H}$;
29: $S^P = S^P \cup u$;
30: update $MIOA(u, \theta)$ and $I_e(u, \mathcal{R}^B)$;
31: *for*each node$v \in I_e(u, \mathcal{R}^B) \backslash (S^P \cup S^N)$*do*
32: update$MIIA(v, \theta)$and$I_r(v, \mathcal{R}^Q)$;
33: **end for**
34: **else if** $State(u) = 1$ **then**
35: compute $decInc(u, \mathcal{R}^B)$ based on (11);
36: push $< u_i, decInc(u, \mathcal{R}^B) >$ into $\mathcal{H}$;
37: $State(u) = 2$;
38: **end if**
39: **end if**
40: **end while**
41: **return** S^P;

cell $\mathcal{L}_{C^i}$, update B_{max} by using the sum of the blocked negative influence of the second candidate in each cell list, and insert all second candidates and their blocked negative influence into the max-heap.

2) SELECTING OTHER SEEDS

Selecting other seeds is different from selecting the first seed because once a candidate u is selected, some nodes which have co-influence with u will be affected and we need to update their blocked negative influence. To achieve this, we use $Aff(u)$ to represent the set of nodes which has co-influence with u, i.e.,$Aff(u) = \{w | w \in MIIA(v, \theta), v \in MIOA(u, \theta)\}$. Then, we use $State(u)$ to represent the relationship between u and the current seed. $State(u) = 0$ means that u has no co-influence with the current seed and we can use prepared cell lists to get the blocked negative influence of u. $State(u) = 1$ means that u is affected by the current seed and the original blocked negative influence of u is outdated, so we need to recompute the blocked negative influence of u. $State(u) = 2$ means that u is affected by the current seed and the blocked negative influence of u has been recomputed.

On this basis, we select other seeds iteratively as follows. When $U_{max} < B_{max}$, it means that we need to add new candidates into $\mathcal{H}$. We check the state of the next candidate in each cell list, if $State(u) = 0$, we use the same method as mentioned in selecting the first seed to update $\mathcal{H}$. Otherwise, we recalculate the blocked negative influence based on (11) and do the same other operations as selecting the first seed. When $U_{max} > B_{max}$, we get the top node u from $\mathcal{H}$. If $State(u) = 0$ or $State(u) = 2$, then u is indeed a seed. So we select u as the next seed, pop it from $\mathcal{H}$ and add it to S^P. If $State(u) = 1$, then we recalculate the blocked negative influence of u based on (11), update its value and set $State(u) = 2$. If $State(u) = 2$, it means the blocked negative influence of u is really the largest one in $\mathcal{H}$. We select u as the next seed, pop it from $\mathcal{H}$ and add it to S^P. Whenever we select a new seed u, we set $State(w) = 1$ and update the MIA structures and influence sets for all $w \in Aff(u)$. Finally, if we find k positive seeds or we have checked all of the nodes in cell lists $\mathcal{L}_{R_B}$, we stop iteration and return S^P as the final result.

C. COMPLEXITY ANALYSIS

For time complexity, let the maximum number of nodes in the cell list $\mathcal{L}_{C^0}$ be n_Q, n_1 be the maximum number of nodes in cell lists $\mathcal{L}_{C_i}$, i.e., $n_1 = \max(|\mathcal{L}_{C_i}|)$, n_c be the number of cell lists. In the offline step, lines 1-5 take $O(nt_\theta)$, building $\mathcal{L}_{C_i}$ takes $O(n_1 \log n_1)$. Thus, line 6 takes $O(n_c n_1 t_\theta + n_c n_1 \log n_1)$. Overall, the offline step takes $O((n + n_c n_1)t_\theta + n_c n_1 \log n_1)$. In the online step, line 7 takes $O(n)$, line 8 takes $O(\log n)$. For C^0, it takes $O(n)$ to identify nodes in C^0 and needs to compute $decInc(u, C^0)$ for at most n_Q nodes. Each node takes $O(t_\theta^2)$ to calculate the blocked negative influence. Building $\mathcal{L}_{C^0}$ takes $O(n_Q \log n_Q)$. Thus, line 9 takes $O(n + n_Q t_\theta^2 + n_Q \log n_Q)$. In the iteration step, there are at most n_{Q1} candidates in the max-heap, so initializing the max-heap takes $O(n_{Q1})$. Based on the IS-LSS + algorithm, we can see that n_{Q1} is much smaller than n_Q. Updating the max-heap takes $O(\log n_{Q1})$. Calculating U_{max} and B_{max} takes constant time. Thus, lines 14-23 take $O(t_\theta^2 + \log n_{Q1})$, lines 26-33 take $O(t_\theta^2)$, lines 35-37 take $O(t_\theta^2)$. Overall, all of the iterations take $O(k(t_\theta^2 + \log n_{Q1}))$. Thus, the online step takes $O(n + \log n + n_Q t_\theta^2 + n_Q \log n_Q + k(t_\theta^2 + \log n_{Q1}))$ which can be simplified as $O((k + n_Q)t_\theta^2 + n + n_Q \log n_Q)$. Overall, the total

TABLE 3. Statistics of the real-world networks.

Dataset	Nodes	Edges	AvgD	MaxD
Ego-Facebook	4039	88234	21.84	1045
Brightkite	58K	214K	3.68	1134
Gowalla	197K	950K	9.67	14730

AvgD = Average Degree, MaxD=Max Degree.

time complexity of IS-LSS + is $O((k+n_{\mathcal{Q}})t_\theta^2+(n+n_cn_1)t_\theta$ $+n_{\mathcal{Q}}\log n_{\mathcal{Q}}+n_cn_1\log n_1)$.

For space complexity, the construction of the Quadtree takes $O(n+4n/c)$. Constructing MIA structures takes $O(nm_\theta)$. For each $\mathcal{L}_{C_i}$, we need to insert n_1 nodes, which takes $O(n_1)$. For $\mathcal{L}_{C^0}$, we need to insert n_Q nodes, which takes $O(n_Q)$. Thus, the space complexity of cell lists is $O(n_Q+n_cn_1)$. Constructing max-heap takes $O(n_Q)$. Overall, the total space complexity of IS-LSS + is $O(nm_\theta+n_cn_1)$.

VII. EXPERIMENTS

In this section, we evaluate the effectiveness and efficiency of our proposed algorithms on three real-world datasets. All of the experiments are implemented in C++ and conducted on a Linux server with 1400MHz AMD Opteron(TM) Processor 6320 and 16G memory.

A. EXPERIMENTAL SETUP

1) DATASETS

We use the following three real-world datasets in our experiments.

- Ego-Facebook: Ego-Facebook is a small social graph derived from Facebook app. In Ego-Facebook, every node corresponds to a user and the edges between users correspond to friendly relations. This dataset is used for experiments compared with the greedy algorithm.
- Brightkite: Brightkite was once a location-based social networking service provider where users shared their locations by checking-in. The Brightkite dataset was collected using their public API.
- Gowalla: Gowalla is a social networking website where users share their locations by checking-in. The Gowalla dataset was collected using their public API.

All of the datasets can be downloaded from website http://snap.stanford.edu/data. The user location is the place the user most frequently check in. All of the datasets are undirected graphs and some basic statistics are given in Table 3.

2) EVALUATED ALGORITHMS AND METRICS

We compare the performance of the following algorithms.

- IS-LSS and IS-LSS+: Our algorithms with $\theta = 0.01$.
- Greedy: Algorithm 1 in our work with the lazy-forward optimization of [19], and we run 10000 simulations to obtain an accurate estimate.
- Random: A heuristic algorithm, simply selecting nodes in the query region at random as positive seeds.

TABLE 4. Index sizes and time on Gowalla (MB).

Methods	Quadtree	MIA	L	Time(s)
Greedy	11	\	\	3.92
Degree	11	\	\	3.92
DD	11	\	\	3.92
Random	11	\	\	3.92
Proximity	11	\	\	3.92
IS-LSS	11	332	\	39.42
IS-LSS+	11	332	27	261.38

- Degree: A heuristic algorithm, simply selecting top-k nodes located in the query region with largest degrees as positive seeds.
- Degree Discount (DD): A heuristic algorithm proposed in [43] with a propagation probability of 0.01, selecting top-k nodes located in the query region.
- Proximity: A heuristic algorithm proposed in [5], selecting the direct out-neighbors of negative seeds as positive seeds to block the negative influence.

We evaluate the efficiency in terms of running time and utilize region-constrained blocked negative influence to evaluate the effectiveness of these algorithms.

3) SETTINGS

We utilize the weighted cascade model [1] to generate influence propagation probability for each edge. In this model, we set $p_{u,v} = 1/d(v)$, where $d(v)$ is the in-degree of v.

We select 50 nodes with the largest degree as negative seeds. For the positive seed set size, we vary k from 0 to 200 and take 200 as the default value. For Quadtree, the cell's capacity is set to 200. For the region size, we range the number of nodes in the given region from 1000 to 5000 and take 3000 as the default value. One important issue is how to generate the query region and the block region with the given region number. To achieve this, we randomly generate a bounding center (x, y), and enlarge *xlim* and *ylim* step by step, once the number of nodes located in the generated region is larger than the expected number, we return the corresponding $[(x, y), (xlim,\ ylim)]$ as the result.

Since the homogeneous CIC model is a probabilistic model, when we evaluate the negatively activated nodes in a given query region for any negative and positive seed sets, we run the propagation simulation 1000 times and take their average as the result.

4) INDEX SIZES AND OFFLINE TIME

Due to limited space, we only report index sizes and offline time on Gowalla dataset. The details are shown in Table 4. All algorithms utilize Quadtree and the MIA structure. IS-LSS + uses cell index $\mathcal{L}$ to improve efficiency. To support efficient queries, we preconstruct these structures in the offline step and we only report the running time of the online step in our following experiments.

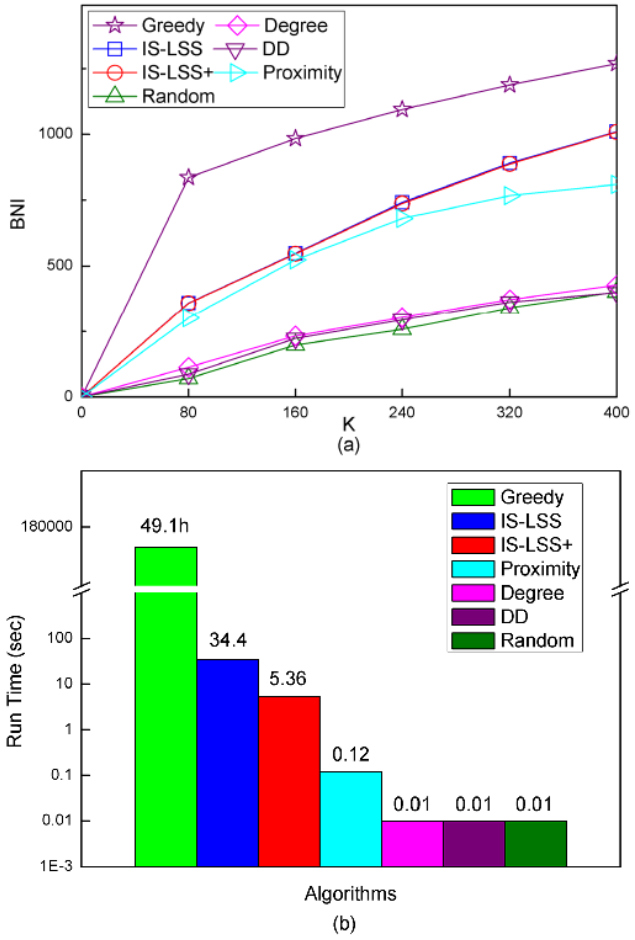


FIGURE 5. Results with the greedy algorithm on Ego-Facebook network.

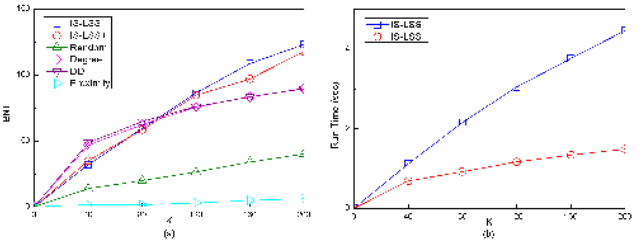


FIGURE 6. Varying*k* on Brightkite network.

B. RESULTS WITH THE GREEDY ALGORITHM

Since the greedy algorithm runs slowly in large networks, we run experiments on Ego-Facebook dataset to include the comparison with the greedy algorithm. For the reason that nodes in Ego-Facebook have no location information, we use the location of nodes in Brightkite as the location of nodes in Ego-Facebook. The query region and the block region are set to contain all nodes in the network. We select 50 nodes with highest degrees as negative seeds and vary positive seeds from 0 to 400 to block the negative influence. The results are shown in Fig. 5.

From Fig. 5 (a), we see that the blocked negative influence increases gradually with the increase in the number of positive seeds. Proximity performs better than Degree, DD and Random. Among all of the algorithms, our algorithms have the best performance in effectiveness except the theoretically optimal greedy algorithm. When the size of the positive seed set is small, the blocking effect of our algorithms is slightly worse than that of the greedy algorithm. However, with the increase in the size of the positive seed set, the blocking effect of our algorithms is more and more close to that of the greedy algorithm, e.g., our algorithms achieve 45% blocking effect of the greedy algorithm when the number of positive seeds is 80 and up to more than 80% blocking effect of the greedy algorithm when the number of positive seeds is 400. Although the greedy algorithm has the best blocking effect in all algorithms, from Fig. 5 (b), we see that the greedy algorithm takes more than 49 hours to select 400 positive seeds. In contrast, IS-LSS takes 34.4 seconds and IS-LSS + only takes 5.36 seconds for selecting 400 positive seeds. That is to say, our algorithms are more than four orders of magnitude faster than the greedy algorithm.

C. RESULTS WITH THE QUERY REGION EQUALS TO THE BLOCK REGION

In this section, to evaluate the performance of our algorithms, we first let $\mathcal{R}^B = \mathcal{R}^Q = \mathcal{R}$, which means selecting the optimal positive seeds located in $\mathcal{R}$ to block the negative influence propagation in $\mathcal{R}$. This is a typical application for rumor control and viral marketing. Then, we generate the query region and block region based on the method mentioned above and set bounding centers to (1.0, 20.0) and (26.0, 43.0) for Gowalla and Brightkite datasets respectively. Since the greedy algorithm cannot return results within 48 hours on Brightkite dataset, it is not included in the following experiments.

1) VARYING *K*

We evaluate different methods by varying positive seeds from 0 to 200. Fig. 6 shows the results on Brightkite dataset and Fig. 7 shows the results on Gowalla dataset. From Fig. 6 (a) and Fig. 7 (a), we have the following observations. First, with the increase of positive seeds, the blocking effect of all algorithms performs better and better except the Proximity. This is because Proximity selects the direct out-neighbors of negative seeds as positive seeds, when these seeds are far from the block region, they may have no blocking effect at all. Second, Degree and DD have similar blocking effect and perform better than Random because they both select nodes with the largest degrees as positive seeds in the query region. Third, we see that when the positive seed set size is smaller than 80, the blocking effect of our algorithms is smaller than that of the degree-based algorithms. This is because that our algorithms use local-based MIA structures to select optimal seeds, while Degree and DD use global-based degrees to select optimal seeds. When the size of the positive seed set is small, the seeds selected by global-based Degree

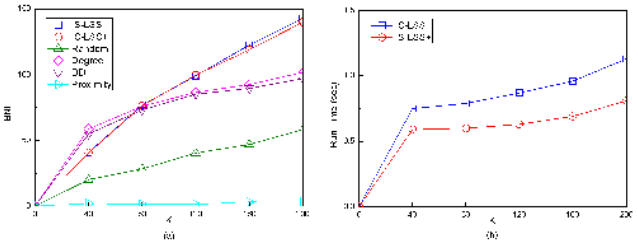


FIGURE 7. Varying*k* on Gowalla network.

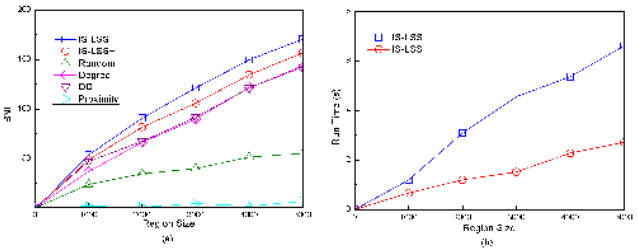


FIGURE 8. Varying region size on Brightkite network.

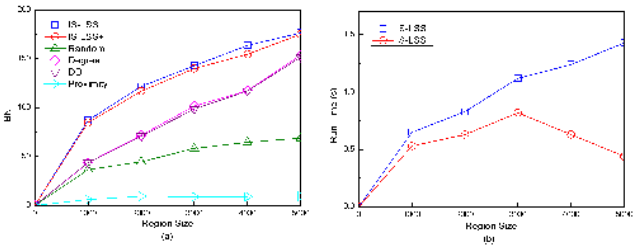


FIGURE 9. Varying region size on Gowalla network.

and DD may have better blocking effect than that of the seeds selected by our local-based algorithms. However, with the increase in the number of positive seeds, the blocking effect of our algorithms performs better and better than Degree and DD.

Fig. 6 (b) and Fig. 7 (b), we see that with the increase of positive seed set size, the running time of our algorithms increases on both datasets, because the more seeds there are, the more time to calculate the blocked negative influence. However, IS-LSS+ significantly improves the efficiency of IS-LSS, e.g., IS-LSS+ only takes 35% and 71% running time of IS-LSS to select 200 positive seeds on Brightkite and Gowalla datasets respectively. This is because IS-LSS+ uses the upper bound method and cell lists to eliminate many insignificant candidates with small blocked negative influence.

2) VARYING REGION SIZE

We evaluate different methods by varying the region size from 1000 to 500 to find 200 positive seeds to block the negative influence. The results are shown in Fig. 8 and Fig. 9. We observe that IS-LSS gives the best performance in

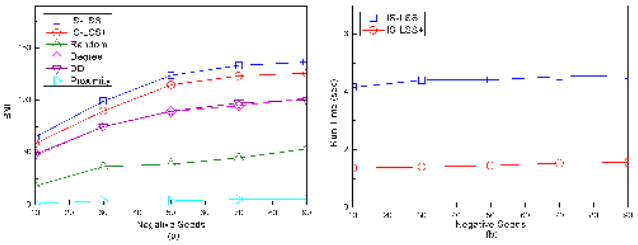


FIGURE 10. Varying negative seeds on Brightkite network.

effectiveness in all datasets. IS-LSS+ performs slightly poorer than IS-LSS in effectiveness because it utilizes the upper bound method to eliminate many candidates with small blocked negative influence. This method may cause some real candidates not be selected as positive seeds. However, IS-LSS+ still outperforms the closest Degree 25% and 10% in Gowalla and Brightkite datasets respectively. Random performs poorly because it does not consider the structure of the network. Proximity still has the worst performance for the reason mentioned above.

For efficiency, we observe that the running time of IS-LSS increases with the increase in region size. This is because that IS-LSS takes all of the nodes in the query region as candidates and recomputes the blocked negative influence for all candidates that have co-influence with the current seed. In contrast, IS-LSS+ is more robust to the region size, e.g., when the region expanded from 3000 to 5000 in Gowalla network, the running time of IS-LSS+ reduces from 0.82 to 0.44. The reason is that IS-LSS+ is based on cell lists and utilizes the upper bound method to select optimal seeds. With the expansion of the region, new candidates are pushed into cell lists and the computation cost of these new candidates is less than that of the old candidates.

3) VARYING NEGATIVE SEEDS

We conduct experiments by varying the number of negative seeds from 10 to 90 to find 200 positive seeds to block the negative influence. Experiment results are shown in Fig. 10 and Fig. 11. We have the following observations. First, our algorithms have the best blocking effect in all of the algorithms. ISS-LSS performs better than IS-LSS+ in terms of effectiveness, while it performs worse than IS-LSS+ in terms of efficiency for the reason mentioned above. Second, with the increase in the number of negative seeds, the increase in the blocked negative influence becomes smaller and smaller. The reason is that more negative seeds have more paths to propagate negative influence, once we block the influence propagation of negative seeds from one path, the negative seeds would choose another path to forward influence again. Last, the increase in the number of negative seeds has little effect on the running time of our algorithms. This is because that the query region and the block region are unchanged, and we still calculate the blocked influence in the original region.

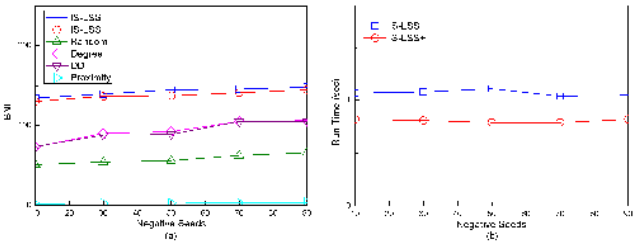


FIGURE 11. Varying negative seeds on Gowalla network.

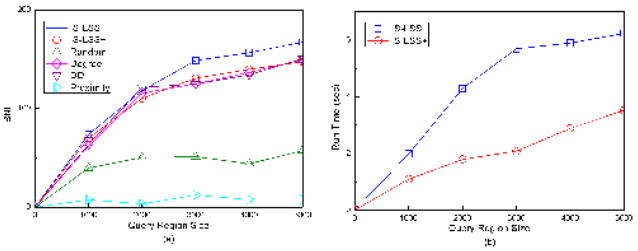


FIGURE 12. Varying query region size on Brightkite network.

D. RESULTS WITH THE QUERY REGION NOT EQUALS TO THE BLOCK REGION

In this section, we test the efficiency and effectiveness of our algorithms when $\mathcal{R}^B \neq \mathcal{R}^Q$. A natural method is to randomly generate a query region and a block query. However, we find this method is useless in practice. This is because that nodes in randomly generated regions have little connections, so there is almost no blocking effects. To solve this, we set the two regions have the same bounding center and one region is covered by the other. We use the same bounding centers as previous experiments.

1) VARYING QUERY REGION SIZE

In this set of experiments, we evaluate different methods by varying the size of the query region from 1000 to 5000 to find 200 positive seeds to block the negative influence. The size of the block region is set to 5000. Fig. 12 and Fig. 13 show the results on Brightkite and Gowalla datasets respectively.

The observations are as follows. First, different query regions can cause different positive seeds selection results. With the increase in the region size, we can select more influential nodes to block the negative influence. Therefore, the blocked negative influence of our algorithms increases. Second, again, IS-LSS has the best performance in all datasets. Degree and DD algorithms perform well because as the size of query region increases, they also choose more influential nodes with largest degrees. Radom does not consider the effect of different regions on seeds selection, when the query region size increases, the blocking effect of Random may decrease, e.g., when the query region expands from 4000 to 5000, the blocked negative influence of Random decreases from 51.2 to 43.9 in Fig. 12 (a). The main reason is that Random may choose new seeds in the enlarged region whose blocking effect is worse than that of the old seeds.

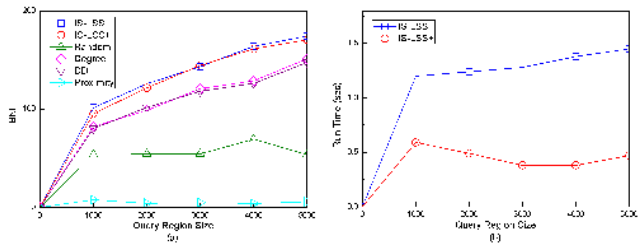


FIGURE 13. Varying query region size on Gowalla network.

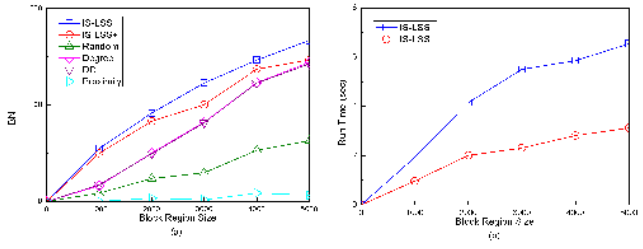


FIGURE 14. Varying block region size on Brightkite network.

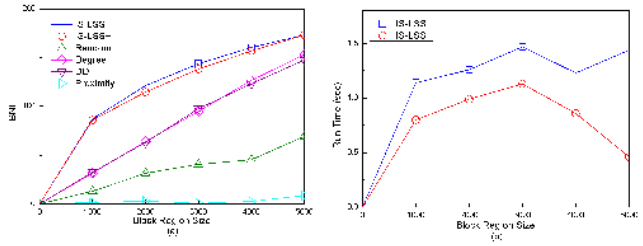


FIGURE 15. Varying block region size on Gowalla network.

Third, the performance of IS-LSS+ is still better than that of IS-LSS in terms of efficiency, and it is more robust to the region size for the reason mentioned above.

2) VARYING BLOCK REGION SIZE

In this set of experiments, to valuate different methods, we vary the size of the block region from 1000 to 5000 to find 200 positive seeds to block the negative influence. We set the size of the query region to 5000. Fig. 14 shows the result on Brightkite dataset and Fig. 15 shows the result on Gowalla dataset respectively.

From the figures, we observe that our algorithms also perform better than other algorithms. We also see that with the increase in the block region size, the blocking effect of degree-based algorithms is getting closer and closer to our algorithms. The reason is that our algorithms use the local-based MIA structure to compute blocked negative influence and select the best seeds, these locally optimal seeds may not be the globally optimal seeds as the size of the region becomes larger and larger. Another interesting observation is in Fig. 15 (b), we find that the running time of IS-LSS does not always increase, e.g., the running time of IS-LSS decreases from 1.47 to 1.23 when the size of the block region increases from 3000 to 4000. Our interpretation is that

when the size of the block region increases, we can find new candidates which have relations with the nodes located in the expanded block region, and the computation cost of these new candidates is less than that of the old candidates.

VIII. CONCLUSION

In this paper, we study location-based seeds selection for influence blocking maximization under competitive social networks. We formally define the LSSIBM problem and prove that the LSSIBM problem is NP-hard and the influence function is monotone and submodular. To improve the low efficiency of the greedy algorithm, we formally define the influence set and region-constrained influence set based on the MIA structure. Then, we propose the IS-LSS algorithm. The algorithm utilizes Quadtree to store location information of nodes, and computes the blocked negative influence of candidates based on the region-constrained influence set and a dynamic programming method. In IS-LSS, many insignificant candidates are computed when updating the blocked negative influence of candidates which have co-influence with the current seed. To improve the efficiency of IS-LSS, we devise IS-LSS + to eliminate insignificant candidates. IS-LSS + estimates the upper bound of the influence of nodes by using Quadtree cell lists and only computes the blocked negative influence of candidates which are added to the max-heap by the upper bound method. Experimental results show that our algorithms are four orders of magnitude faster than the greedy algorithm and often have a better blocking effect than many other baseline algorithms.

REFERENCES

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

[2] H. T. Nguyen, M. T. Thai, and T. N. Dinh, "A billion-scale approximation algorithm for maximizing benefit in viral marketing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2419–2429, Aug. 2017.

[3] C. Budak, D. Agrawal, and A. El Abbadi, "Limiting the spread of misinformation in social networks," presented at the 20th Int. Conf. World Wide Web, Hyderabad, India, 2011.

[4] B. Guler, K. Tutuncuoglu, and A. Yener, "Maximizing recommender's influence in a social network: An information theoretic perspective," in *Proc. IEEE Inf. Theory Workshop-Fall (Itw)*, Oct. 2015, pp. 262–266.

[5] X. He, G. Song, W. Chen, and Q. Jiang, "Influence blocking maximization in social networks under the competitive linear threshold model," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 463–474.

[6] P. Wu and L. Pan, "Scalable influence blocking maximization in social networks under competitive independent cascade models," *Comput. Netw.*, vol. 123, pp. 38–50, Aug. 2017.

[7] C. Song, W. Hsu, and M. L. Lee, "Temporal influence blocking: Minimizing the effect of misinformation in social networks," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 847–858.

[8] W. Zhu, W. Yang, S. Xuan, D. Man, W. Wang, and X. Du, "Location-aware influence blocking maximization in social networks," *IEEE Access*, vol. 6, pp. 61462–61477, 2018.

[9] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Math. Programm.*, vol. 14, no. 1, pp. 265–294, Dec. 1978.

[10] L. V. S. Lakshmanan, C. Castillo, and W. Chen, *Information and Influence Propagation in Social Networks*. San Rafael, CA, USA: Morgan & Claypool, 2013, p. 177.

[11] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," presented at the 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Washington, DC, USA, 2010.

[12] Y. Xiao, X. Du, J. Zhang, and S. Guizani, "Internet protocol television (IPTV): The killer application for the next generation Internet," *IEEE Commun. Mag.*, vol. 45, no. 11, pp. 126–134, Nov. 2007.

[13] X. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "A routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 3, pp. 1223–1229, Mar. 2009.

[14] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Netw.*, vol. 5, no. 1, pp. 24–34, Jan. 2007.

[15] P. Domingos and M. Richardson, "Mining the network value of customers," presented at the 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Francisco, CA, USA, 2001.

[16] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," presented at the 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Edmonton, AB, Canada, 2002.

[17] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," presented at the IEEE Int. Conf. Data Mining, Dec. 2010.

[18] C. Wang, W. Chen, and Y. Wang, "Scalable influence maximization for independent cascade model in large-scale social networks," *Data Mining Knowl. Discovery*, vol. 25, no. 3, pp. 545–576, Nov. 2012.

[19] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," presented at the 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Jose, CA, USA, 2007.

[20] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "CELF++: Optimizing the greedy algorithm for influence maximization in social networks," presented at the 20th Int. Conf. Companion World Wide Web, Hyderabad, India, 2011.

[21] R. S. Melo and A. L. Vignatti, "A preselection algorithm for the influence maximization problem in power law graphs," presented at the 33rd Annu. ACM Symp. Appl. Comput., Pau, France, 2018.

[22] K. Jung, W. Heo, and W. Chen, "IRIE: Scalable and robust influence maximization in social networks," in *Proc. 12th IEEE Int. Conf. Data Mining*, Dec. 2012, pp. 918–923.

[23] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "SIMPATH: An efficient algorithm for influence maximization under the linear threshold model," in *Proc. IEEE 11th Int. Conf. Data Mining*, Dec. 2011, pp. 211–220.

[24] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng, "IMRank: Influence maximization via finding self-consistent ranking," presented at the 37th Int. ACM SIGIR Conf. Res., Develop. Inf. Retr., Gold Coast, QLD, Australia, 2014.

[25] S. Galhotra, A. Arora, and S. Roy, "Holistic influence maximization: Combining scalability and efficiency with opinion-aware models," presented at the Int. Conf. Manage. Data, San Francisco, CA, USA, 2016.

[26] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," presented at the 25th Annu. ACM-SIAM Symp. Discrete Algorithms, Portland, OR, USA, 2014.

[27] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," presented at the ACM SIGMOD Int. Conf. Manage. Data, Snowbird, UH, USA, 2014.

[28] G. Tong, W. Wu, S. Tang, and D. Du, "Adaptive influence maximization in dynamic social networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 112–125, Feb. 2017.

[29] H. T. Nguyen, M. T. Thai, and T. N. Dinh, "Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks," presented at the Int. Conf. Manage. Data, San Francisco, CA, USA, 2016.

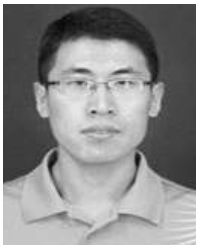
[30] H. T. Nguyen, T. P. Nguyen, N. Phan, and T. N. Dinh, "Importance sketching of influence dynamics in billion-scale networks," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2017, pp. 337–346.

[31] N. Ohsaka, T. Sonobe, S. Fujita, and K.-I. Kawarabayashi, "Coarsening massive influence networks for scalable diffusion analysis," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 635–650.

[32] D. Popova, N. Ohsaka, K.-I. Kawarabayashi, and A. Thomo, "NoSingles: A space-efficient algorithm for influence maximization," presented at the 30th Int. Conf. Sci. Stat. Database Manage., Bozen-Bolzano, Italy, 2018.

[33] G. Li, S. Chen, J. Feng, K.-L. Tan, and W.-S. Li, "Efficient location-aware influence maximization," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Snowbird, UT, USA, 2014, pp. 87–98.

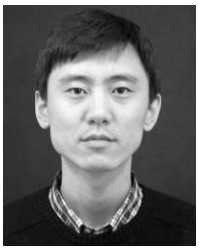
[34] T. Zhou, J. Cao, B. Liu, S. Xu, Z. Zhu, and J. Luo, "Location-based influence maximization in social networks," presented at the 24th ACM Int. Conf. Inf. Knowl. Manage., Melbourne, VIC, Australia, 2015.
[35] X. Y. Wang, Y. Zhang, W. J. Zhang, and X. M. Lin, "Distance-aware influence maximization in geo-social network," in *Proc. 32nd IEEE Int. Conf. Data Eng.*, May 2016, pp. 1–12.
[36] X. Li, X. Cheng, S. Su, and C. Sun, "Community-based seeds selection algorithm for location aware influence maximization," *Neurocomputing*, vol. 275, pp. 1601–1613, Jan. 2018.
[37] S. Su, X. Li, X. Cheng, and C. Sun, "Location-aware targeted influence maximization in social networks," *J. Assoc. Inf. Sci. Technol.*, vol. 69, no. 2, pp. 229–241, 2018.
[38] W.-Y. Zhu, W.-C. Peng, L.-J. Chen, K. Zheng, and X. Zhou, "Modeling user mobility for location promotion in location-based social networks," presented at the 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Sydney, NSW, Australia, 2015.
[39] S. Bharathi, D. Kempe, and M. Salek, "Competitive influence maximization in social networks," presented at the 3rd Int. Conf. Internet Netw. Econ., San Diego, CA, USA, 2007.
[40] T. Carnes, C. Nagarajan, S. M. Wild, and A. V. Zuylen, "Maximizing influence in a competitive social network: A follower's perspective," presented at the 9th Int. Conf. Electron. Commerce, Minneapolis, MN, USA, 2007.
[41] A. Borodin, Y. Filmus, and J. Oren, "Threshold models for competitive influence in social networks," in *Internet and Network Economics*. Berlin, Germany: Springer, 2010, pp. 539–550.
[42] J. Lv, B. Yang, Z. Yang, and W. Zhang, "A community-based algorithm for influence blocking maximization in social networks," *Cluster Comput.*, vol. 17, no. 1, pp. 1–16, Nov. 2017.
[43] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," presented at the 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Paris, France, 2009.



WENLONG ZHU received the M.E. degree from the Department of Computer Science and Technology, Harbin Engineering University, Harbin, China, in 2010, where he is currently pursuing the Ph.D. degree. His main research interests include social networks and data mining.

WU YANG received the Ph.D. degree in computer system architecture from the Computer Science and Technology School, Harbin Institute of Technology. He is currently a Professor and a Doctoral Supervisor with Harbin Engineering University. His main research interests include wireless sensor networks, peer-to-peer networks, and information security. He is a member of ACM and a Senior Member of CCF.

SHICHANG XUAN received the Ph.D. degree in computer science and technology from Harbin Engineering University, in 2017, where he is currently a Lecturer. His main research interests include social networks and information security.



DAPENG MAN received the Ph.D. degree in computer science and technology from Harbin Engineering University, in 2009, where he is currently an Assistant Professor. His main research interests include wireless sensor networks and mobile computing.

WEI WANG received the Ph.D. degree in computer system architecture from the Computer Science and Technology School, Harbin Institute of Technology. He is currently a Professor with Harbin Engineering University. His main research interests include social networks and information security. He is a member of CCF.

XIAOJIANG DU (SM'09) received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, China, in 1996 and 1998, respectively, and the Ph.D. degree in electrical engineering from the University of Maryland at College Park, in 2003. He is currently a Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, USA. His research interests include security, wireless networks, and systems. He has authored over 240 journal and conference papers in these areas. He is a Life Member of ACM.

MOHSEN GUIZANI (S'85–M'89–SM'99–F'09) received the bachelor's (Hons.) and master's degrees in electrical engineering and the master's and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He served as the Associate Vice President of Graduate Studies with Qatar University, the Chair of the Computer Science Department, Western Michigan University, and the Chair of the Computer Science Department, University of West Florida. He also served in academic positions at the University of Missouri-Kansas City, the University of Colorado at Boulder, Syracuse University, and Kuwait University. He is currently a Professor and the ECE Department Chair with the University of Idaho. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid.