Neurocomputing 275 (2018) 1601–1613

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Community-based seeds selection algorithm for location aware influence maximization

Xiao Li, Xiang Cheng, Sen Su*, Chenna Sun

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

ARTICLE INFO

Article history:
Received 4 May 2017
Revised 12 September 2017
Accepted 3 October 2017
Available online 11 October 2017

Communicated by Yongdong Zhang

Keywords:
Social network
Influence maximization
Social influence
Community detection
Location awareness

ABSTRACT

In this paper, we study the location aware influence maximization problem, which finds a seed set to maximize the influence spread on targeted users for a given query. In particular, we consider users who have geographical preferences on queries as targeted users. One challenge of the problem is how to find the targeted users and compute their preferences efficiently for given queries. To address this challenge, based on the R-tree, we devise a PR-tree index structure, in which each tree node stores the location and information of users' geographical preferences. By traversing the PR-tree from the root in depth-first order, we can efficiently find the targeted users. Another challenge of the problem is to devise an algorithm for efficient seeds selection. To solve this challenge, we adopt the maximum influence arborescence (MIA) model to approximate the influence spread, and propose an efficient community-based seeds selection (CSS) algorithm. The proposed CSS algorithm finds seeds efficiently by constructing the PR-tree based indexes offline which precompute users' community based influences, and preferentially computing the marginal influences of those who would be selected as seeds with high probability online. In particular, we propose a community detection algorithm which first computes the social influence based similarities by the MIA model and then adopts the spectral clustering algorithm to find optimal communities of the social network. Experimental results on real-world datasets collected from DoubanEvent demonstrate our proposed algorithm has superiority as compared to several state-of-the-art algorithms in terms of efficiency, while keeping large influence spread.

1. Introduction

In recent years, social networks have become prevalent platforms for product promotion (e.g., viral marketing). Previous studies have proven that the viral marketing strategy is more effective than TV or newspaper advertising. Aiming to find a certain number of users (called seeds) to maximize the expected number of influenced users (called influence spread) through the word-of-mouth effect, influence maximization is the key problem behind viral marketing in social networks [1], which has been extensively studied recently [2–5]. With the proliferation of geo-social networks (such as Foursquare[1], and Facebook[2]), location-based products promotion is becoming more necessary in real applications. For instances, a new opened restaurant in Chelsea, New York, wants to be promoted in a social network platform with viral marketing. The promoting strategy of this restaurant is to provide free meals for a limit k users, who can maximize the influence spread over the targeted users through the powerful word-of mouth effect to attract them dining here. Obviously, the targeted users of this promotion are those who have geographical preferences on Chelsea, New York (i.e., those who frequently have dining near Chelsea, New York). In this promotion, how to select this k users is critical.

Recently, some researchers take the location information of the promotion into consideration, and extend the traditional influence maximization to the location aware influence maximization. Zhou et al. [6] take users' historical mobility behaviors into consideration, and devise two heuristic algorithms to find seeds by using the proposed two phase information diffusion model. However, they do not consider co-influences of the selected seeds. Li et al. [7] study to find k users who have the highest influence over a group of users in a specified region. Wang et al. [8] define the distance-aware influence maximization, which considers the distance between users and the promoted location. However, these studies all assume each user has a known fixed location, which is not realistic. In fact, users in social networks usually check-in

* Corresponding author.
E-mail addresses: lixiao@bupt.edu.cn (X. Li), susen@bupt.edu.cn (S. Su).

[1] www.foursquare.com
[2] www.facebook.com/about/location

https://doi.org/10.1016/j.neucom.2017.10.007
0925-2312/

multiple places and have geographical preferences on multiple locations.

Different from these works, in this paper, we define a location aware influence maximization (LAIM) problem. In particular, given a social network, where each user has different preferences on different locations, and a query with a spatial region R and an integer k, the LAIM requires finding k seeds to maximize the influence spread over the targeted users, who have geographical preferences on the query region R. We show that, under the maximum influence arborescence (MIA) model, the LAIM problem is NP-hard, and its influence spread is submodular and monotone. Therefore, we could extend the existing greedy algorithm [3] with $1-1/e$ approximation ratio to solve this problem. However, since the extended greedy algorithm requires to compute the exact marginal influence of each user and select the user with the largest marginal influence in each iteration, it suffers from poor efficiency. Therefore, we focus on the design of efficient solution for the LAIM problem.

There are two main challenges in solving the LAIM problem efficiently. The first is how to efficiently identify the targeted users and compute their geographical preferences for given queries. Thanks to the check-in information on social networks (e.g., Twitter[3], and Foursquare), we could easily obtain users' geographical preferences. Leveraging such information, we devise a PR-tree index structure for finding the targeted users, where each tree node stores the location and information of users' geographical preferences. In particular, for given queries, we traverse the PR-tree from the root in depth-first order, and prune some tree nodes which have no region intersections with the queries to find targeted users efficiently.

The second challenge is to devise algorithms for efficient seeds selection. To address this challenge, we propose a community-based seeds selection (CSS) algorithm. The basic idea is that, we divide the whole network into communities, and then we find seeds within communities instead of the whole network, i.e., we utilize users' influences within their communities to approximate their influences in the whole network. Our assumption is that each node's influence propagation is limited to the community it resides. In particular, we define a community as a group of users who have frequent contact and are more likely to influence each other within the group than outside of it. To find good communities, we propose a community detection algorithm which first computes the social influence based similarities by the maximum influence arborescence (MIA) diffusion model, and then adopts the spectral clustering algorithm for the weighted directed graphs. To find the seeds efficiently, based on the detected communities, our CSS algorithm first constructs the PR-tree based indexes to store users' community-based influences offline. Then, for the given queries, it assembles the corresponding indexes online to construct the priority queue for each community, which stores the community-based influences of users in descending order. Finally, based on the community-based priority queues, it finds seeds efficiently by preferentially computing the marginal influences of those who would be selected as seeds with high probability online. In particular, we devise a fast method to compute the upper bound of the marginal influences (i.e., estimated marginal influence) of users, which is time-saving for seeds selection.

In summary, the major contributions of our work are listed as follows:

- We formally introduce the location aware influence maximization (LAIM) problem, which is NP-hard, and propose an efficient solution to solve this problem.
- To efficiently identify the targeted users and compute their preferences for given queries, we propose a PR-tree index structure, in which each tree node stores the location and information of users' geographical preferences.
- Based on the spectral clustering, we devise a social influence based community detection algorithm by adopting the MIA model. In addition, we propose an efficient community based seeds selection algorithm by utilizing the offline community-based influence indexes and the online community-based priority queues.
- We conduct a comprehensive performance evaluation on real-world datasets. Experimental results show the effectiveness and efficiency of our proposed algorithm.

The remainder of this paper is organized as follows. We review the related work in Section 2. Section 3 presents necessary background on influence maximization and influence spread computation. In Section 4, we formulate our location aware influence maximization (LAIM) problem. In Section 5, we present our community-based seeds selection solution, followed by experimental evaluation in Section 6. Finally, Section 7 concludes the paper.

2. Related work

2.1. Influence maximization

Influence maximization in social network. Influence maximization (IM) is first described as an algorithmic problem by Demongos and Richardson [9]. Then, Kempe et al. [1] formulate the IM problem as a discrete optimization problem. They prove that this problem is NP-hard and give a greedy algorithm with provable approximation guarantee (i.e., $1-1/e$). However, the greedy algorithm needs to execute the Monte Carlo simulation [10] to obtain the approximation ratio, which faces the drawback of high time complexity. Thus, there has been a large body of research work devoted to improving the efficiency while keeping the theoretical bound. Leskovec et al. [2] develop a "lazy-forward" algorithm, i.e., the CELF algorithm, which greatly reduces the number of evaluations on the influence spread of nodes by taking advantages of the submodularity property. Based on the CELF algorithm, Goyal et al. [11] propose a CELF++ to further decrease the amount of calculation. Instead of using the Monte Carlo simulation, Chen et al. [3] first use the maximum influence arborescence (MIA) model to approximate the influence propagation and propose a PMIA algorithm to solve the IM problem. The similar idea has been applied in [7,8,12]. In this paper, we also use the MIA diffusion model to efficiently approximate users' influences. Another solution to the IM problem is the Reverse Influence Set (RIS), which is based on the sampling technique. Borgs et al. [4] first exploit the RIS method under the IC model, which is more efficient than the greedy algorithm. This motivates some studies [13–15] to further improve the sampling efficiency. Similar to the greedy algorithm, these methods can have provable theoretical guarantee.

Moreover, some heuristic algorithms are proposed to improve the efficiency by discarding the theoretical bound. Zhao et al. [16] propose a IM-PLA algorithm to find influential nodes with community structure which are based on label propagation. As IM-PLA only considers the degree of each node, it has near linear time complexity. However, compared with the greedy algorithm, it has smaller influence spread. Wang et al. [17] propose a community-based greedy algorithm to find top-k influential nodes in mobile social networks. Different from them, our proposed community-based seeds selection algorithm is based on the offline indexes, which is more efficient than algorithm in [17]. Chen et al. [5] design a degree discount heuristic algorithm for a special case of the IC model, where all propagation probabilities between nodes

[3] www.twitter.com

are the same. Kim et al. [18] propose an independent path algorithm for the IC model, which can be parallelized by OpenMP meta-programming expressions. Jiang et al. [19] propose a simulated annealing approach for the IM problem. To find the influential nodes, Wang et al. [20] propose a multi-attribute ranking method by considering the neighborhood's effect upon the influence capability of a node. Lu et al. [21] analyse the bottleneck of the greedy algorithm and design a CascadeDiscount algorithm to solve the influence maximization problem. Zhang et al. [22] propose a genetic algorithm to solve the influence maximization problem through multi-population competition.

Influence maximization in geo-social network. To meet the location-aware requirement in IM, Li et al. [7] study the location-aware IM problem (i.e., finds k users in a geo-social network who have the maximum influence spread over a group of users in a specified region), and propose algorithms with $1-1/e$ approximation ratio and algorithms with $\varepsilon\cdot(1-1/e)$ approximation ratio for any $\varepsilon\in(0,\ 1]$ for online queries. However, they assume each user has a known fixed location, which is not realistic. In fact, users have different preferences on different locations. Wang et al. [8] define the distance-aware IM problem, which takes the distance between users and the promoted location into consideration, and propose a priority based algorithm with $1-1/e$ approximation ratio to solve the problem. Zhu et al. [23] propose two user mobility models, namely the Gaussian based and distance based mobility models, to derive the location aware propagation probability in LBSN. Zhou et al. [6] study the IM under the O2O environment which takes users' historical mobility behaviors into account. Moreover, they propose a two phase (TP) model, which is an improved influence diffusion model capturing both the online influence diffusion process and offline product acceptance process.

Other extensions of influence maximization. Some researchers propose to solve other extended IM problems [24]. By considering users are more likely influenced by their friends with similar topics, topic-aware IM is proposed [12,25,26]. Different from traditional IM, which gives free products to trigger social cascades, Li et al. [27] study the positive IM problem in signed social networks containing both positive relationships (e.g., friend or like) and negative relationships (e.g., enemy or dislike). Aiming to improve the efficiency, they propose a simulated annealing (SA) based method for solving the positive IM problem. Kim et al. [28] propose a RSRS algorithm for the influence maximization problem in dynamic graph, which has faster running time while keeping large influence spread.

2.2. Community detection

Community detection is a challenging problem in social networks. There are a number of algorithms to solve this problem [29,30]. Most studies, which focus on the topological structures, attempt to discover non-overlap communities based on different measures and objectives, such as modularity maximization [31], Louvain algorithm [32], random walk based methods [33] and spectral clustering [34]. With the recent advances in machine learning research [35,36], deep learning has been well employed in the graph clustering [37] by utilizing its advantages in feature learning [38,39]. Wu and Pan [40] propose a novel community detection method by realizing structure clustering technique and attribute categorization technique simultaneously. There are also some works focusing on the overlapping community detection [41–43]. Based on density peaks, Bai et al. [41] present a novel overlapping community detection algorithm, in which a three-step process method is proposed to select cores of communities. Based on structural clustering, Ma et al. [42] convert structural similarity between vertices to weights of network and propose an efficient algorithm to detect overlapping communities.

In addition, some works propose to detect communities by using the social influence between users. Ghosh and Lerman [44] propose to use the paths between users as a measure of network connectivity. Wang et al. [17] propose to detect influence based communities. They first extend the label propagation method with the Independent Cascade (IC) model to partition users in the social network, and then define the combination entropy to combine the partitioned communities. Yang and Liu [45] devise a social influence based the vertex similarity metric and use the k-medoids algorithm to obtain the K clusters of the heterogenous social networks. Lu et al. [46] propose to partition a social network into K disjoint communities such that the sum of influence propagation within each community is maximized for the IC and LT propagation models. Different from these works, we propose to use the maximum influence arborescence (MIA) diffusion model to compute the social influence between users, define the social influence based similarity metric, and adopt the spectral clustering algorithm for the directed weighted graph to detect the social influence based communities in social networks.

3. Preliminaries

3.1. Influence maximization (IM)

A social network is modeled as a directed graph $G=(V,E)$, where nodes in $V=\{v_1,v_2,\ldots,v_n\}$ model the users in the network and edges in E model the friendships or follow relationships between them. The influence maximization (IM) can be formally defined as follows [1].

Definition 1. Given a social network $G=(V,E)$, a specific propagation model C and an integer k, the influence maximization (IM) problem is to find a set of seed nodes S in G, where $|S|=k$, such that under model C, the influence spread of S, denoted by $\sigma(S)$, is maximum. It can be expressed as,

$$S=\arg\max_{T\subseteq V,|T|=k}\sigma(T), \tag{1}$$

where T is any k-node set in G.

Notice that, the influence spread $\sigma(S)$ is defined as the expected number of active users after the propagation, and we will introduce how to compute $\sigma(S)$ next. It is shown in [1] that the influence maximization problem is NP-hard, but a constant-ratio (i.e., $1-1/e$) approximation algorithm is available.

3.2. Influence spread computation model

An important block for solving the IM problem is to compute influence spread $\sigma(S)$ of a seed set S under a certain information diffusion model. A widely adopted dynamic model in information diffusion is the Independent Cascade (IC) model and the linear threshold (LT) model [1]. However, under IC and LT model, computing $\sigma(S)$ is #P-hard [3]. To achieve better performance in computing the influence spread, the maximum influence arborescence (MIA) model [3] is proposed to approximate the IC model. In this paper, we adopt the MIA model to compute the influence spread. Hereafter, when there is no ambiguity, $\sigma(S)$ denotes the influence of S under the MIA model.

Under the MIA model, user u activates v only through the maximum influence path from u to v, denoted by $u\to v$, which is the one with the maximum activation probability between them, i.e., $u\to v=\arg\max_p(P(p))$. Specifically, for a path $p=\langle u=w_1,w_2,\ldots,w_m=v\rangle$ from u to v, $P(p)$ is the propagation probability of the path which is defined as $P(p)=\prod_{i=1}^{m-1}P(w_i,w_{i+1})$. Using the MIA model, we approximate the influence of u to v by $P(u,v)=\max_p(P(p))$. Moreover, MIA prunes insignificant paths

through a threshold θ. In particular, if $P(u, v) < \theta$, we consider v is not influenced by u.

Since different users in seed set S may have correlations when they influence user v, the influence $P(S, v)$ of seed set S to user v could not be simply computed as $\Sigma_{u \in S} P(u, v)$. Thus, in order to compute $P(S, v)$, for user v, we construct a tree structure where v is the root node and all other users in V are the children nodes. Note that, the path in the tree from one node s ($s \in V$) to the root node v is the maximum influence path from user s to user v in the given social network. Given a seed set S, the influence $P(S, v)$ is computed as follows. If $v \in S$, $P(S, v) = 1$; Otherwise, v must be influenced by his/her children (i.e., $D(v)$) in the tree. The probability that v is influenced by the seed set S through one of his/her child $c \in D(v)$ is $P(S, c) \cdot P(c, v)$. Combining influences of all the children, we obtain the influence of S to v, $P(S, v) = 1 - \prod_{c \in D(v)} (1 - P(S, c) \cdot P(c, v))$. In short, $P(S, v)$ is computed as follows:

$$P(S, v) = \begin{cases} 1 & v \in S \\ 1 - \prod\limits_{c \in D(v)} (1 - P(S, c) \cdot P(c, v)) & v \notin S \end{cases}. \quad (2)$$

Finally, the influence spread $\sigma(S)$ of S can be computed by adding up the influence of S to all the users in the social graph, i.e., $\sigma(S) = \sum_{v \in V} P(S, v)$.

4. Problem formulation

4.1. Query model

A location aware influence maximization (LAIM) query Q consists of a region R and a budget number k (i.e., the number of seeds), denoted by $Q = (R, k)$.

4.2. Data model

Users in the social network have geographical preferences. For user v, we define the ratio $\gamma(v, Q)$ ($0 \le \gamma(v, Q) \le 1$) of v's local check-ins in R over the total as his geographical preference on Q:

$$\gamma(v, Q) = \frac{\sum_{l \subseteq C(v) \cap R} n_v(l)}{\sum_{l \subseteq C(v)} n_v(l)}, \quad (3)$$

where $C(v)$ is the set of locations user v has checked-in and $n_v(l)$ denotes the number of check-ins of v at location l.

4.3. Location aware influence maximization

Definition 2. Given a social network G, an information propagation model *MIA*, and a query Q, location aware influence maximization (LAIM) problem is to find k seeds to maximize the influence spread, i.e., $S = \arg\max_{T \subseteq V, |T| = k} \sigma_{RA}(T, Q)$.

Definition 3. Under the MIA model, the influence spread $\sigma_{RA}(S, Q)$ of the LAIM is calculated as $\sigma_{RA}(S, Q) = \sum_{v \in V} P(S, v) \cdot \gamma(v, Q)$, where $\gamma(v, Q)$ is the geographical preference of v on Q and is calculated by Eq. (3).

Example 1. Fig. 1 shows a social network with 6 users. The numbers on each directed edge are the influence probabilities between users. Suppose a LAIM query is $Q = (R, 2)$, and the geographical preference of user d on R is 0.3, i.e., $\gamma(d, Q) = 0.3$. When the seed set is $S = \{a, b\}$, we show its influence to user d, i.e., $\sigma_{RA}(\{a, b\}, d)$. As the child node of d is c, $\sigma_{RA}(\{a, b\}, d) = P(\{a, b\}, d) \cdot \gamma(d, Q) = [1 - (1 - P(\{a, b\}, c) \cdot P(c, d))] \cdot \gamma(d, Q)$. As $P(\{a, b\}, c) = 1 - (1 - P(a, c)) \cdot (1 - P(b, c)) = 0.55$ and $P(c, d) = 0.5$, $\sigma_{RA}(\{a, b\}, d) = 0.275 * 0.3 = 0.08$.

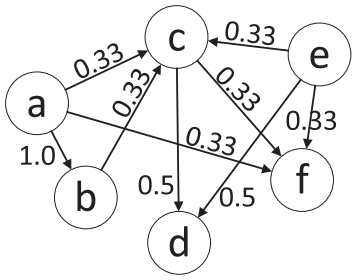


Fig. 1. A social network.

4.4. Problem hardness

The traditional influence maximization problem under the MIA model is NP-hard [3]. As the traditional influence maximization is a special case of the LAIM, where each user's geographical preference on query Q is equal to 1, our LAIM problem is NP-hard.

Theorem 1. *Function $\sigma_{RA}(S, Q)$ is submodular and monotone, and $\sigma_{RA}(\emptyset) = 0$.*

Proof. Obviously, the influence spread function is monotonic and $\sigma_{RA}(\emptyset) = 0$, that is, for any $S \subseteq T$, $\sigma_{RA}(S, Q) \le \sigma_{RA}(T, Q)$. As $P(S, v)$ is submodular [3] and submodular functions are closed under nonnegative linear combinations, the function $\sigma_{RA}(S, Q)$ is submodular. □

Based on the submodular and monotone properties of the influence spread function $\sigma_{RA}(S, Q)$, we can extend the greedy algorithm [3] to solve the LAIM problem (i.e., Algorithm 1), denoted

Algorithm 1: Greedy-E.

Input: $G = (V, E)$: social graph; $Q = (R, k)$: query
Output: S: A seed set
1 Initialize $S = \emptyset$;
2 **for** $i = 1$ *to* k **do**
3 $\quad u = \arg\max_{w \in V \setminus S} (\triangle\sigma_{ST}(w|S, Q))$;
4 $\quad S = S \cup \{u\}$;
5 **return** S;

by Greed-E, which achieves an approximation ratio of $1 - 1/e$. In particular, Greedy-E finds seeds iteratively, and it selects the user who has the largest marginal influence as the current seed in each iteration. The marginal influence $\triangle\sigma_{RA}(u|S, Q)$ of user u under the current seed set S is defined as,

$$\triangle\sigma_{RA}(u|S, Q) = \sigma_{RA}(S \cup \{u\}, Q) - \sigma_{RA}(S, Q). \quad (4)$$

However, since Greedy-E requires to compute the marginal influence of each user (i.e., $\triangle\sigma_{RA}(u|S, Q)$) in each iteration, it suffers from poor efficiency. Thus, in this paper, we focus on the design of efficient solution for the LAIM problem.

5. Our proposed solution

An overall view of our proposed solution is illustrated in Fig. 2. Our framework consists of three components: PR-tree index structure, community detection algorithm and community-based seeds selection algorithm.

PR-tree index structure. Based on R-tree, we propose a PR-tree index structure to find targeted users efficiently. Each node of the PR-tree index stores the location and information of users' geographical preferences. For online queries, we traverse the PR-tree

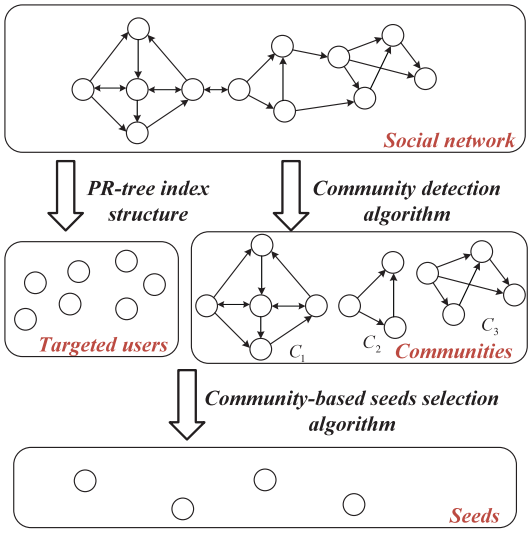


Fig. 2. Overview of the proposed solution.

from the root in depth-first order and prune the tree nodes which have no region intersections with queries.

Community detection algorithm. Based on social influence between users which are computed by using the MIA diffusion model, we propose a social influence based community detection algorithm for the social network. The algorithm detects communities of the social network by utilizing the spectral clustering algorithm for the weighted directed graph.

Community-based seeds selection algorithm. By utilizing the PR-tree index and the communities which are detected in the second component, we devise an efficient community-based seeds selection algorithm. In the algorithm, we first construct the PR-tree based indexes to store users' community-based influences offline. Then, for online queries, we construct a priority queue for each community by assembling the corresponding indexes. Finally, based on the community-based priority queues, we find seeds efficiently by computing the marginal influences of those who would be selected as seeds with high probability. In particular, to avoid unnecessary computation, we devise a fast method to compute the upper bound of marginal influences (i.e., estimated marginal influences) of users.

In the following, we will describe the details of these components, respectively.

5.1. The PR-tree index structure

As the LAIM problem considers users who have geographical preferences on query Q as targeted users, one challenge of the LAIM problem is to efficiently find these targeted users. To address this challenge, based on R-tree [47] and the information retrieval technique [48], we construct a PR-tree index structure, where each tree node stores users' geographical preferences. For a query, we traverse the PR-tree from root to leaf nodes in depth-first order, and prune the nodes which have no region intersections with the query.

In the PR-tree, a leaf node O contains a number of entries of the form $\langle PD, M\rangle$, where PD is the pointer to a document D, and M refers to the minimum bounding rectangle (MBR) of locations within this entry. Let E denote an entry of O. In particular, document D to which $E.PD$ points consists of the following three components:

1. U, which denotes users who have preferences on $E.M$.

User	Locations
a	$(l_0,3),(l_3,1),(l_8,4)$
b	$(l_1,1)(l_2,2)$
c	$(l_6,2),(l_{11},3),(l_{13},1)$
d	$(l_7,2)$
e	$(l_{10},1),(l_{12},1)$
f	$(l_4,2),(l_5,2),(l_9,2)$

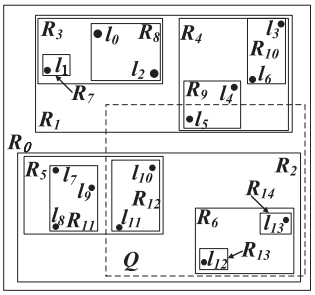


Fig. 3. Users' check-in records and their rectangles.

2. TN, which records the total number of check-ins for each user in U.
3. LN, which records the number of check-ins at each location contained in $E.M$ for each user in U.

In addition, a leaf node O contains a pointer to a pseudo document, which aggregates all documents pointed to by entries of O. Let $O.M$ denote the MBR of O, which can be obtained by aggregating all its entries. The pseudo document also contains three components: U (i.e., users who have preferences on $O.M$), TN (i.e., the total number of check-ins for each user in U) and LN (i.e., the number of check-ins within $O.M$ of each user, which adds up the number of check-ins at locations contained in entries of O).

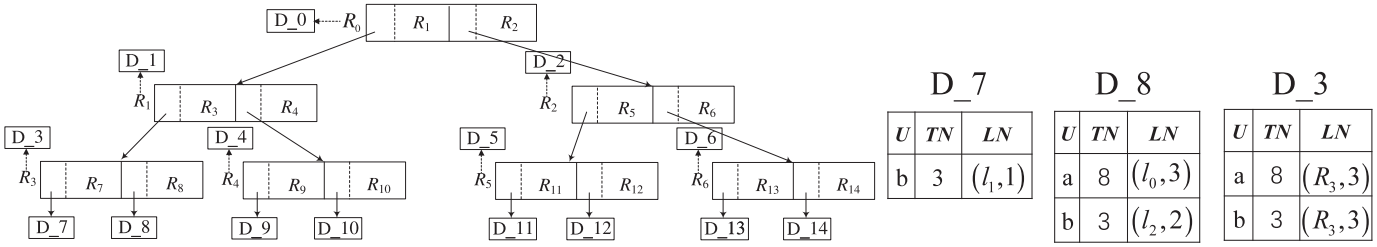
In the PR-tree, a non-leaf node N contains a number of entries of the form $\langle PC, M\rangle$ where PC is the pointer to a child node of N, and M is the MBR of all entries within this child node. A non-leaf node also contains a pointer to a pseudo document, which aggregates all pseudo documents contained in the children of N. The pseudo documents of non-leaf nodes are similar to those of leaf-nodes.

Example 2. In Fig. 3, we describe the check-in records of users in Fig. 1 and their MBRs. Fig. 4 shows the PR-tree index for these users, document D_7, document D_8, and pseudo document D_3.

For a given query $Q=(R,k)$, we introduce how to find the targeted users and obtain their preferences efficiently by traversing the PR-tree in depth-first order. For a tree node B, $B.D$ denotes its pseudo document. As the targeted users of a given query Q are those who have preferences on R, if $B\cap R=\emptyset$, users stored in B are not the targeted users. Thus, we prune node B, and do not visit the subtree rooted at B. Otherwise, (1) if B is fully contained in R (i.e., $B\subseteq R$), all users stored in B have preferences on Q. Thus, we do not visit the children of B and access to the pseudo document $B.D$ to compute each user's geographical preference on B, (i.e., the geographical preference of user v on B is computed by $\gamma(v,B)=\frac{\sum_{l\subseteq C(v)\cap B} n_v(l)}{\sum_{l\subseteq C(v)} n_v(l)}$). (2) If B is not fully contained in R, we visit the children of B. In particular, if B is the leaf node, we visit its entries. We repeat this process iteratively until we identify all targeted users (denoted by P_Q) and compute their geographical preferences. Notice that, as users may exist in not only one tree node, we should compute their preferences on Q by adding up their preferences on tree nodes which store them. Through pruning some tree nodes and accessing the pseudo documents of tree nodes, we can efficiently find the targeted users and obtain their preferences.

5.2. Social influence based community detection algorithm

Community detection is an important but difficult problem in network analysis and has attracted a great deal of effort from many disciplines [32,49]. It provides insight into how the social networks are internally organized. Universally, most researchers describe a



D_7

U	TN	LN
b	3	$(l_1,1)$

D_8

U	TN	LN
a	8	$(l_0,3)$
b	3	$(l_2,2)$

D_3

U	TN	LN
a	8	$(R_3,3)$
b	3	$(R_3,3)$

Fig. 4. The PR-tree index for records in Fig. 3.

community as a group of nodes with similar patterns. In this paper, since we want to utilize the detected communities to select seeds (i.e., we make use of one user's influence spread among the community to approximate his influence spread among the whole social network), we define a community as a group of users who have frequent contact and are more likely to influence each other within the group than outside of it. Thus, existing community detection algorithms could not satisfactorily solve our problem. To this end, we propose a social influence based community detection algorithm, which contains the following two steps: we compute the social influence between two users by adopting the MIA information diffusion model; we adopt the spectral clustering algorithm for the weighted directed graph [50] to find optimal communities of the social networks.

5.2.1. Social influence computation

Given a social network with edge probabilities, we can adopt the MIA model to compute the influence from user u to user v, i.e., $P(u,v) = \max_p(P(p))$, where $P(p)$ is the propagation probability of the path p (i.e., $p = \langle u = w_1, w_2, \ldots, w_m = v\rangle$) which is defined as $P(p) = \prod_{i=1}^{m-1} P(w_i, w_{i+1})$.

5.2.2. Community detection

As our goal is to depart the whole social network into social influence based communities, we construct a social influence based similarity matrix $A = \{a_{ij}\}$, $(i, j = 1, 2, \ldots, n)$, and a_{ij} represents the social influence of user i to user j, i.e., $P(i, j)$. After obtaining the user similarities, a number of graph partition approaches can be adopted to partition the social network. Among these approaches, spectral clustering is one of the most popular modern clustering algorithm, which is simple to implement and outperforms traditional clustering algorithms such as the k-means algorithm. In addition, the social influence based similarity A is usually asymmetric, i.e., $P(u, v) \neq P(v, u)$. Thus, in this paper, we adopt the spectral clustering algorithm for weighted directed graph [50] for the network partition. In particular, we use $H(B)$ computed by Eq. (5) to represent the asymmetric A, in this way, the spectral clustering techniques designed for symmetric similarity can be applied to $H(B)$ to obtain the optimal communities of the social networks.

$$H(B) = \frac{1}{2}T^{-1/2}(2D - A - A^T)T^{-1/2}, \tag{5}$$

where $A = T'A$, $D_i = \sum_{j=1}^{n} A_{ij}$, $D = diag\{D_i\}_i$, $T = T' = \Pi$, $\Pi = diag(\pi_1, \ldots, \pi_n)$, and π_i is the stationary distribution of the random walk.

The pseudo-code of the community detection algorithm is shown in Algorithm 2.

Algorithm 2: Community detection algorithm.

Input: $G = (V, E)$: social graph; m: the number of communities
Output: $C = \{C_1, \ldots, C_m\}$: Set of communities
1 Compute $P(u, v), \forall u, v \in V$;
2 Construct influence based similarity matrix A;
3 Compute $H(B)$ by using Eq. (5);
4 Compute Y: the $n \times K$ matrix with orthonormal columns containing the K smallest eigenvectors of $H(B)$;
5 Cluster $X = T^{-1/2}Y$ by using k-means algorithm;
6 **return** m *communities* $C = \{C_1, \ldots, C_m\}$;

5.3. Community-based seeds selection method

To improve the efficiency for seeds selection, we propose to use users' influence spread among the community to approximate their influence spread among the whole network. To further improve the efficiency, we propose to precompute users' community-based influences offline. In this section, based on PR-tree structure, we first propose two kinds of offline indexes to efficiently store the community-based influences of users, and then devise a community-based seeds selection (CSS) algorithm for online queries.

5.3.1. Indexes

Community-based influence index $\mathcal{I}$. For a PR-tree node R_i, each community $C_j \subseteq C$ is associated with a list $\mathcal{I}(R_i, C_j)$, which stores the influences of users in the community in descending order. Notice that, these communities are detected by using our community detection algorithm introduced in Section 5.2. In particular, we denote P_{R_i} as the set of users who have geographical preferences on region R_i, and $U_{R_i,C_j} = \{v | v \in P_{R_i} \cap C_j\}$ denote the set of users who are in P_{R_i} and reside in community C_j. For a PR-tree node R_i and community C_j, we precompute the influencer sets for users in U_{R_i,C_j}, denoted by F_{R_i,C_j}, i.e., $F_{R_i,C_j} = \{u | P(u, v) \geq \theta, v \in U_{R_i,C_j}\}$. For each user $u \in F_{R_i,C_j}$, we also precompute its influence to users in U_{R_i,C_j}, denoted by $\sigma_{RA}(u, U_{R_i,C_j}) = \sum_{v \in U_{R_i,C_j}} P(u, v) \cdot \gamma(v, R_i)$, where $\gamma(v, R_i)$ is the geographical preference of v on region R_i. Finally, for a PR-tree node R_i, each community C_j has a descending order list $\mathcal{I}(R_i, C_j)$, which is comprised of 2-tuples $\langle u, \sigma_{RA}(u, U_{R_i,C_j})\rangle$, $\forall u \in F_{R_i,C_j}$, sorted by $\sigma_{RA}(u, U_{R_i,C_j})$.

Community-based user index $\mathcal{U}$. We also maintain a user index $\mathcal{U}$. For each user u, we keep a list $\mathcal{U}(u)$ of triples $\langle R_i, C_j, \sigma_{RA}(u, U_{R_i,C_j})\rangle$, where each triple denotes the influence of user u to users who have geographical preferences on R_i and who reside in community C_j.

Example 3. Fig. 5 shows the influence index for PR-tree node R_4 corresponding to communities $C_1, \ldots, C_m$, and Fig. 6 shows the user index for all users.

5.3.2. Community-based seeds selection algorithm

Based on the two indexes $\mathcal{I}$ and $\mathcal{U}$, we devise a community-based seeds selection (CSS) algorithm. Similar to Greedy-E, our CSS algorithm also finds seeds iteratively. The main difference between

$\mathcal{I}(R_4, C_1)$

user	*influence*
$u_{1,1}$	$\sigma_{RA}(u_{1,1}, U_{R_4,C_1})$
$u_{1,2}$	$\sigma_{RA}(u_{1,2}, U_{R_4,C_1})$
$u_{1,3}$	$\sigma_{RA}(u_{1,3}, U_{R_4,C_1})$
…	…

.....

$\mathcal{I}(R_4, C_m)$

user	*influence*
$u_{m,1}$	$\sigma_{RA}(u_{m,1}, U_{R_4,C_m})$
$u_{m,2}$	$\sigma_{RA}(u_{m,2}, U_{R_4,C_m})$
$u_{m,3}$	$\sigma_{RA}(u_{m,3}, U_{R_4,C_m})$
…	…

Fig. 5. The community-based influence index $\mathcal{I}$ for tree node R_4.

$\mathcal{U}$

user	*influence*
$u_{1,1}$	$\langle R_2, C_1, \sigma_{RA}(u_{1,1}, U_{R_2,C_1})\rangle, \langle R_4, C_1, \sigma_{RA}(u_{1,1}, U_{R_4,C_1})\rangle, \ldots$
$u_{1,2}$	$\langle R_1, C_2, \sigma_{RA}(u_{1,2}, U_{R_1,C_2})\rangle, \langle R_4, C_1, \sigma_{RA}(u_{1,2}, U_{R_4,C_1})\rangle, \ldots$
$u_{m,3}$	$\langle R_3, C_m, \sigma_{RA}(u_{m,3}, U_{R_3,C_m})\rangle, \langle R_4, C_m, \sigma_{RA}(u_{m,3}, U_{R_4,C_m})\rangle, \ldots$
…	…

Fig. 6. The community-based user index $\mathcal{U}$.

CSS and Greedy-E is that, CSS utilizes each user's influence among the community to approximate its influence among the whole social network.

For a given query Q, we first introduce how to construct priority queues for m communities which store the candidate seeds and their influences for query Q. Then, we introduce to find the actual seeds efficiently by utilizing the constructed priority queues.

1. Priority queue construction. First, for query $Q = (R, k)$, we traverse the PR-tree from the root and identify the PR-tree nodes that are fully covered by the query region R, denoted by $R_Q = \{R_1, R_2, \ldots, R_i, \ldots, R_r\}$, where $R_i \subseteq R$, and the region $R_0 = R - R_Q$ which is not fully covered by the query region R.

Then, for each node $R_i \in R_Q$, we select lists $\mathcal{I}(R_i, C_j)$, $\forall C_j \subseteq C$, where C is the set of the detected communities. For each community C_j, we combine lists $\mathcal{I}(R_1, C_j)$, $\mathcal{I}(R_2, C_j)$,...,$\mathcal{I}(R_i, C_j)$,...,$\mathcal{I}(R_r, C_j)$ to obtain list $\mathcal{I}(R_Q, C_j)$, which consists of 2-tuples $\langle u, \sigma_{RA}(u, U_{R_Q,C_j})\rangle$. The influence $\sigma_{RA}(u, U_{R_Q,C_j})$ of user u can be computed by

$$\begin{aligned}\sigma_{RA}(u, U_{R_Q,C_j}) &= \sum_{v \in U_{R_Q,C_j}} p(u,v) \cdot \gamma(v, R_Q) \\ &= \sum_{v \in U_{R_i,C_j}} \sum_{R_i \in R_Q} p(u,v) \cdot \gamma(v, R_i) \\ &= \sum_{R_i \in R_Q} \sigma_{RA}(u, U_{R_i,C_j}), \end{aligned} \tag{6}$$

where U_{R_Q,C_j} is the set of users who have geographical preferences on R_Q and reside in community C_j, (i.e., $U_{R_Q,C_j} = \{v | v \in P_{R_Q} \cap C_j\}$, and $P_{R_Q} = \{v | \gamma(v, R_Q) > 0\}$). Notice that, $\sigma_{RA}(u, U_{R_i,C_j})$ is stored in the list $\mathcal{I}(R_i, C_j)$.

Next, let us move to the query region R_0, i.e., $R_0 = R - R_Q$. We denote P_{R_0} as the set of users who have geographical preferences on region R_0, and $U_{R_0,C_j} = \{v | v \in P_{R_0} \cap C_j\}$ as the set of users who have preferences on R_0 and reside in community C_j. For region R_0 and each community $C_j \subseteq C$, we compute the influencer set for users in U_{R_0,C_j}, denoted by F_{R_0,C_j}, i.e., $F_{R_0,C_j} = \{u | P(u,v) \geq \theta, v \in U_{R_0,C_j}\}$. For each user $u \in F_{R_0,C_j}$, we also compute its influence to users in U_{R_0,C_j}, denoted by $\sigma_{RA}(u, U_{R_0,C_j}) = \sum_{v \in U_{R_0,C_j}} P(u,v) \cdot \gamma(v, R_0)$, where $\gamma(v, R_0)$ is the geographical preference of v on region R_0. In this way, for each community $C_j \subseteq C$, we generate a list $\mathcal{I}(R_0, C_j)$ online with 2-tuples (i.e., $\langle u, \sigma_{RA}(u, U_{R_0,C_j})\rangle$).

Finally, for each community $C_j \subseteq C$, we combine lists $\mathcal{I}(R_0, C_j)$ and $\mathcal{I}(R_Q, C_j)$ to efficiently derive list $\mathcal{I}(C_j)$ of 2-tuples $\langle u, \sigma_{RA}(u, U_{C_j})\rangle$. In particular, U_{C_j} denotes the set of users who have preferences on query Q and reside in community C_j. The influence $\sigma_{RA}(u, U_{C_j})$ of user u to users in U_{C_j} can be computed by

$$\sigma_{RA}(u, U_{C_j}) = \sum_{R_i \in \{R_Q \cup R_0\}} \sigma_{RA}(u, U_{R_i,C_j}). \tag{7}$$

In this way, we derive m lists $\mathcal{I}(C_1)$,...,$\mathcal{I}(C_j)$,...,$\mathcal{I}(C_m)$ for m communities. Obviously, users in these m lists are the candidate seeds. Based on these lists, we construct m priority queues, (i.e., each community C_j is corresponding to a priority queue), denoted by $\mathcal{H}(C_1)$, $\mathcal{H}(C_2)$,..., $\mathcal{H}(C_j)$,..., $\mathcal{H}(C_m)$, where $\mathcal{H}(C_j)$ contains a number of triples $\langle u, \sigma_{RA}(u, U_{C_j}), invalid\rangle$, where *invalid* denotes the state of user u. We utilize the states of users to find the seeds efficiently by updating these priority queues.

$\mathcal{H}(C_1)$

user	*influence*	*state*
$u_{1,1}$	$\sigma_{RA}(u_{1,1}, U_{C_1})$	*invalid*
$u_{1,2}$	$\sigma_{RA}(u_{1,2}, U_{C_1})$	*invalid*
$u_{1,3}$	$\sigma_{RA}(u_{1,3}, U_{C_1})$	*invalid*
…	…	…

.....

$\mathcal{H}(C_m)$

user	*influence*	*state*
$u_{m,1}$	$\sigma_{RA}(u_{m,1}, U_{C_m})$	*invalid*
$u_{m,2}$	$\sigma_{RA}(u_{m,2}, U_{C_m})$	*invalid*
$u_{m,3}$	$\sigma_{RA}(u_{m,3}, U_{C_m})$	*invalid*
…	…	…

Fig. 7. Initial states of m priority queues.

Example 4. Fig. 7 shows the initial states of the m priority queues.

2. Seeds selection. Based on m priority queues and the priority-based framework [12], we devise an efficient method to find the seeds iteratively. The basic idea of this priority-based framework is to find seeds by updating the priority queues, and to prune the insignificant candidate seeds by computing their estimated marginal influences. In each iteration, we look up the candidate seeds according to their significance order, i.e., their estimated marginal influences. In particular, we preferentially compute the exact marginal influences of candidate seeds who have larger estimated marginal influences. The reason is that candidate seeds with larger estimated marginal influences might also have larger exact marginal influences, and are selected as seeds with high probability.

In the updating process of each priority queue $\mathcal{H}(C_j)$, $\forall j \in [1, m]$, each user may have three states: (1) *invalid* (denoted by $flag = 0$), the influence of user u is the initial influence (i.e., $\sigma_{RA}(u, U_{C_j})$) or obtained from the last iteration; (2) *estimated* (denoted by $flag = 1$), the influence of u is the upper bound of the exact marginal influence obtained from a fast estimation method, which will be introduced later; 3) *exact* (denoted by $flag = 2$), the influence of user u is the exact marginal influence computed by

$$\triangle\sigma_{RA}(u | S_j, Q) = \sigma_{RA}(S_j \cup \{u\}, Q) - \sigma_{RA}(S_j, Q), \tag{8}$$

where S_j is the current seed set in community C_j and u resides in community C_j.

Next, we introduce how to find the seeds by updating the priority queues in detail.

First seed selection. For the initial m priority queues, we select the first user of each priority queue, i.e., $u_j = \mathcal{H}(C_j).front()$, $\forall j \in [1, m]$. Comparing these first users' influences, i.e., $\sigma_{RA}(u_1, U_{C_1})$, $\sigma_{RA}(u_2, U_{C_2})$,..., $\sigma_{RA}(u_j, U_{C_j})$,..., $\sigma_{RA}(u_m, U_{C_m})$, we select the user who has the largest influence as the first seed, and remove it from the priority queue.

$\mathcal{H}(C_1)$

user	*influence*	*state*
~~$u_{1,1}$~~	~~$\sigma_{RA}(u_{1,1}, U_{C_1})$~~	~~*invalid*~~
$u_{1,2}$	$\triangle\sigma_{RA}(u_{1,2} \mid S_1, Q)$	*exact*
$u_{1,3}$	$\sigma_{RA}(u_{1,3}, U_{C_1})$	*invalid*
…	…	…

.....

$\mathcal{H}(C_m)$

user	*influence*	*state*
$u_{m,1}$	$\Delta\tilde{\sigma}_{RA}(u_{m,1} \mid S_m, Q)$	*estimated*
$u_{m,2}$	$\sigma_{RA}(u_{m,2}, U_{C_m})$	*invalid*
$u_{m,3}$	$\sigma_{RA}(u_{m,3}, U_{C_m})$	*invalid*
…	…	…

Fig. 8. m updated priority queues.

Subsequent seeds selection. Candidate seeds may have co-influences with the current seeds, thus, we require computing candidate seeds' marginal influences under the current seeds to select the subsequent seeds. The subsequent seeds are selected iteratively. In each iteration, we first set states of all users in priority queues as *invalid* (denoted by $flag = 0$) and then update the priority queues continually. For community $C_j \subseteq C$, let I_u denote the set of users who are influenced by user u in C_j, and I_{S_j} denote the set of users who are influenced by the current seed set S_j, respectively.

In each iteration, we select the first user in each priority queue $\mathcal{H}(C_j)$, $\forall j \in [1, m]$. Comparing these first users' influences, we denote the user who has the largest influence as u_t, who is stored in priority queue $\mathcal{H}(C_t)$. If $I_{u_t} \cap I_{S_t} = \emptyset$, u_t has no co-influence with the current seed set S_t in community C_t. Thus, u_t is the next seed. Otherwise, u_t has co-influence with S_t (i.e., $I_{u_t} \cap I_{S_t} \neq \emptyset$), (1) if the state of u_t is *invalid* (i.e., its influence is the initial influence $\sigma_{RA}(u_t, U_{C_t})$, or obtained from the last iteration), we update u_t's influence as estimated marginal influence $\triangle\tilde{\sigma}_{RA}(u_t|S_t, Q)$ computed by Eq. (9), update its state as *estimated* (denoted by $flag = 1$), and re-insert it into $\mathcal{H}(C_t)$; (2) if the state of u_t is *estimated* (i.e., its influence is the estimated marginal influence $\triangle\tilde{\sigma}_{RA}(u_t|S_t, Q)$), we update u_t's influence as exact marginal influence $\triangle\sigma_{RA}(u_t|S_t, Q)$ computed by Eq. (8), update its state as *exact* (denoted by $flag = 2$), and re-insert it into $\mathcal{H}(C_t)$; (3) if the state of u_t is *exact* (i.e., its influence is the exact marginal influence $\triangle\sigma_{RA}(u_t|S_t, Q)$), u_t is the next seed. We repeat this process until we find all the seeds. In this way, we can avoid computing the exact marginal influences of a large number of candidate seeds, which is time-consuming, and find the subsequent seeds iteratively and efficiently.

Example 5. Fig. 8 shows examples of the updated priority queues, which are used to find the subsequent seeds. We assume user $u_{1,1}$ is one of the current seeds.

Next, Theorem 2 provides a fast method to compute the estimated marginal influences of users.

Theorem 2. *Given a query Q, for user u in community C_j, the upper bound of its marginal influence under the current seeds S_j, denoted by $\triangle\tilde{\sigma}_{RA}(u|S_j, Q)$, can be computed by*

$$\triangle\tilde{\sigma}_{RA}(u|S_j, Q) = \sigma_{RA}(u, U_{C_j}) - \min_{v \in U_{C_j}} P(u, v) \cdot \sum_{v \in U_{C_j}} \sigma_{RA}(S_j, v), \tag{9}$$

where $\sigma_{RA}(u, U_{C_j})$ is computed by Eq. (7).

Proof. When S_j and u have independent influences to user v, $P(S_j \cup \{u\}, v) = P(S_j, v) + P(u, v) - P(S_j, v) \cdot P(u, v)$; Otherwise, $P(S_j \cup \{u\}, v) \leq P(S_j, v) + P(u, v) - P(S_j, v) \cdot P(u, v)$.

As $\forall v \in U_{C_j}, 0 < \gamma(v, Q) \leq 1$,

$$P(S_j, v) \cdot \gamma(v, Q) + P(u, v) \cdot \gamma(v, Q) - P(S_j, v) \cdot P(u, v) \cdot \gamma(v, Q) \geqslant P(S_j \cup \{u\}, v) \cdot \gamma(v, Q).$$

Thus,

$$\sum_{v \in U_{C_j}} P(S_j, v) \cdot \gamma(v, Q) + \sum_{v \in U_{C_j}} P(u, v) \cdot \gamma(v, Q) - \sum_{v \in U_{C_j}} P(S_j, v) \cdot P(u, v) \cdot \gamma(v, Q) \geqslant \sum_{v \in U_{C_j}} P(S_j \cup \{u\}, v) \cdot \gamma(v, Q).$$

Thus,

$$\sigma_{RA}(S_j, Q) + \sigma_{RA}(u, U_{C_j}) - \sum_{v \in U_{C_j}} P(S_j, v) \cdot P(u, v) \cdot \gamma(v, Q) \geqslant \sigma_{RA}(S_j \cup \{u\}, Q).$$

As

$$\sum_{v \in U_{C_j}} P(S_j, v) \cdot P(u, v) \cdot \gamma(v, Q) \geqslant \min_{v \in U_{C_j}} P(u, v) \cdot \sum_{v \in U_{C_j}} \sigma_{RA}(S_j, v),$$

$$\sigma_{RA}(S_j, Q) + \sigma_{RA}(u, U_{C_j}) - \min_{v \in U_{C_j}} P(u, v) \cdot \sum_{v \in U_{C_j}} \sigma_{RA}(S_j, v) \geqslant \sigma_{RA}(S_j \cup \{u\}, Q).$$

Thus,

$$\triangle\sigma_{RA}(u|S_j, Q) \leqslant \sigma_{RA}(u, U_{C_j}) - \min_{v \in U_{C_j}} P(u, v) \cdot \sum_{v \in U_{C_j}} \sigma_{RA}(S_j, v).$$

□

The pseudo-code of the CSS algorithm is shown in Algorithm 3 .

Algorithm 3: CSS algorithm.

Input: $G = (V, E)$: social graph; $Q = (R, k)$: query
Output: S: A seed set

//Off-line Processing: Build Index
$C \leftarrow$ Community detection algorithm;
For each PR-tree node R_i, precompute the influence list $\mathcal{I}(R_i, C_j)$ on each community $C_j \in C$;
For each user $u \in V$, precompute the user list $\mathcal{U}(u)$;
Compute the influence set I_u, $\forall u \in V$;
//Online Processing: Get Priority Queue
Find list $\mathcal{I}(R_i, C_j)$, $\forall R_i \in R_Q, C_j \in C$;
for $j = 1$ *to* m **do**
  Get $\mathcal{I}(R_Q, C_j)$ by combining $\mathcal{I}(R_i, C_j), \forall R_i \in R_Q$;
  Compute $\mathcal{I}(R_0, C_j)$;
  Get $\mathcal{I}(C_j)$ by combining $\mathcal{I}(R_Q, C_j)$ and $\mathcal{I}(R_0, C_j)$;
  Construct priority queue $\mathcal{H}(C_j)$;
//Online Processing: First Seed Selection
for $j = 1$ *to* m **do**
  $u_j = \mathcal{H}(C_j).front()$;
Find the user u_l who has the largest influence among $u_j \in [u_1, \ldots, u_m]$, and the corresponding queue $\mathcal{H}(C_l)$;
$S \leftarrow \{u_l\}$; $S_l \leftarrow \{u_l\}$;
Remove u_l from $\mathcal{H}(C_l)$;
//Online Processing: Subsequent Seeds Selection
while $|S| < k$ **do**
  for $j = 1$ *to* m **do**
    Initialize $flag = 0$ for all users in $\mathcal{H}(C_j)$;
    $u_j = \mathcal{H}(C_j).front()$;
  Find the user u_t who has the largest influence among $u_t \in [u_1, \ldots, u_m]$, and $\mathcal{H}(C_t)$;
  if $I_{u_t} \cap I_{S_t} = \emptyset$ **then**
    $S = S \cup \{u_t\}$; $S_t = S_t \cup \{u_t\}$; $I_{S_t} = I_{S_t} \cup I_{u_t}$; Remove u_t from $\mathcal{H}(C_t)$;
  else
    if $u_t.flag = 0$ **then**
      Compute $\triangle\tilde{\sigma}_{RA}(u|S_t, Q)$;Insert $(u_t, \triangle\tilde{\sigma}_{RA}(u|S_t, Q))$ into $\mathcal{H}(C_t)$; $u_t.flag = 1$;
    if $u_t.flag = 1$ **then**
      Compute $\triangle\sigma_{RA}(u|S_t, Q)$;Insert $(u_t, \triangle\sigma_{RA}(u|S_t, Q))$ into $\mathcal{H}(C_t)$; $u_t.flag = 2$;
    if $u_t.flag = 2$ **then**
      $S = S \cup \{u_t\}$; $S_t = S_t \cup \{u_t\}$; $I_{S_t} = I_{S_t} \cup I_{u_t}$;break;
return S;

In the offline processing, based on the detected communities, CSS stores community-based influence index $\mathcal{I}$ for each PR-tree node, and community-based user index $\mathcal{U}$ for each user (lines 3–4). In the online processing, given a query Q, for each community $C_j \in C$, CSS first gets the corresponding lists $\mathcal{I}(R_i, C_j)$ (line 7), computes list $\mathcal{I}(R_0, C_j)$, and obtains list $\mathcal{I}(C_j)$ by combing $\mathcal{I}(R_Q, C_j)$ and $\mathcal{I}(R_0, C_j)$ (lines 9–11). Then, it constructs priority queue $\mathcal{H}(C_j)$ for each community $C_j \in C$ (line 12). Next, it compares influences of the first users in m priority queues to identify the user who has the largest influence, and selects this user as the first seed (lines 13–17). Finally, it finds the subsequent seeds iteratively by computing the estimated marginal influences and exact marginal influences of users (lines 20–33).

6. Experiments

In this section, we study the performance of our method and compare our method with the state-of-the-art algorithms on real-world datasets.

6.1. Experimental setup

6.1.1. Datasets

We use the real-world dataset DoubanEvent for performance evaluation, which is crawled from sh.douban.com. DoubanEvent is the largest event-based social network (EBSN) in China, which not only provides online social networking facilities (e.g., making friends) but also provides a platform for users organizing and participating a variety of real-world social events (e.g., an outdoor activity). In this dataset, each event has a physical location, and users' geographical preferences are obtained from the physical locations of their attending events [51]. To test the scalability with increasing number of users, we randomly sample 10K users, 100K users and 400K users for the dataset.

6.1.2. Propagation probabilities

We utilize two widely used models to generate the propagation probabilities: weighted cascade model and trivalency model [3]. In the weighted cascade model, $P(u, v)$ for edge $u \rightarrow v$ is $1/d(v)$, where $d(v)$ is the in-degree of user v. In the trivalency model, we uniformly at random select a probability from {0.1, 0.01, 0.001} on each edge, which corresponds to high, medium and low probabilities.

6.1.3. Queries

A query $Q = (R, k)$ contains a region R and an integer k. To evaluate the performance in terms of the number of seeds and the region size, we vary the number of seeds k from 10 to 50 with 30 as the default value. The query region sizes are set to 1 km*1 km (small region), 5 km*5 km (medium region), 10 km*10 km (large region) with medium region as the default value. There are 100 queries in each type and we report the average performance. In particular, in all algorithms, we set the influence propagation threshold $\theta = 0.008$ for the weighted cascade model and $\theta = 0.001$ for the trivalency model. We set the number of communities to 10, 20, 35 for the 10K users dataset, 100K users dataset and 400K users dataset, repectively.

6.1.4. Evaluated algorithms

We compare our proposed method with the extended versions of Greedy Algorithm, CELF [2], PMIA [3], Assembly algorithm [7], TPH algorithm [6] and Degree Discount [5], which can be used to solve our problem and are denoted by Greedy-E, CELF-E, PMIA-E, Ass-E, TPH-E and DD, respectively. In particular, Assembly algorithm and TPH algorithm are proposed for solving the location-based influence maximization problem. In order to evaluate the performance of our CSS algorithm, we also use the extension of the CGA [17] (denoted by CGA-E) as a comparison algorithm, which is a community-based seeds selection algorithm and is proposed for solving the traditional influence maximization problem. Besides, in order to evaluate the performance of our proposed community detection algorithm, we first utilize the popular Louvain algorithm [32] for the community detection and then utilize our proposed CSS algorithm for the seeds selection (denoted by L-CSS). In our experiments, given a query $Q = (R, k)$, the offline probability of user u in the TPH-E algorithm is calculated by his geographical preference on the query region (i.e., $\gamma(u, R)$).

6.2. Experimental results

6.2.1. Effectiveness

To evaluate the effectiveness, we compare the influence spread with varying seeds on different datasets, as shown in Figs. 9 and10. Since Greedy-E takes a long time for seeds selection, we only report its influence spread on the 10K dataset. As we can see, our algorithm achieves a slightly smaller influence spread than Greedy-E, CELF-E, PMIA-E and Ass-E, which are greedy algorithms and have approximation guarantee. This is because: Greedy-E, CELF-E, PMIA-E and Ass-E select seeds iteratively and select the user who has the largest marginal influence among the whole network in each iteration, while our CSS algorithm utilizes a user's influence spread among the community to approximate his influence spread among the whole social network. Since our proposed community detection algorithm is based on social influences between users by adopting the MIA propagation model, the detected communities are optimal, i.e., a user's influence spread among his resided community could approximate his influence spread among the whole social network very well. Our CSS has similar influence spread with CGA-E on each dataset. This is because both of our CSS and CGA-E are greedy based algorithm. In each iteration, they select the user who has the largest exact marginal influence among his resided community as the seed. Our community detection algorithm detects communities based on the social influences of users while the Louvain algorithm only utilizes the structure of the social network. Thus, the influence spread of our CSS algorithm is also larger than that of L-CSS algorithm in all datasets under different propagation probabilities. Besides, our CSS algorithm has a significant improvement in terms of effectiveness than TPH-E and DD. The reasons are that, (1) TPH-E is a heuristic algorithm, which selects the top-k influential users in the whole social network as the final seeds, and does not consider the co-influences between these seeds; (2) DD only selects the user who has the largest discount degree as the seed in each iteration, while our CSS algorithm computes the marginal influences of users in each iteration and selects the user who has the largest marginal influence in the resided community as the next seed.

Moreover, we can see that, in each dataset, with the increase of the seeds (i.e., $Q.k$), the influence spread of all algorithms increases, which also happens when we increase the size of datasets. The reasons lie in that more seeds would activate more targeted users, and there are more targeted users in the larger dataset.

6.2.2. Efficiency

We evaluate the efficiency of these algorithms on the three datasets. We report the running time, which is shown in Figs. 11 and12. Notice that, since the difference between our proposed CSS algorithm and the L-CSS algorithm is how to detect communities offline, their running time is the same. Therefore, we do not report the efficiency of the L-CSS algorithm. It is obvious that the Greedy-E takes a long time for seeds selection, as it computes all users' marginal influences and selects the one with the largest marginal

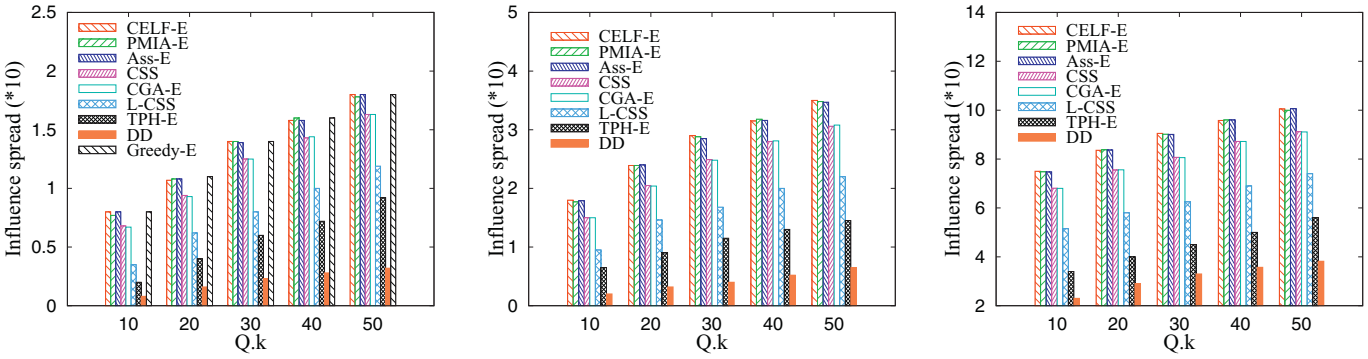


Fig. 9. Influence spread on weighted cascade model.

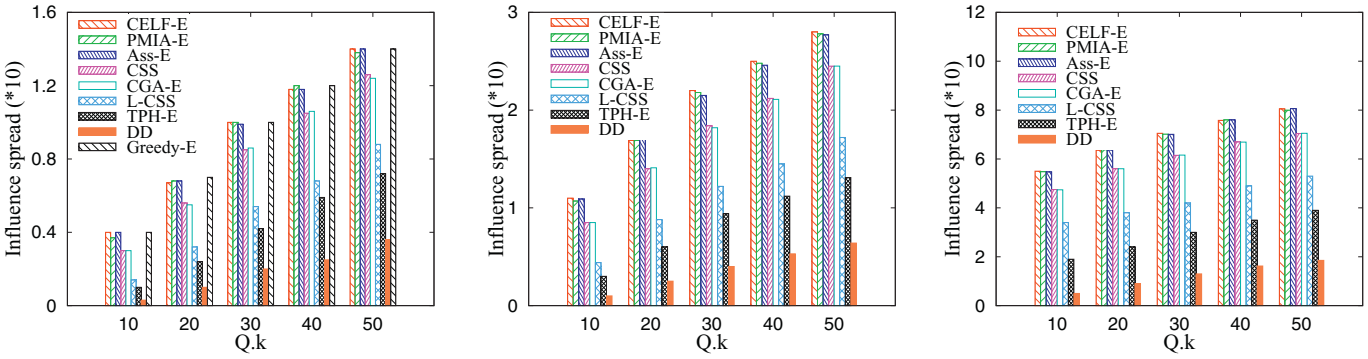


Fig. 10. Influence spread on trivalency model.

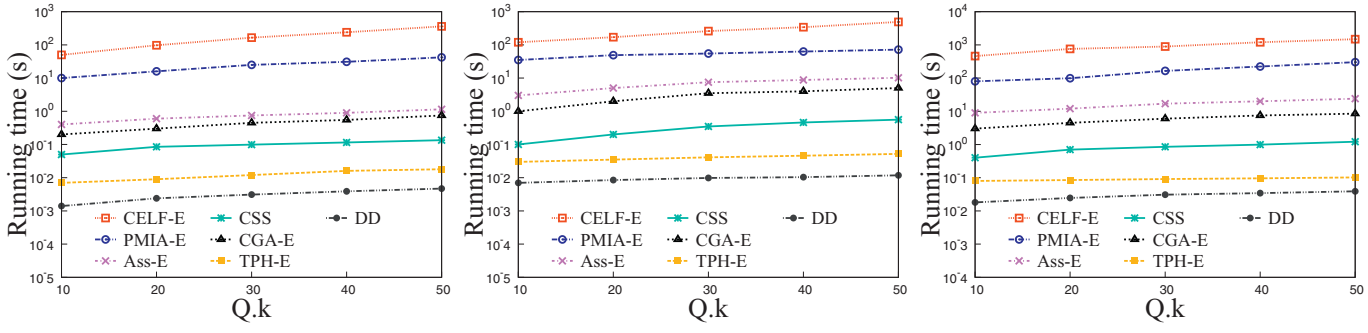


Fig. 11. Efficiency on weighted cascade model.

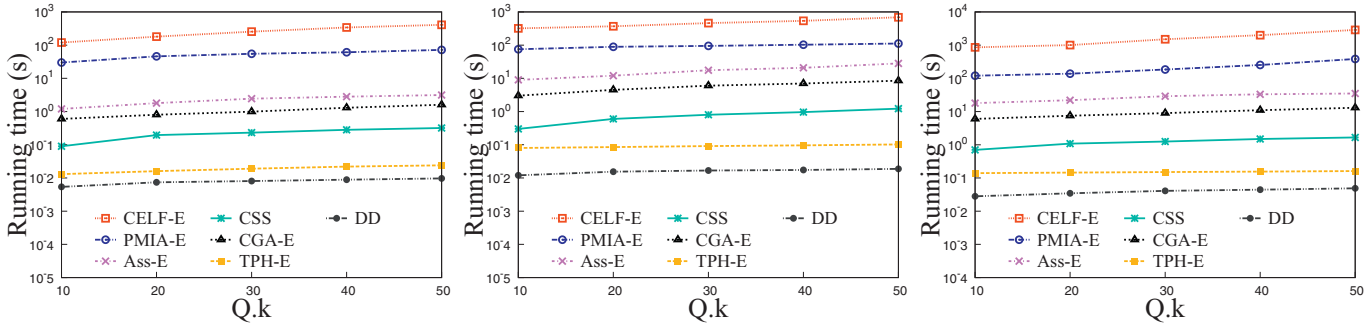


Fig. 12. Efficiency on trivalency model.

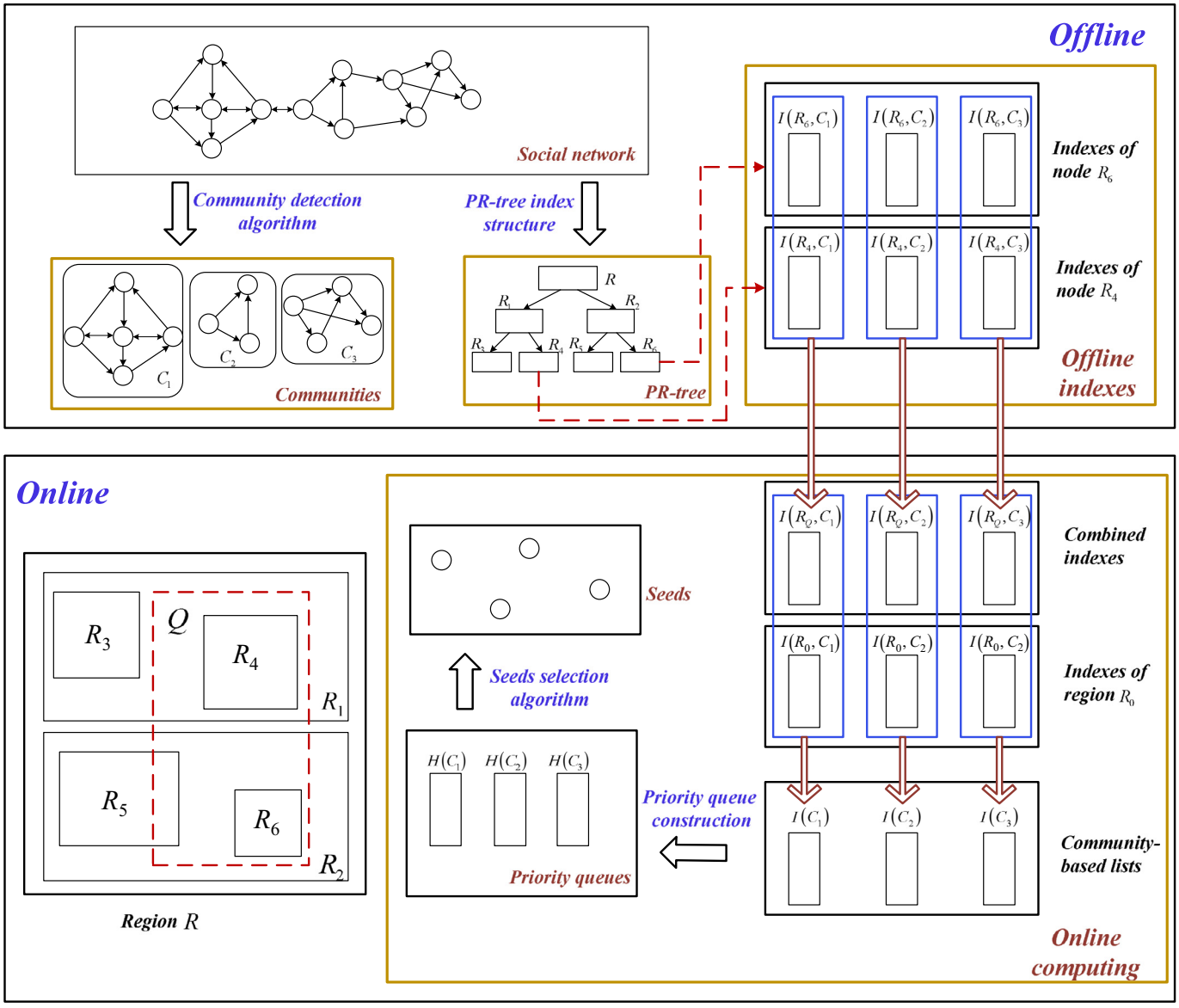


Fig. 13. The whole procedures of our proposed solution.

influence as the seed in each iteration. Thus, we do not report its results. CELF-E runs faster than Greedy-E, as it reduces the computation of the marginal influences of some users by utilizing the submodular property. PMIA-E runs faster than CELF-E, as in each iteration, CELF-E still requires to update many users' marginal influences while PMIA-E only updates users that are affected by the current seeds. Since Ass-E selects seeds by using the constructed offline indexes, it runs faster than PMIA-E.

Compared with all these approximation algorithms (i.e., Greedy-E, CELF-E, PMIA-E and Ass-E), our proposed CSS algorithm has great advantage in terms of efficiency. The whole procedures of our proposed solution are shown in Fig. 13. For an online query Q, the four parts (i.e., four yellow boxes in Fig. 13) are combined together for the seeds selection. In the offline processing, we obtain the PR-tree index by using the PR-tree building method proposed in Section 5.1, find the communities of social networks by using the social influence based community detection algorithm proposed in Section 5.2 and get the community-based indexes by using the method proposed in Section 5.3.1. In the online processing, for each community we first get the community-based lists for query Q and the community-based priority queues by using the method proposed in the first part of Section 5.3.2, and then find the seeds efficiently by using the seeds selection method proposed in the second part of Section 5.3.2.

The detailed reasons why our proposed CSS algorithm has faster running time are shown as follows. Firstly, CSS constructs the community-based influence index and community-based user index which store users' influences based on the PR-tree. For an online query, through combining these indexes, CSS constructs the community-based priority queues and identifies the candidate seeds efficiently. Secondly, as we all know, for the seeds selection, computing the marginal influence of each current user is time-consuming. However, CSS could prune many candidate seeds and only compute the exact marginal influences for small portion of candidate seeds by using a fast method to estimate the upper bound of exact marginal influences. Thirdly, since CSS utilizes the influence spread of users among their resided communities to approximate their influence spread among the whole social network, it requires less running time to compute the estimated and exact

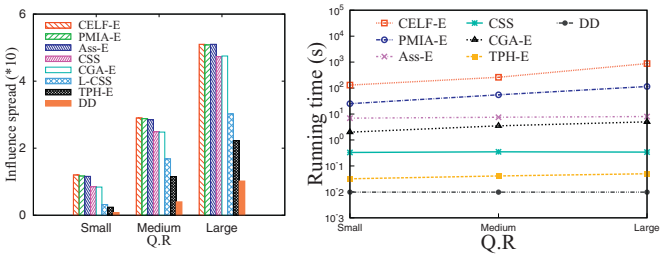


Fig. 14. Varying the query region size: *Q.R*.

marginal influences of candidate seeds. We can also see that, our CSS runs faster than CGA-E. The reason is that, in each iteration, CGA-E requires to compute the exact marginal influences of each user which is time-consuming. While our CSS only computes the exact marginal influences for small portion of candidate seeds by using our proposed strategy. Besides, TPH-E and DD are heuristic algorithms, and they run faster than other algorithms. The reasons lie in, (1) TPH-E selects the top-k influential users as the seeds, and it can save much time as it does not compute the co-influences between seeds; (2) DD finds seeds just by computing each user's discount degree and it does not compute the marginal influences of users.

As shown in Figs. 11 and12, with the increase of $Q.k$, the efficiency of all algorithms decreases. With the increase of the size of datasets, the efficiency of all algorithms also decreases. The reason is that, it costs more time to compute users' influence spread for large dataset. Notice that, in order to improve the efficiency, in all algorithms (except DD), we adopt the PR-tree index to identify the targeted users and compute their geographical preferences.

6.2.3. Varying the query region size

We evaluate the results by varying the query region size $Q.R$ (i.e., small region, medium region and large region) in the 100K users dataset by using the weighted cascade model. We set $Q.k = 30$. Fig. 14 shows the results for the effectiveness and efficiency. We can see that, with the increase of $Q.R$, the influence spread of all algorithms increases. The reasons lie in that with the increase of $Q.R$, the number of targeted users would increase and these targeted users would have larger preference values on larger query region. With the increase of $Q.R$, the running time of CELF-E and PMIA-E increases sharply. It is because with the increase of the number of targeted users, it spends more time to compute each user's exact marginal influence. Although the number of candidate seeds of CSS increases when $Q.R$ increases, the efficiency of CSS is little changed. The reason is that CSS avoids computing the exact marginal influences of a large number of candidate seeds by estimating the upper bound of their exact marginal influences through a fast method. A very interesting observation is that, in some cases, CSS on large query region is even more efficient than that on small region. This is because CSS utilizes the PR-tree based indexes. In particular, when $Q.R$ increases, it may only use a small amount of corresponding lists stored in the non-leaf tree nodes, resulting in less running time in combining these lists and constructing the community-based priority queues for seeds selection. Therefore, CSS has a greater advantage in terms of efficiency for large query region.

7. Conclusion

In this paper, we study the location aware influence maximization (LAIM) problem in social networks. Under the MIA model, we prove that this problem is NP-hard, and the influence spread is monotone and submodular. To obtain the targeted users, (i.e., who have geographical preferences on the query region), we devise a PR-tree index structure. In addition, to find seeds efficiently for the LAIM problem, we propose a community-based seeds selection (CSS) algorithm, which iteratively selects users who have the largest marginal influences in the resided communities by constructing the PR-tree based indexes offline. In particular, based on the spectral clustering, we propose a social influence based community detection algorithm to find optimal communities. Finally, experimental results on real-world datasets show that our algorithm has superiority as compared to several state-of-the-art algorithms.

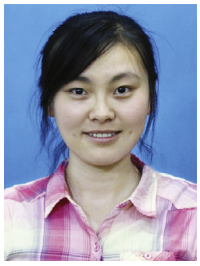
Acknowledgment

The work was supported by the National Natural Science Foundation of China under grant 61502047, the Co-construction Program with the Beijing Municipal Commission of Education.

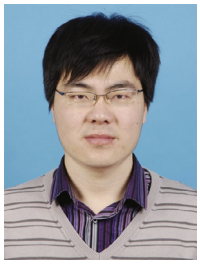
References

[1] D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, in: Proceedings of the 2003 SIGKDD, 2003, pp. 137–146.
[2] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost–effective outbreak detection in networks, in: Proceedings of the 2007 SIGKDD, 2007, pp. 420–429.
[3] W. Chen, C. Wang, Y. Wang, Scalable influence maximization for prevalent viral marketing in large-scale social networks, in: Proceedings of the 2010 SIGKDD, 2010, pp. 1029–1038.
[4] C. Borgs, M. Brautbar, J.T. Chayes, B. Lucier, Influence maximization in social networks: towards an optimal algorithmic solution, arXiv 2012, abs/1212.0884, http://arxiv.org/abs/1212.0884.
[5] W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in: Proceedings of the SIGKDD, 2009, pp. 199–208.
[6] T. Zhou, J. Cao, B. Liu, S. Xu, Z. Zhu, J. Luo, Location-based influence maximization in social networks, in: Proceedings of the Conference on Information and Knowledge Management, 2015, pp. 1211–1220.
[7] G. Li, S. Chen, J. Feng, K.-l. Tan, W.-S. Li, Efficient location-aware influence maximization, in: Proceedings of the 2014 SIGMOD, 2014, pp. 87–98.
[8] X. Wang, Y. Zhang, W. Zhang, X. Lin, Distance-aware influence maximization in geo-social network, in: Proceedings of the IEEE International Conference on Data Engineering, 2016, pp. 1–12.
[9] P. Domingos, M. Richardson, Mining the network value of customers, in: Proceedings of the 2001 SIGKDD, 2001, pp. 57–66.
[10] D.J. MacKay, Introduction to monte carlo methods, in: Learning in Graphical Models, Springer, 1998, pp. 175–204.
[11] A. Goyal, W. Lu, L.V. Lakshmanan, Celf++: optimizing the greedy algorithm for influence maximization in social networks, in: Proceedings of the 2011 WWW, 2011, pp. 47–48.
[12] S. Chen, J. Fan, G. Li, J. Feng, K.-l. Tan, J. Tang, Online topic-aware influence maximization, Proceedings of the VLDB Endowment. Vol. 8 (6) (2015) 666–677.
[13] Y. Li, D. Zhang, K.-L. Tan, Real-time targeted influence maximization for online advertisements, Proceedings of the VLDB Endowment Vol. 8 (10) (2015).
[14] Y. Tang, X. Xiao, Y. Shi, Influence maximization: near-optimal time complexity meets practical efficiency, in: Proceedings of the 2014 SIGMOD, 2014, pp. 75–86.
[15] Y. Tang, Y. Shi, X. Xiao, Influence maximization in near-linear time: a martingale approach, in: Proceedings of the 2014 SIGMOD, 2015, pp. 1539–1554.
[16] Y. Zhao, S. Li, F. Jin, Identification of influential nodes in social networks with community structure based on label propagation, Neurocomputing 210 (2016) 34–44.
[17] Y. Wang, G. Cong, G. Song, K. Xie, Community-based greedy algorithm for mining top-k influential nodes in mobile social networks, in: Proceedings of The Sixteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 1039–1048.
[18] J. Kim, S.-K. Kim, H. Yu, Scalable and parallelizable processing of influence maximization for large-scale social networks? in: Proceedings of the IEEE International Conference on Data Engineering, 2013, pp. 266–277.
[19] Q. Jiang, G. Song, C. Gao, Y. Wang, W. Si, K. Xie, Simulated annealing based influence maximization in social networks, in: Proceedings of the 2011 AAAI, 2011.
[20] Z. Wang, C. Du, J. Fan, Y. Xing, Ranking influential nodes in socialnetworks based on node position and neighborhood, Neurocomputing 260 (2017) 466–477.
[21] F. Lu, W. Zhang, L. Shao, X. Jiang, P. Xu, H. Jin, Scalable influence maximization under independent cascade model, J. Netw. Comput. Appl. 86 (2017) 15–23.
[22] K. Zhang, H. Du, M.W. Feldman, Maximizing influence in a social network: improved results using a genetic algorithm, Phys. A Stat. Mech. Appl. 478 (2017) 20–30.

[23] W.-Y. Zhu, W.-C. Peng, L.-J. Chen, K. Zheng, X. Zhou, Exploiting viral marketing for location promotion in location-based social networks, Trans. Knowl. Discov. Data 11 (2) (2016) 25.
[24] Y. Mehmood, F. Bonchi, D. García-Soriano, Spheres of influence for more effective viral marketing, in: Proceedings of the SIGMOD, 2016.
[25] C. Aslay, N. Barbieri, F. Bonchi, R.A. Baeza-Yates, Online topic-aware influence maximization queries., in: Proceedings of the Extending Database Technology, 2014, pp. 295–306.
[26] W. Chen, T. Lin, C. Yang, Efficient topic-aware influence maximization using preprocessing, CoRR (2014). abs/1403.0057, http://arxiv.org/abs/1403.0057
[27] D. Li, C. Wang, S. Zhang, G. Zhou, D. Chu, C. Wu, Positive influence maximization in signed social networks based on simulated annealing, Neurocomputing 260 (2017) 69–78.
[28] D. Kim, D. Hyeon, J. Oh, W.-S. Han, H. Yu, Influence maximization based on reachability sketches in dynamic graphs, Inf. Sci. 394 (2017) 217–231.
[29] M. Guerrero, F.G. Montoya, R. Baños, A. Alcayde, C. Gil, Adaptive community detection in complex networks using genetic algorithms, Neurocomputing 266 (2017) 101–113.
[30] K. Xu, K. Zou, Y. Huang, X. Yu, X. Zhang, Mining community and inferring friendship in mobile social networks, Neurocomputing 174 (2016) 605–616.
[31] M.E. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. 103 (23) (2006) 8577–8582.
[32] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) P10008.
[33] P. Pons, M. Latapy, Computing communities in large networks using random walks, in: Proceedings of the International Symposium on Computer and Information Sciences, Springer, 2005, pp. 284–293.
[34] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.
[35] C. Yan, H. Xie, S. Liu, J. Yin, Y. Zhang, Q. Dai, Effective Uyghur language text detection in complex background images for traffic prompt identification, IEEE Trans. Intell. Transp. Syst. (2017).
[36] H. Yao, S. Zhang, Y. Zhang, J. Li, Q. Tian, Coarse-to-fine description for fine-grained visual categorization, IEEE Trans. Image Process. 25 (10) (2016) 4858–4872.
[37] F. Tian, B. Gao, Q. Cui, E. Chen, T.-Y. Liu, Learning deep representations for graph clustering., in: Proceedings of the 2014 AAAI, 2014, pp. 1293–1299.
[38] C. Yan, H. Xie, D. Yang, J. Yin, Y. Zhang, Q. Dai, Supervised hash coding with deep neural network for environment perception of intelligent vehicles, IEEE Trans. Intell. Transp. Syst. (2017).
[39] X. Zhang, H. Zhang, Y. Zhang, Y. Yang, M. Wang, H. Luan, J. Li, T.-S. Chua, Deep fusion of multiple semantic cues for complex event recognition, IEEE Trans. Image Process. 25 (3) (2016) 1033–1046.
[40] P. Wu, L. Pan, Multi-objective community detection method by integrating users' behavior attributes, Neurocomputing 210 (2016) 13–25.
[41] X. Bai, P. Yang, X. Shi, An overlapping community detection algorithm based on density peaks, Neurocomputing 226 (2017) 7–15.
[42] T. Ma, Y. Wang, M. Tang, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, Led: a fast overlapping communities detection algorithm based on structural clustering, Neurocomputing 207 (2016) 488–500.
[43] X. Wang, G. Liu, L. Pan, J. Li, Uncovering fuzzy communities in networks with structural similarity, Neurocomputing 210 (2016) 26–33.
[44] R. Ghosh, K. Lerman, Community detection using a measure of global influence, in: Advances in Social Network Mining and Analysis, Springer, 2010, pp. 20–35.
[45] Y. Zhou, L. Liu, Social influence based clustering of heterogeneous information networks, in: Proceedings of the Nineteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 338–346.
[46] Z. Lu, Y. Zhu, W. Li, W. Wu, X. Cheng, Influence-based community partition for social networks, Comput. Soc. Netw. 1 (1) (2014) 1.
[47] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: Proceedings of the SIGMOD, 1984, pp. 47–57.
[48] L. Zhang, Y. Zhang, X. Gu, J. Tang, Q. Tian, Scalable similarity search with topology preserving hashing, IEEE Trans. Image Process. 23 (7) (2014) 3025–3039.
[49] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3) (2010) 75–174.
[50] M. Meilă, W. Pentney, Clustering by weighted cuts in directed graphs, in: Proceedings of the SIAM Conference on Data Mining (SDM), Vol. 1, SIAM, February 2007, p. 2008. Retrieved.
[51] X. Li, X. Cheng, S. Su, S. Li, J. Yang, A hybrid collaborative filtering model for social influence prediction in event-based social networks, Neurocomputing 230 (2017) 197–209.



Xiao Li received her M.S. degree from Shandong Normal University in 2014. She is currently a Ph.D. candidate at Beijing University of Posts and Telecommunications, China. Her major is Computer Science. Her research interests include social influence analysis and machine learning.



Xiang Cheng received the Ph.D. degree from Beijing University of Posts and Telecommunications, China, in 2013. He is currently an Associate Professor at the Beijing University of Posts and Telecommunications. His research interests include data mining and data privacy.

Sen Su received the Ph.D. degree in Computer Science from the University of Electronic Science and Technology, China, in 1998. He is currently a Professor at the Beijing University of Posts and Telecommunications. His research interests include distributed systems and service computing.

Chenna Sun received the B.S. degree from Beijing University of Posts and Telecommunications, China, in 2016. She is currently a M.S. candidate at Beijing University of Posts and Telecommunications. Her major is Computer Science. Her research interests include social network analysis and machine learning.