ARTICLE IN PRESS

Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Event recommendation in social networks based on reverse random walk and participant scale control

Yijun Mo [a], Bixi Li [a], Bang Wang [a], Laurence T. Yang [b,c], Minghua Xu [d,*]

[a] School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, China

[b] School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China

[c] Department of Computer Science, St. Francis Xavier University, Antigonish, Canada

[d] School of Journalism and Information Communication, Huazhong University of Science and Technology (HUST), Wuhan, China

HIGHLIGHTS

- Construct a heterogeneous graph for an event based social network.
- Propose a reverse random walk with restart to compute node proximity on the graph.
- Propose two algorithms to implement event participant scale control.
- Conduct experiments to confirm the superiority of the proposed scheme.

ARTICLE INFO

Article history:
Received 18 September 2016
Received in revised form
6 December 2016
Accepted 24 February 2017
Available online xxxx

Keywords:
Event recommendation
Reverse random walk
Participant scale control
Event-based social networks

ABSTRACT

With the merging of cyber world and physical world, event-based social networks have been playing an important role in promoting the spread of offline social events through online channels. Event recommendation in social networks, which is to recommend a list of upcoming events to a user according to his preference, has attracted a lot of research interests recently. In this paper, we study the event recommendation problem based on the graph theory. We first construct a heterogeneous graph to represent the interactions among different types of entities in an event-based social network. Based on the constructed graph, we propose a novel event scoring algorithm called reverse random walk with restart to obtain the user–event recommendation matrix. In practice, the participant capacity of an event may be constrained to a limited number of users. Then based on the user–event recommendation matrix, we further propose two participant scale control algorithms to coordinate unbalanced user arrangements among events. After the rearrangement, each user will be assigned a list of recommended events, which considers both local user preference and global event capacity. Experiment results on Meetup dataset show that the proposed method outperforms the state-of-art algorithms in terms of higher recommendation precision and larger recommendation coverage.

1. Introduction

Recent years have witnessed the popularity of *event-based social networks* (EBSNs), such as Meetup,[1] Plancast[2] and Tencent Event, acting as a bridge between the cyber world and the physical world. These EBSNs offer a convenient platform for users to establish and share various social events, including cocktail parties, concerts, outdoor hiking and etc. Furthermore, these EBSNs also apply some event detection techniques to discover upcoming events from the internet. For example, a real time event detection method based on social streams has been introduced in [1]. How to mine the variety internet data and discover new events remains the research interest of the Social Event Detection (SED) task of MediaEval.[3]

With the help of EBSN platforms, users could be easily informed about upcoming events from the cyber space and decide to attend

* Corresponding author.
E-mail addresses: moyj@hust.edu.cn (Y. Mo), 429715329@qq.com (B. Li), wangbang@hust.edu.cn (B. Wang), ltyang@gmail.com (L.T. Yang), xuminghua@hust.edu.cn (M. Xu).

1 http://www.meetup.com/.

2 http://www.plancast.com/.

3 http://www.multimediaeval.org/.

http://dx.doi.org/10.1016/j.future.2017.02.045
0167-739X/

offline events in the physical world according to his preferences. However, the number of upcoming events is often very huge, which makes searching interested events an cumbersome task for EBSN users. For example, Meetup currently has 16 million users with more than 300,000 monthly events [2]. Although some of the existing EBSNs provide users with the interface to retrieve, classify, and rank the events by manually setting some search conditions, it is still very difficult for users to express their preferences and locate interested events from the mass of resources, let alone expanding his preferences to some novel events. In order to help a user to quickly obtain his interested events, a good social event recommendation system is in demand according to his preference.

Event recommendation can be simply formulated as the following scoring problem. Let $U = \{u_1, u_2, \ldots, u_M\}$ and $E = \{e_1, e_2, \ldots, e_N\}$ denote the set of M users and N events, respectively. For a given user u_i, event recommendation is to rank the upcoming events in his preference order. Let $s_i(e_j)$ denote the preference score of user u_i to attend the event e_j. Let $L_i^K = \langle e_i^{(1)}, e_i^{(2)}, \ldots, e_i^{(K)} \rangle$ (normally, $K \ll N$) denote the list of recommended K events for user u_i, and in the order of preference score from large to small. For each user, the challenging work of event recommendation is to predict the preference score $s_i(\cdot)$ of each upcoming event. The basic objective is to improve the recommendation accuracy. Besides, recommendation diversity is also an important metric, which indicates the serendipity capability of finding novel yet interesting events for catering to users [3].

Unlike general item recommendation, such as recommending books, films and etc., a social event has its unique temporal and spatial characteristics. Before an event takes place, no one could have '*consumed*' it and offered a valid evaluation about the event [4], which raises the issue of new item cold-start problem. In practice, a common approach to alleviate this problem is to adopt the registration or RSVP mechanism. After an event is announced, a user can register this event, which reveals his intention of participation [5]. The location of a social event also plays a role in event recommendation [6]. Some users like to attend events near their homes; While some other users tend to participate in events in their favorite regions [7]. These two specific dimensions of social events determine that event recommendation would be much different from general item recommendation.

Event recommendation has more interesting characteristics in terms of heterogeneous social relations among users [8]. In Meetup, users form various online groups, and a user often joins multiple groups. Generally, users of a same online group share some similar interests. On the other hand, users share offline social interactions by co-attending offline social events, and those who have attended a same event are likely to show some similar preferences. People who share close connections in both the online and offline social networks are likely to have similar preferences towards future events. Thus capturing such characteristics of both online and offline social networks can help to predict a user's decision more effectively [9].

Since the emergence of collaborative filtering technology, the recommender system has been extensively researched in the last decade. This method is based on the past item evaluations from a large group of users. For each user, the system will firstly find its taste mates by computing similarity of the past evaluation vectors, and predict his preference to an unvisited item with the weighted average value of his fellows towards this item [10]. Another mainstream of research focus is the content-based recommendation, which recommends items that are similar to the content of previously preferred items of a target user [11]. However, the collaborative filtering approaches often encounter the data sparsity and cold-start problem [12]; While the content-based methods also have the technical bottleneck of content abstraction and preference localization. Furthermore, the two classic approaches have not fully explored the unique characteristics for event recommendation and all the possible connections in an event-based social network.

In this paper, we study the event recommendation from a graph-based approach with further considerations of event participant capacity. We first identify the factors that could represent a user's preference, including his geographic location, his online and offline social network connections, as well as his interests expressed by tags. To relieve the data sparsity problem, we conduct dimension reduction to abstract latent entities upon locations and tags. We then construct a heterogeneous graph model to express the interactions of multiple entities and convert the recommendation problem into a node-dependent proximity calculation problem on the heterogeneous graph. We propose a novel Reverse Random walk with Restart (RRWR) method to obtain the user–event recommendation matrix. Then based on this matrix, we further propose two participant scale control algorithms to coordinate unbalanced user arrangements among events. After the rearrangement, each user will be assigned a list of recommended events, which considers both local user preference and global event capacity. Experiment results on Meetup data show that our algorithm can outperform the state-of-the-art methods in terms of higher recommendation precision and larger recommendation coverage.

The rest of paper is organized as follows: Section 2 reviews the related work. Section 3 abstracts latent entities and constructs the heterogeneous graph. Section 4 proposes the reverse random walk with restart algorithms, and two participant scale control algorithms are presented in Section 5. Section 6 provides our experiment results, and the paper is concluded in Section 7.

2. Related work

We briefly review the related work on event recommendation from two aspects: factor-based and graph-based recommendation. We also review some recent efforts on event recommendation that incorporates some practical constraints.

2.1. Factor-based recommendation

In the factor-based recommendation, some latent factors that could impact on event recommendation are firstly extracted and then used as inputs for some score computation model [5,7,13–15]. For example, Guo et al. [14] construct two kinds of networks, and define three variations of relations between users in their work. In [15], Chen et al. propose to exploit a user's social interaction relations and collaborative friendships for event recommendation. The geographic location also influence a user's choice by either distance factor or his regional preference [7,16–19]. However some observations in [20] show that the distance factor is not a pivotal factor within the domain of a city. Tag-aware recommendation systems consider a user's interest expressed by tags [21]. As tags are not predefined by administrators, the number of tags is always very large and hard to interpret. Many approaches have been proposed to analyze tags, including topic-based models [22–24], network-based models [25] and tensor-based models [26,27]. In our work, we comprehensively consider the geographic locations, social interactions as well as the interest subjects to predict users' preferences to upcoming events.

2.2. Graph-based recommendation

In graph-based event recommendation, a graph is constructed to express the interactions of entities in a recommender system, where entities are represented as nodes and interactions between them are represented as edges [28–31]. Based on the graph model, the recommendation task is then converted into a node

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

proximity problem. The nodes with larger proximity values with the query nodes are selected to compose the recommendation list. To calculate the structural proximity of nodes in a graph model, several types of methods have been proposed, e.g., node-dependent methods, path-dependent methods, and random walk methods [3]. For example, random walk with restart (RWR) is a widely used technique to calculate the proximity of nodes in a graph model [28].

The random walk methods conduct a Markov process on the graph model. Given an initial state, the probability distribution iterates according to the transition matrix. The state of the $(t+1)$th iteration only depends on that of the tth iteration. According to the node types, a graph can be divided into homogeneous graph and heterogeneous graph. The univariate and multivariate Markov chain are applied for the two types of graphs, respectively. For a heterogeneous graph, a simple method is to transform it into a homogeneous graph by treating all the nodes and edges as the same type. The main stream of research interest focuses on assigning weights to different types of edges according to expert knowledge or some supervised methods [28–30]. For example, in [31] a learning scheme is proposed to automatically determine the transition parameters between different types of entities.

2.3. Event participant capacity

Many event recommendation systems do not take the event participant capacity into account, which may lead to some popular events having been recommended to too many users beyond its host capacity. Recently, She et al. [32] propose a conflict aware event-participation arrangement algorithm, which solves the participant capacity problem by taking into consideration both the user and event capacity. However, the capacity and conflict information is based on randomly produced synthetic data. They extended their work by including more constraints including cost, distance and time limitations in [33,34]. Their initial approaches can achieve better and balanced recommendation results, yet they only examine their results based on randomly synthesized data.

Compared with the state-of-the-art work, our proposed method consisting of the RRWR algorithm and the scale control algorithm can achieve higher recommendation precision and larger recommendation coverage.

3. Model construction

Fig. 1 illustrates a typical example of event-based social network, which consists of five types of entities: users, events, groups, tags and regions, as well as their relations. In this example, John joins the group G1 and Mike joins Group G2; While Billy joins both groups. The home address of Billy is in region R1, in which the riding activity (E1) is held by group G1. John prefers the tag "outdoor" while group G1 also chooses this tag to indicate their interests. We construct a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ denotes the set of nodes, $\mathcal{E}$ denotes the set of edges, and $\mathcal{W}$ records the weights of edges. In this section, we present our heterogeneous graph construction based on the analysis of Meetup dataset.

3.1. Node selection

There are five kinds of entities in Meetup dataset, including user, event, group, tag, location (longitude and latitude). We directly adopt the first three entities as nodes in our graph. Furthermore, to relieve the sparsity problem in the graph, we conduct clustering operation to abstract locations and tags into conceptually advanced entities, namely, regions and subjects, respectively.

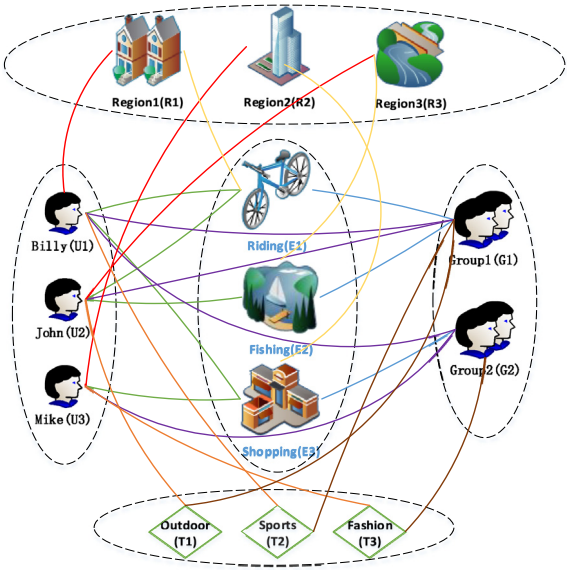


Fig. 1. An illustration of event-based social network.

3.1.1. Region node

Although a user's geographical preference can be measured through the distance between this user and an event, the volume of such distance information might be very prohibitive to be practically processed. Instead, we can divide a city into several regions such that the geographical preference of a user can be measured by his attendance to events in each region. Indeed, people are likely to attend events which will happen in their favorite regions. Let l_{u_i} and l_{e_j} denote the location of user u_i and event e_j, respectively. We adopt the k-means clustering algorithm to group users and events according to their locations into $|R|$ clusters, $\{r_1, r_2, \ldots, r_{|R|}\}$, each corresponding to a region. After the clustering process, each user/event belongs to only one region, i.e., $l_{u_i} \in r_k (1 \leq k \leq |R|)$.

3.1.2. Subject Node

A tag, which contains one or more keywords, is usually used to describe user interest and event characteristics. In Meetup, users and groups can mark themselves with different tags. Users who possess common tags are likely to share similar preference. Furthermore, when a user joins a group, it is likely that he endorses the tags of the group. However, the number of tags is very large. To solve this scalability problem, we apply clustering algorithm to group all tags into a few subjects.

The tag clustering is based on the historical statistics of user–group relation, group–tag description, and user–tag description. Let $\mathbf{UG} = [a_{ig}]_{M\times G}$ denote the user–group relation matrix: $a_{ig} = 1$ indicates that the user u_i is a member of the group G_g; Otherwise, $a_{ig} = 0$. Let T denote the number of total available tags for all users and groups. Let $\mathbf{GT} = [b_{gt}]_{G\times T}$ denote the group–tag description matrix: $b_{gt} = 1$ indicates that the group G_g is tagged with a tag T_t; Otherwise, $b_{gt} = 0$. Let $\mathbf{UT} = [c_{it}]_{M\times T}$ denote the user–tag description matrix: $c_{it} = 1$ indicates that the user u_i is tagged with a tag T_t; Otherwise, $c_{it} = 0$.

Although a user may have its own tags, he may also be implicitly tagged by the group he joined in. To include both explicit and implicit tags for each user, we construct a new user–tag matrix $\mathbf{M} = \mathbf{UT} + \mathbf{UG} \bullet \mathbf{GT}$ with size of $M \times T$. Each element m_{it} can be explained as the frequency that user u_i has possessed tag T_t. Let $\vec{\mathbf{m}}_t$ denote the tth column vector of $\mathbf{M}$. The clustering is to group in total T tags into $|S|$ latent subjects, and $|S| < T$. In this paper, we use the K-means clustering algorithm, and the clustering

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

is performed over the column vectors of $\mathbf{M}$. That is, we cluster T such column vectors $\overrightarrow{\mathbf{m}}_t$ into $|S|$ subjects. We adopt the Euclidean distance in $\overrightarrow{\mathbf{m}}_t$ and $\overrightarrow{\mathbf{m}}_{t'}$ as the difference measure in the clustering. Thus, we can obtain the latent subjects $S = \{S_1, S_2, \ldots, S_{|S|}\}$. Each tag belongs to a certain subject and a subject may contain a series of tags.

Let $U = \{u_1, u_2, \ldots, u_M\}, E = \{e_1, e_2, \ldots, e_N\}, G = \{g_1, g_2, \ldots, g_{|G|}\}, S = \{s_1, s_2, \ldots, s_{|S|}\}, R = \{r_1, r_2, \ldots, r_{|R|}\}$ be the set of users, events, groups, subjects and regions, respectively. Then $\mathcal{V} = U \cup E \cup G \cup S \cup R$ denote the node set of the graph model.

3.2. Definition of edges

Edges in the graph represent the correlation in between nodes. In our heterogeneous graph, we only consider the following correlations in between two node sets: (U, E), (U, G), (U, S), (U, R), (E, G), (E, R), (G, S), because they are the most important edges to describe the interactions between entity nodes.

We take the edge construction for (U, E) as an example. If a user u_i has attended an event e_j, two edges $\langle u_i, e_j\rangle$ and $\langle e_j, u_i\rangle$ each with initial weight one exist in between the node u_i and e_j. Let $\mathcal{E}_{UE}$ and $\mathcal{E}_{EU}$ denote the set of edges from U to E and the set of edges from E to U, respectively. And let A_{UE} and A_{EU} denote their respective adjacent matrix. Note that $A_{UE}(u_i, e_j) = 1$ and $A_{EU}(e_j, u_i) = 1$ for the user u_i having attended the event e_j; And otherwise, equal to zero.

Similarly, we construct the edge sets and adjacent matrices for (U, G), (E, G), (U, R), (E, R) as follows. If a user u_i (an event e_i) belongs to group g_j, two edges exist in between them. If a user u_i (e_i) is located in region r_j, two edges exist in between them. And we can construct the edge sets and adjacent matrices correspondingly.

Specifically, for (U, S), we also need to consider the preference degree for a user u_i to a subject s_j. Note that a user may possess more than one tag, and a subject may also include more than one tag. We then use the times that u_i possesses the tags which also belong to s_j to reveal the preference degree. Recall that in the user–tag matrix $\mathbf{M}$ defined in the previous section, an element m_{it} denote the frequency that a user u_i has possessed tag T_t. Let $\mathcal{T}_j$ denote the set of tags by the subject s_j. We compute the weight between u_i and s_j by:

$$A_{US}(u_i, s_j) = \sum_{t=1}^{T} m_{it} \times \mathbb{I}(T_t \in \mathcal{T}_j), \quad j = 1, 2, \ldots, |S| \tag{1}$$

where $\mathbb{I}(\cdot)$ is an indicator function. If a tag T_t belongs to subject s_j, $\mathbb{I}(\cdot) = 1$; Otherwise, $\mathbb{I}(\cdot) = 0$. Note that only if $A_{US}(u_i, s_j) > 0$, two edges $\langle u_i, s_j\rangle$ and $\langle s_j, u_i\rangle$ exist in between a user u_i and a subject s_j.

For (G, S), since each group is described by a set of tags, we can simply counter the intersection between a group tag set and a subject tag set to calculate the weight between a group g_i and a subject s_j.

4. Reverse random walk with restart

4.1. The basic idea

In this section, we propose a Reverse Random Walk with Restart (RRWR) algorithm for event recommendation based on the constructed heterogeneous graph. The basic idea of RRWR is to compute the proximity of user nodes to each candidate event node to be recommended. At first, we randomly set the initial row probability vectors for users, events, groups, regions and subjects by $\mathbf{u}^{(0)}$, $\mathbf{e}^{(0)}$, $\mathbf{g}^{(0)}$, $\mathbf{r}^{(0)}$, $\mathbf{s}^{(0)}$, while ensuring the sum of elements in each vector equal to one. It then performs random walk on the heterogeneous graph based on the transition matrix in an iterative way. In each iteration, each node transfers some of its probability to its adjacent nodes in proportion to the weight of each edge connecting them. In the meanwhile each node is also assigned some probability to return back to the restarting node. In our RRWR, to recommend a candidate event, we use this candidate event node as the restarting node in order to strengthen the nodes that have closer relation with the querying event node. The iteration terminates, until the probability value assigned to each node converges to a steady state. After the iteration, each user node is assigned a probability value which measures its proximity to this candidate event.

4.2. Transition matrix

To conduct the random walk process, the transition matrix is obtained by normalizing the corresponding adjacent matrices by rows. Let $\mathbf{P}_{UE}$ denote the transition matrix normalized from the adjacent matrix A_{UE}. In (U, E), it may exist some *dangling user node* which does not connect to any event. To ensure the connectivity of the graph, we assume that this user node connects to all events each with equal weight. Similarly, we can construct all other transition matrices.

4.3. Reverse random walk with restart method

Based on the constructed heterogeneous graph and transition matrices, the RRWR method follows the iterative equations:

$$\mathbf{u}^{(t+1)} = \alpha_{EU}\mathbf{e}^{(t)}\mathbf{P}_{EU} + \alpha_{GU}\mathbf{g}^{(t)}\mathbf{P}_{GU} + \alpha_{SU}\mathbf{s}^{(t)}\mathbf{P}_{SU} + (1 - \alpha_{EU} - \alpha_{GU} - \alpha_{SU})\mathbf{r}^{(t)}\mathbf{P}_{RU} \tag{2}$$

$$\mathbf{e}^{(t+1)} = \alpha_{UE}\mathbf{u}^{(t)}\mathbf{P}_{UE} + \alpha_{GE}\mathbf{g}^{(t)}\mathbf{P}_{GE} + \alpha_{RE}\mathbf{r}^{(t)}\mathbf{P}_{RE} + (1 - \alpha_{UE} - \alpha_{GE} - \alpha_{RE})\mathbf{q}_e \tag{3}$$

$$\mathbf{g}^{(t+1)} = \alpha_{UG}\mathbf{u}^{(t)}\mathbf{P}_{UG} + \alpha_{EG}\mathbf{e}^{(t)}\mathbf{P}_{EG} + (1 - \alpha_{UG} - \alpha_{EG})\mathbf{s}^{(t)}\mathbf{P}_{SG} \tag{4}$$

$$\mathbf{s}^{(t+1)} = \alpha_{US}\mathbf{u}^{(t)}\mathbf{P}_{US} + (1 - \alpha_{US})\mathbf{g}^{(t)}\mathbf{P}_{GS} \tag{5}$$

$$\mathbf{r}^{(t+1)} = \alpha_{UR}\mathbf{u}^{(t)}\mathbf{P}_{UR} + (1 - \alpha_{UR})\mathbf{e}^{(t)}\mathbf{P}_{ER}. \tag{6}$$

In the equations, $\mathbf{u}^{(t)}$, $\mathbf{e}^{(t)}$, $\mathbf{g}^{(t)}$, $\mathbf{r}^{(t)}$, $\mathbf{s}^{(t)}$ are distribution probability vectors representing the probabilities that users, events, groups, regions and subjects are visited at the tth iteration, respectively. The query vector is set as $\mathbf{q}_e^{1\times N}$. For a candidate event e_j, $\mathbf{q}_e(j) = 1$, and we set $\mathbf{q}_e(k) = 0$, if $k \neq j$. Those non-negative αs denote the transition weight from one node type to another. For example, in Eq. (3) event nodes receive α_{UE} probability from user nodes, α_{GE} probability from group nodes, α_{RE} probability from region nodes, and $(1-\alpha_{UE}-\alpha_{GE}-\alpha_{RE})$ probability to restart on the query event node. In this work, we do not train the parameter αs, as the training process is computation intensive and time consuming. Note that the training process should be executed for different cities as they have different constructed graphs. Furthermore, when the graph changes due to node inclusion or deletion, the training process needs to be performed again. In this work, we set the equal coefficients in each equation instead. For example, in Eqs. (2),we set $\alpha_{EU} = \alpha_{GU} = \alpha_{SU} = (1 - \alpha_{EU} - \alpha_{GU} - \alpha_{SU}) = 1/4$.

The RRWR algorithm runs for each of the candidate events and computes all users' proximity values to one candidate event in terms of the steady state probability distribution. For example, for event e_j, after the iteration terminates, a user u_i will be assigned a proximity value $\mathbf{u}_j(i)$ from e_j, where $\mathbf{u}_j$ is the steady state probability vector of event e_j over all users. After repeating these steps for all candidate events, each user will receive the proximity values from all candidate events. Finally, the system can rank the events for user u_i according to $\mathbf{u}_j(i)$ $(1 \leq j \leq N)$ in an

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

decreasing order and choose the first K events to compose his recommendation list.

Compared with the random walk with restart (RWR) algorithm which runs for each user and sets a user node as the query node, our proposed RRWR algorithm runs for each of the upcoming events and sets a candidate event node as the query node. After the termination of random walk, both RWR and RRWR will obtain a user–event proximity matrix $\mathbf{P}_{M\times N}$, where an element p_{ij} indicates the preference score of user u_i to event e_j. According to the proximity matrix, the system can rank the candidate events and generate the recommendation list for each user. The matrix $\mathbf{P}^{RWR}$ obtained by RWR is row normalized, i.e., $\sum_{j=1}^{N} p_{ij} = 1$ for all i, since each row vector corresponds to the steady state probability vector of all candidate events for one user. The matrix $\mathbf{P}^{RRWR}$ obtained by RRWR is column normalized, i.e., $\sum_{i=1}^{M} p_{ij} = 1$ for all j, since a column vector corresponds to the steady state probability vector of all users for one event. Fig. 2 illustrates the two matrices for five users and four events.

4.4. Discussions

Dangling nodes: An event is initialized by a group, however, it is likely that many groups have not organized an event. These group nodes are called *dangling group nodes* while considering group–event connections. When the random walk meets with such dangling group nodes, the iteration process will lead into deviation as these nodes are assumed to connect to all the event nodes. This assumption is necessary to maintain the graph connectivity and to ensure the random walk convergency. It should be noted that the query node can be considered as the source of probability values, which will transfer its probability to other neighboring nodes and can also obtain restarting probability from other nodes at each iteration until the random walk converges. As the probability value of query node maintains larger than $(1 - \alpha_{UE} - \alpha_{GE} - \alpha_{RE})$ at each iteration, so the nodes adjacent to the query node will receive higher transferring probability than farther away nodes. Since these dangling group nodes are closer to the initial query user node in RWR, they will transfer their probabilities equally to all event nodes, which might distort the real interactions in between event and user nodes and cause negative impact on the steady state probability distribution. The RRWR can relieve this problem to some extent, since the source query node is an event node, which is absolutely connected to its originated group node other than many dangling group nodes. Therefore, the dangling nodes in RRWR would receive smaller transferring probabilities than that in RWR and lead to more reliable steady state probabilities.

Popular events: An event node connecting with many other nodes generally implies a *popular event*. In RWR, the random walk and event recommendation are per user basis. It may happen that many users will be assigned very high proximity values for a same popular event. As a result, many users might be recommended to this same popular event, which could affect the recommendation diversity and cause participant capacity problem. In RRWR, the random walk and event recommendation are per event basis. Even for a popular event, the probability value will be distributed to all users, yet each user will receive probability from all events. This helps to reduce the likelihood of assigning a high probability of one event to most users. Take Fig. 2 for example. If each user needs two recommended events, the RWR will recommend the event e_2 to all five users. While in RRWR, the arrangement of users upon candidate events is more balanced.

New events: When a new event is announced and included into the heterogeneous graph, the RWR needs to rerun the random walk for all users so as to update the event probability vector for each user. While the RRWR only needs to rerun the random walk for this new event, and each user will be added this new event proximity value. For example, in Fig. 2, if a new event e_5 is added, to ensure row vector normalized, the RWR needs to run the random walk five times for the five users; While the RRWR only needs to run the random walk once. Furthermore, the number of users is normally much larger than that of events. So the RRWR is more computation efficient.

Convergency analysis: In our graph construction, we propose to connect each dangling node of one type to all the nodes of its destination type, which helps to remove isolated nodes in the constructed heterogeneous graph. An isolated node is such a node, regardless of its type, that does not connect to any other node in the graph. Furthermore, as a result of k-means location clustering, the number of region nodes is less than ten for a typical middle or large city. Each region node actually connects to a large number of user and event nodes. These procedures further help to improve the likelihood of ensuring connectivity for the constructed graph. The following theorem states that if the constructed graph is connected, the proposed RRWR algorithm will converge to a steady state.

Theorem 1. *If the constructed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is connected, the probability vectors in Eqs. (2)–(6) will converge to a unique steady state regardless of the choice of their initial state, as the iteration times tends to infinity.*

Proof. The proof is presented in the Appendix. □

Indeed, our experiments observe the convergence property. We set a threshold to terminate the iteration. That is, when the difference value of two distribution probability vectors in two adjacent iterations is less than the threshold, the probability vectors are regarded as converged. In our experiments, the iteration times is around twenty times in average.

5. Participant scale control algorithms

Event recommendation has a distinctive attribute of *participant capacity* issue. Unlike general item recommendation, such as music, films and etc., where one item can be recommended to as many as possible users, an event may only take a limited number of participants due to resource constraints. However, it may happen that a popular event is recommended to more than its participant capacity. Therefore, a participant scale control algorithm is in demand to generate a reasonable recommendation list L with the limitation of both user and event capacity.

For each user u_i, the *user capacity* is set to a same value K, which also denotes the length of the recommendation list. That is, each user is recommended at most K events. On the other hand, for each event e_j, the *event capacity* C_j indicates the largest number of users to host. It should be noted that the capacity of users and events are generally not matched. For example, in an event-based social network which consists of M users and N upcoming events. Let $\mathbf{V}_{M\times N}$ denote the user–event allocation matrix, where $v_{ij} = 1$ indicates that an event e_j is allocated to the user u_i; Otherwise, $v_{ij} = 0$. When only considering user capacity, $M \times K$ elements in $\mathbf{V}$ should be filled with 1. However, while taking event capacity into consideration, $C = \sum_{j=1}^{N} C_j$ entries at most will be selected. Generally, $M \times K > C$. As a result, many events would exceed their respective capacity. On the other hand, when an event e_j with capacity C_j is announced, the host anticipates that at most C_j users will actually attend the event. Yet taking the proportion factor that not all the recommended users might attend the event, we define a new scale coefficient as follows:

$$\rho = \frac{M \times K}{C}. \tag{7}$$

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

	e_1	e_2	e_3	e_4
u_1	0.4	0.3	0.15	0.15
u_2	0.2	0.35	0.1	0.35
u_3	0.1	0.5	0.2	0.2
u_4	0.2	0.4	0.25	0.15
u_5	0.1	0.6	0.2	0.1

(a) Matrix P in RWR.

	e_1	e_2	e_3	e_4
u_1	0.4	0.2	0.1	0.15
u_2	0.2	0.1	0.5	0.25
u_3	0.15	0.3	0.1	0.2
u_4	0.2	0.3	0.1	0.1
u_5	0.05	0.1	0.2	0.3

(b) Matrix P in RRWR.

Fig. 2. An illustration of proximity matrix after random walk.

In this paper, we assume that each event is equally treated. So the anticipated number of users for event e_j is

$$m_j^* = \rho C_j. \tag{8}$$

Given the recommendation list L_i for each user u_i, let $\mathbb{I}(e_j \in L_i)$ denote the indicator function of whether the event e_j has been recommended to user u_i. Then the recommended number of users for an event e_j is $m_j = \sum_{i=1}^{M} \mathbb{I}(e_j \in L_i)$, and the recommended number of events for an user u_i is $n_i = \sum_{j=1}^{N} \mathbb{I}(e_j \in L_i)$

With the user–event proximity matrix $\mathbf{P}_{M\times N}$, the user capacity K, and the capacity C_j for each event e_j, the participant scale control algorithm can be formulated as the following optimization problem:

$$\text{maximize } S_{all} = \sum_{i=1}^{M}\sum_{j=1}^{N} p_{ij} \times \mathbb{I}(e_j \in L_i), \tag{9}$$

$$\text{subject to : } n_i \le K, \quad i = 1, \ldots, M. \tag{10}$$

$$m_j \le m_j^*, \quad j = 1, \ldots, N. \tag{11}$$

It can be shown that this user–event arrangement problem with capacity constraints is NP-hard. By using $\mathbf{V}$ to denote the user–event allocation matrix, we can transform the above optimization problem to the following zero–one integer programming problem:

$$\text{maximize } S(\mathbf{V}) = \sum_{i=1}^{M}\sum_{j=1}^{N} p_{ij} \times v_{ij}, \tag{12}$$

$$\text{subject to : } \sum_{j=1}^{N} v_{ij} \le K, \quad i = 1, \ldots, M \tag{13}$$

$$\sum_{i=1}^{M} v_{ij} \le m_j^*, \quad j = 1, \ldots, N. \tag{14}$$

Since the zero–one integer programming problem is a well-known NP-hard problem [35], so is the original optimization problem.

We next propose two heuristic algorithms to approximately solve the participant scale control problem. The first one is called greedy arrangement scale control algorithm. We prove that it has a theoretical approximation ratio of $\frac{1}{2}$ to an optimal solution, but it has high computation complexity. The second one is a local greedy adjustment scale control algorithm. Although we are not able to theoretically analyze its approximation ratio, its performance is comparable to the first algorithm according to our experiment results. Besides, its running time is much shorter.

5.1. The greedy arrangement scale control algorithm

The main idea is to, in each iteration, add an unselected user–event pair (u^*, e^*) which has the largest proximity value and has no conflicts with those already selected user–event pairs, into the user–event allocation $\mathbf{V}$. To reduce the search complexity at each iteration, we maintain a heap H to store the candidate pairs having the largest proximity values for each user and event. In each iteration, we select the pair (u^*, e^*) from H instead of from all unselected pairs.

The pseudo-codes of the proposed algorithm are presented in Algorithm 1. The inputs for the algorithm are the user–event proximity matrix $\mathbf{P}_{M\times N}$, user capacity limit K for each user u_i, event capacity limit C_j for each event e_j. Note that an element (u_i, e_j) in the matrix $\mathbf{P}$ is derived from the recommendation algorithm. The output is the user–event allocation matrix $\mathbf{V}_{M\times N}$.

The algorithm first computes the scale coefficient ρ and the anticipated number of users m_j^* for each event e_j (line 1). Since each user–event pair (u_i, e_j) only needs to be pushed into the heap H once, we use a flag matrix $\mathbf{F}$ to mark whether a user–event pair (u_i, e_j) has been pushed into the heap H, which is initialized to zero. (line 2). We next initialize the heap H by selecting the user–event pairs with the largest proximity value for each user and for each event (line 3–line 12). After the initialization step, the main while loop is to iteratively remove a pair (u_i, e_j) from H with the largest proximity value and add it into the matrix $\mathbf{V}$, if it does not introduce user capacity and event capacity conflicts with those already selected pairs in $\mathbf{V}$ (line 13 to line 25). In the main loop, if (u_i, e_j) is added into V, we then decrease the capacity of u_i and e_j by one (line 17). If a user u_i or an event e_j still has remaining capacity, the next unvisited user–event pair with the largest proximity value is pushed into H (line 19 to line 24). The iteration terminates, when $H = \emptyset$. Finally, the algorithm returns the user–event allocation matrix $\mathbf{V}$ (line 26).

Note that to generate the recommendation list, we need to sort the selected user–event pairs according to their proximity values in a decreasing order for each user. We next provide a theoretical analysis for the approximation ratio of our algorithm to an optimal solution.

Lemma 2. *Let $\mathbf{V}_{OPT}$ denote the optimal user–event allocation matrix, which maximizes $S(\mathbf{V}_{OPT})$. For the user–event allocation matrix $\mathbf{V}$ returned by our greedy arrangement algorithm, the following holds:*

$$S(\mathbf{V}) \ge \frac{1}{2} S(\mathbf{V}_{OPT}). \tag{15}$$

Proof. For any pair $(u, e) \in \mathbf{V}$, i.e., $\mathbf{V}(u_i, e_j) = 1$, either $\mathbf{V}_{OPT}(u_i, e_j) = 1$, or $\mathbf{V}_{OPT}(u_i, e_j) = 0$. There is at most one user–event pair (u_i, e') that is selected in $\mathbf{V}_{OPT}$ but unselected in $\mathbf{V}$ due to the selection of (u_i, e_j). This is because (u_i, e_j) occupies one capacity of user u_i. Similarly, there is at most one pair (u', e_j) that is selected in $\mathbf{V}_{OPT}$ but unselected in $\mathbf{V}$ due to the selection of (u_i, e_j), because (u_i, e_j) also occupies one capacity of event e_j. As the proposed algorithm greedily chooses the user–event pair with the highest proximity value, it holds that $\mathbf{P}(u_i, e_j) \ge \mathbf{P}(u_i, e')$ and $\mathbf{P}(u_i, e_j) \ge \mathbf{P}(u', e_j)$. Hence there are at most two user–event pairs that are selected in $\mathbf{V}_{OPT}$ but unselected in $\mathbf{V}$ because of (u_i, e_j). As $\mathbf{P}(u_i, e_j)$ is larger than each of the proximity value of these two pairs, thus it is larger than the mean value of them. That is, $\mathbf{P}(u_i, e_j) \ge \frac{1}{2}(\mathbf{P}(u', e_j) + \mathbf{P}(u_i, e'))$. It should be noted that, for any $(u', e') \in \mathbf{V}_{OPT} \setminus \mathbf{V}$, there must be at least one user–event pair (u_i, e') or $(u', e_j) \in \mathbf{V} \setminus \mathbf{V}_{OPT}$ that is "responsible for" the unselection of

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

Algorithm 1 The greedy arrangement scale control algorithm

Input: The proximity matrix $\mathbf{P}_{M\times N}$, user capacity limit K for each user u_i, the event capacity C_j for each event e_j.
Output: ° The user–event allocation matrix $\mathbf{V}_{M\times N}$
1: Compute $n_i = K$, $\rho = \frac{M\times K}{\sum_{j=1}^{N} C_j}$, $m_j = \lceil \rho C_j \rceil$
2: Initialize $H \leftarrow \emptyset$, $\mathbf{V}_{M\times N} = \mathbf{0}$, $\mathbf{F}_{M\times N} = \mathbf{0}$
3: **for** $i = 1 : M$ **do**
4: $e_{j^*} \leftarrow \arg\max_{e_j} \{\mathbf{P}(u_i, e_j) | \mathbf{F}(u_i, e_j) = 0\}$
5: Push (u_i, e_{j^*}) into H; Set $\mathbf{F}(u_i, e_{j^*}) = 1$
6: **end for**
7: **for** $j = 1 : N$ **do**
8: $u_i^* \leftarrow \arg\max_{u_i} \{\mathbf{P}(u_i, e_j) | \mathbf{F}(u_i, e_j) = 0\}$
9: **if** $(u_i^*, e_j) \notin H$ **then**
10: Push (u_{i^*}, e_j) into H; Set $\mathbf{F}(u_{i^*}, e_j) = 1$
11: **end if**
12: **end for**
13: **while** $H \neq \emptyset$ **do**
14: Extract (u_i, e_j) pair with the largest proximity value in H
15: **if** $n_i > 0$ **and** $m_j > 0$ **then**
16: $v_{ij} = 1$
17: $n_i = n_i - 1$, $m_j = m_j - 1$
18: **end if**
19: **if** $n_i > 0$ **and** $\exists e_{j^*} \leftarrow \arg\max_{e_j} \{\mathbf{P}(u_i, e_j) | \mathbf{F}(u_i, e_j) = 0\}$ **then**
20: Push (u_i, e_{j^*}) into H; Set $\mathbf{F}(u_i, e_{j^*}) = 1$
21: **end if**
22: **if** $m_j > 0$ **and** $\exists u_{i^*} \leftarrow \arg\max_{u_i} \{\mathbf{P}(u_i, e_j) | \mathbf{F}(u_i, e_j) = 0\}$ **then**
23: Push (u_{i^*}, e_j) into H; Set $\mathbf{F}(u_{i^*}, e_j) = 1$
24: **end if**
25: **end while**
26: return $\mathbf{V}$

(u', e') in $\mathbf{V}$; Otherwise, (u', e') would be added into $\mathbf{V}$. Thus we can obtain that:

$$S(\mathbf{V}) = S(\mathbf{V} \cap \mathbf{V}_{OPT}) + S(\mathbf{V} \setminus \mathbf{V}_{OPT}) \tag{16}$$

$$= S(\mathbf{V} \cap \mathbf{V}_{OPT}) + \sum_{(u,e)\in \mathbf{V}\setminus\mathbf{V}_{OPT}} P(u, e) \tag{17}$$

$$\geq S(\mathbf{V} \cap \mathbf{V}_{OPT}) + \frac{1}{2} \sum_{(u',e')\in \mathbf{V}_{OPT}\setminus\mathbf{V}} P(u', e') \tag{18}$$

$$= S(\mathbf{V} \cap \mathbf{V}_{OPT}) + \frac{1}{2} S(\mathbf{V}_{OPT} \setminus \mathbf{V}) \tag{19}$$

$$\geq \frac{1}{2}(S(\mathbf{V} \cap \mathbf{V}_{OPT}) + S(\mathbf{V}_{OPT} \setminus \mathbf{V})) \tag{20}$$

$$= \frac{1}{2} S(\mathbf{V}_{OPT}). \tag{21}$$

In the above deduction, for an user–event pair (u', e') that is selected in $\mathbf{V}_{OPT}$ but unselected in $\mathbf{V}$, there may be more than one user–event pair $(u, e) \in \mathbf{V} \setminus \mathbf{V}_{OPT}$ that is responsible for that. Hence we repeatedly consider some entries in $\mathbf{V}_{OPT} \setminus \mathbf{V}$. In addition, we reduce the common part $S(\mathbf{V} \cap \mathbf{V}_{OPT})$ by $\frac{1}{2}$, which again decreases the approximation ratio of $S(\mathbf{V})$, unless $\mathbf{V} \cap \mathbf{V}_{OPT} = \emptyset$. Therefore, a approximation ratio of $\frac{1}{2}$ can be obtained. □

Complexity analysis. For the greedy arrangement scale control algorithm, the initialization step takes $O(MN)$ computations. The iterations can be conducted $M \times N$ times in the worst case. In each iteration, the computation of extracting a pair from the heap H is at most $O(M + N)$, the computation of finding next candidate pair to push into H is $O(M) + O(N)$ in the worst case. As a result, the time complexity of the algorithm is $O(MN) + MN(O(M + N) + O(M) + O(N))$, which can be reduced as $O(MN(M + N))$.

5.2. The greedy adjustment scale control algorithm

The greedy arrangement algorithm first obtains the user–event allocation matrix and then generates the recommendation lists. In the greedy adjustment algorithm, we first recommend each user K events according to the decreasing order of their proximity values. This obtains an initial user–event allocation matrix, which, however, may breach the event capacity constraint. We next eliminate the conflicts by greedily transferring some users in overstaffed events to those events with spare capacities. In each step, we try to minimize the loss of $S(\mathbf{V})$ due to such adjustment.

Algorithm 2 The greedy adjustment scale control algorithm

Input: The proximity matrix $\mathbf{P}_{M\times N}$, the recommendation list $L_{M\times K}$, the participant capacity C_j for each event e_j.
Output: ° The modified recommendation lists L
1: Compute $\mathcal{U}_j = \{u_i | e_j \in L_i\}$
2: Compute $\rho = \frac{M\times K}{\sum_{j=1}^{N} C_j}$
3: **for** $j = 1 : N$ **do**
4: **while** $|\mathcal{U}_j| \geq \rho C_j + 1$ **do**
5: $min = 1$; $i^* = 0$; $j^* = 0$;
6: **for** $i = 1 : M$ **do**
7: **if** $e_j \in L_i$ **then**
8: $max = 0$; $k^* = 0$;
9: **for** $k = 1 : N$ **do**
10: **if** $|\mathcal{U}_k| < \rho C_k$ **then**
11: **if** $e_k \notin L_i$ and $p_{ik} > max$ **then**
12: $max = p_{ik}$; $k^* = k$;
13: **end if**
14: **end if**
15: **end for**
16: **if** $k^* \neq 0$ **then**
17: **if** $p_{ij} - p_{ik^*} < min$ **then**
18: $i^* = i$; $j^* = k^*$; $min = p_{ij} - p_{ik^*}$
19: **end if**
20: **else**
21: **if** $p_{ij} < min$ **then**
22: $i^* = i$; $j^* = k^*$; $min = p_{ij}$;
23: **end if**
24: **end if**
25: **end if**
26: **end for**
27: $\mathcal{U}_j = \mathcal{U}_j \setminus \{u_{i^*}\}$; $L_{i^*} = L_{i^*} \setminus \{e_j\}$
28: **if** $j^* \neq 0$ **then**
29: $\mathcal{U}_{j^*} = \mathcal{U}_{j^*} \bigcup \{u_{i^*}\}$; $L_{i^*} = L_{i^*} \bigcup \{e_{j^*}\}$;
30: **end if**
31: **end while**
32: **end for**
33: return L

The pseudo-codes of the proposed algorithm are presented in Algorithm 2. The input of the algorithm is the user–event proximity matrix $\mathbf{P}$, the initial recommendation list L_i for each user u_i, the participant capacity C_j for each event e_j. The output is the modified recommendation list L_i for all users.

The algorithm first computes the set of recommended users $\mathcal{U}_j$ for each event e_j (line 1) and the scale coefficient ρ (line 2). The inner while loop is to adjust the number of recommended users for each event not to exceed its participant capacity. For each event e_j, if the recommended users are more than its capacity, for all its recommended users, we find the next feasible event that still lacks users (line 9–15), then we count the reduction of $S(\mathbf{V})$ by removing each recommended user of event e_j to the next feasible event (line 16–24). We record the user i^* and the next feasible event j^* to minimize the reduction of $S(\mathbf{V})$ (line 18,22). The event e_j then removes the user u_{i^*}, and user u_{i^*} removes event e_j (line 27). Then the user u_{i^*} includes the new event e_{j^*}, and the event e_{j^*} includes the user u_{i^*} if this new event exists(line 28–30). We repeat these steps until all the events have been traversed. Then the final arrangement satisfies the capacity of both users and events. It should be noted that in some extreme cases a removed user may not be recommended to a new event due to the conflicts. If this happens, we simply no longer allocate new event to this removed user.

Compared with the greedy arrangement algorithm, we cannot provide the approximation ratio of the greedy adjustment algorithm. According to our experiment results in the next section, this algorithm achieves close performance to the greedy arrangement algorithm, and its computation time is much smaller.

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

Table 1
Statistics of the dataset.

	Atlanta	Houston	Chicago	San Francisco	Los Angeles
Users	10 618	9 872	13 279	16 890	7930
Events	8 734	10 414	11 547	6 551	7672
Groups	2 806	2 071	3 228	5 855	4917
Tags	7 514	7 175	11 387	10 560	9747

Complexity analysis. For the greedy adjustment scale control algorithm, the computation cost to initialize L recommendation lists for all the M users is $M \times O(N \log N)$. In the worst case, all the M users are recommended to the same K events, and these events only has one capacity, which means $(M \times K - K)$ times of adjustments should be conducted to eliminate the conflicts. Furthermore, the time cost for each iteration in the while loop is $O(MN)$. As K is a small constant, thus the overall computation cost of the algorithm is $O(MN \log N) + O(M) \times O(MN)$, which can be reduced as $O(MN(M + \log N))$.

6. Experiment results

6.1. Dataset description

Our dataset is based on the work of Liu et al. 2012 [8], which is crawled from meetup.com from Oct 2011 to Jan 2012. The dataset contains about 4 million users, 2 million events, 70 thousand online groups and 80 thousand tags. In Meetup, social events are created by specifying when, where and what the event is. Users can express their intents to attend an event through RSVP choices. In order to facilitate online interaction and event propagation, users can form online groups to share comments, photos and event plans. Three kinds of data is available in the Meetup dataset, including geographic location of users and events, tags possessed by users and groups, as well as the connections between users, events and groups. In our experiment, we extract the dataset in five typical large cities, namely, Atlanta, Houston, Chicago, San Francisco and Los Angeles, in USA to test our algorithms. We first preprocess data according to the user and event locations, that is, we only consider the local event and local participant within the geographical domain of each city. Furthermore, we filter out the users that have not attended an event and also the events that have no participants. The data statistics of the five cities are presented in Table 1.

For all the five cities, we adopt the 5-fold cross validation method for performance assessment. In each city, we first randomly divide the dataset into 5 equal parts. Let one part of the data be the probe set at a time, the remaining parts as training set to construct the heterogeneous graph model. It should be noted that the dataset partition is based on the user–event attendance pairs, instead of the division according to individual users or events. For each user in the testing set, the system will produce a recommendation list consisting of K events in the decreasing order of user–event proximity value. The user–event pair in the training set, as known information, would not be recommended. In the process of scale control, the system will additionally consider the user and event capacity. As the event capacity information is not available in the dataset, we assume that an event capacity is in proportion to the number of users who have registered to attend this event. Note that in each test set, we only recommend to the users who have participated in at least three events. This process is conducted for all the five data parts to obtain the average values of performance metrics.

6.2. Performance metrics

We first adopt four standard evaluation metrics: P@n (Precision at Position n), MAP (Mean Average Precision), AUC (Area Under ROC Curve) and Recall. Furthermore, we also adopt the coverage index and propose a new S-Pearson metric to evaluate the proposed scale control algorithm. For a user u_i in the testing set, let L_i denote his recommendation list and K the list length. Let $\mathcal{H}_i$ and $\mathcal{B}_i$ denote the set of events that user u_i has attended and the set of events it has not attended, respectively.

P@n and MAP are widely used in ranking problems. P@n computes the percentage of relevant items in top n positions of a recommendation list, ignoring items ranked lower than n. They are used to assess the effectiveness of the recommendation list.

$$P@n = \frac{\sum_{i=1}^{M}\sum_{j=1}^{n}\mathbb{I}(L_i^{(j)} \in \mathcal{H}_i)}{M \times n}, \tag{22}$$

where $\mathbb{I}(\cdot)$ is an indicator function and $L_i^{(j)}$ the jth event in the user u_i's recommendation list. Precision measures the overall hit ratio of the recommendation list, corresponding to the special case of P@n when all events in the recommendation list are considered.

For a user u_i, the average precision AP_i is defined as follows,

$$AP_i = \frac{\sum_{n=1}^{K} P@n \cdot \mathbb{I}(L_i^{(n)} \in \mathcal{H}_i)}{|\mathcal{H}_i|}, \tag{23}$$

where K is the length of recommendation list. $L_i^{(n)}$ denotes the nth event in the ranking event list L_i. $|\mathcal{H}_i|$ represents the number of events joined by u in the test sets. Furthermore, we can obtain MAP by averaging AP_i for all users.

AUC measures the overall results of classification. It is suitable for highly imbalanced dataset and implicit feedback, as in our case where the negative user–event pairs takes a high proportion. We adopt AUC to measure the performance of the system.

$$AUC = \frac{\sum_{i=1}^{M}\sum_{e_j \in \mathcal{H}_i}\sum_{e_k \in \mathcal{B}_i}\mathbb{I}(p_{ij} > p_{ik})}{\sum_{i=1}^{M}|\mathcal{H}_i||\mathcal{B}_i|}, \tag{24}$$

where p_{ij} is the proximity value between user u_i and event e_j in matrix $P_{M\times N}$. $\mathbb{I}(.)$ is also an indicator function that equals to 1 when $p_{ij} > p_{ik}$, otherwise 0.

Since real users are usually concerned only with the top part of the recommendation list, a more practical approach is to consider the number of a user's relevant objects ranked in the top-n places. Recall is one of the most popular metrics based on this. For a target user u_i, $R_i(L)$ is defined as:

$$R_i(L) = \frac{d_i(L)}{|\mathcal{H}_i|} \tag{25}$$

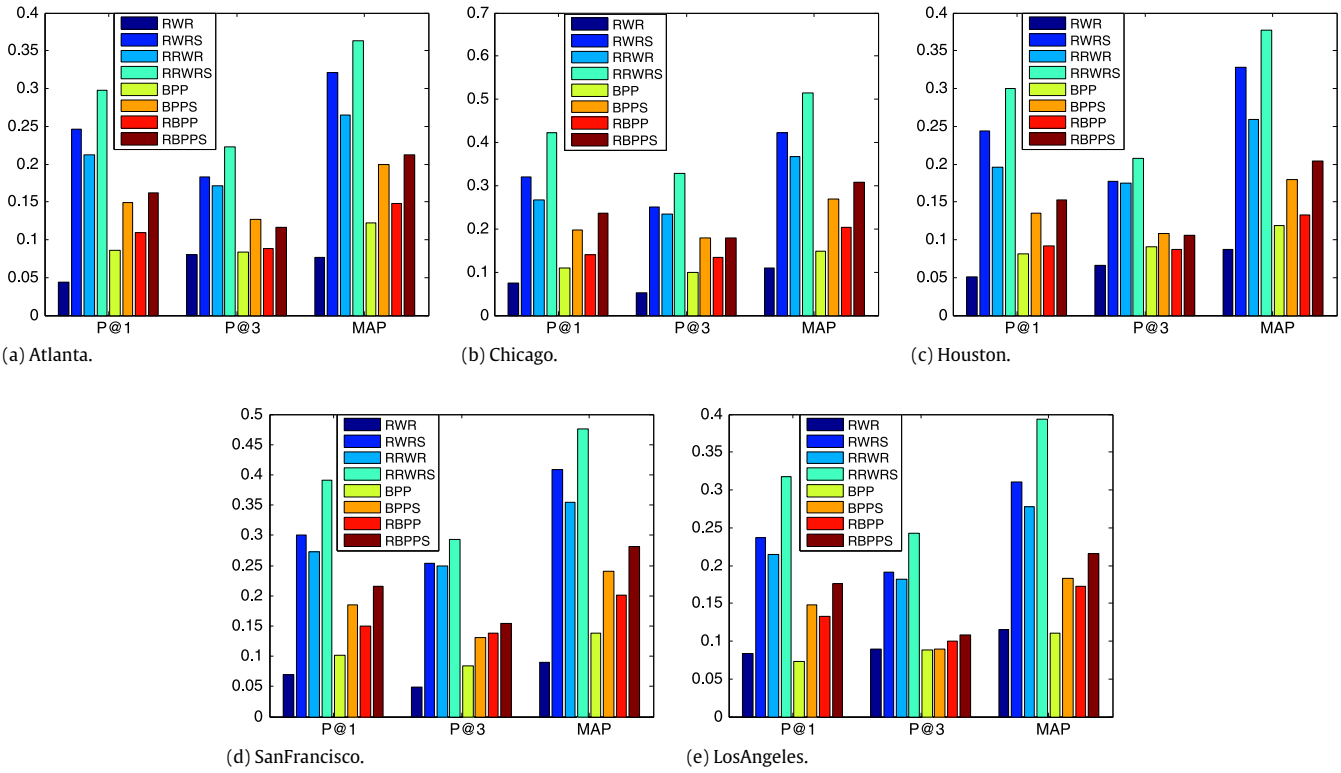
where $d_i(L)$ indicates the number of u_i's relevant events in the top-n places of the recommendation list L_i, and $|\mathcal{H}_i|$ is the total number of u_i's relevant events. Averaging the individual recall over all users with at least one relevant event, we obtain the mean recall, $R(L)$. In order to combine the Recall and Precision result, the $F_1(L)$ metric is used:

$$F_1(L) = \frac{2PR}{P + R}, \tag{26}$$

where P and R are Precision and Recall metric, respectively. They are dependent on the length of the recommendation list.

Coverage measures the proportion of events that the system can recommend to users. In the experiment, if an event appears in the recommendation list of at least a user, it is considered as an

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045



(a) Atlanta. (b) Chicago. (c) Houston.

(d) SanFrancisco. (e) LosAngeles.

Fig. 3. Experiment results of P@n and MAP of different cities.

available event, otherwise not. In this way, the coverage metric can be calculated as follows:

$$COV = \left| \bigcup_{i=1}^{M} L_i \right| / N. \tag{27}$$

To evaluate the distribution of users among the candidate events, we propose the S-Pearson metric, which measures the ability of the system to recommend proper number of relevant users to the upcoming events. C_j is the capacity of candidate event e_j. Y_j records the number of relevant users recommended to the event e_j. The S-Pearson metric counts the Pearson correlation between the event capacity vector ($\mathbf{Z}^{1\times N}$) and the recommended relevant user vector of events ($\mathbf{Z_1}^{1\times N}$), in which $\mathbf{Z}(j) = C_j$, $\mathbf{Z_1}(j) = Y_j$:

$$S - Pearson = \frac{\sum_{k=1}^{N} (\mathbf{Z}(k) - \overline{\mathbf{Z}})(\mathbf{Z_1}(k) - \overline{\mathbf{Z}}_{\mathbf{1}})}{\sqrt{\sum_{k=1}^{N} (\mathbf{Z}(k) - \overline{\mathbf{Z}})^2} \sqrt{\sum_{k=1}^{N} (\mathbf{Z_1}(k) - \overline{\mathbf{Z}}_{\mathbf{1}})^2}}, \tag{28}$$

where $\overline{\mathbf{Z}}$ is the average value of $\mathbf{Z}$. S-Pearson values in $[-1, 1]$, and high value of it indicates that the system is effective to distribute proper people to the candidate events.

6.3. Experiment result

We compare our methods with the following state-of-art methods.

- *RWR*: This method is conducted on our constructed heterogeneous graph, it is the uni-HeteRS algorithm in [31].
- *BPP*: This method is categorized as path-dependent method on the heterogeneous graph, which counts the proximity of nodes according to the weighted sum of the paths that connect them [36]. It adopts the breadth-first strategy to search the valid paths.

Table 2
AUC results.

	RWR	RRWR	BPP	RBPP
Atlanta	0.8811	0.9796	0.9133	0.9489
Chicago	0.9289	0.9926	0.9638	0.9829
Houston	0.8937	0.9826	0.8958	0.9279
San Francisco	0.9197	0.9851	0.9246	0.9579
Los Angeles	0.9035	0.9798	0.9182	0.9495

- *R-BPP*: R-BPP is path-dependent BPP algorithm with the idea of reverse path.
- *Δ-S*: This is Δ method added with the greedy adjustment scale control algorithm. For example, BPP-S is BPP algorithm added with the process of greedy adjustment algorithm.

Table 2 compares the AUC results for the four algorithms. Note that the scale control algorithm does not impact on the proximity matrix, as well as the AUC results. It is first observed that RRWR is better than RWR in terms of high AUC values in all cities. This is because the reverse random walk can help to relieve the dangling node problem. We also observe that RRWR is better than RBPP. The reason lies in that the RRWR method also considers the indirect correlations between nodes, compared to RBPP which only considers the relation of nodes within five steps.

Table 3 presents the coverage results of different algorithms. The coverage of RRWR has increased two to four times larger than that of RWR. As explained in Section 4, the RWR method is very sensitive to popular events, leading to a popular event being recommended to many users. Since the total $M \times K$ times of events will be recommended, so fewer candidate events could be recommended by RWR. For RRWR, this problem can be relieved because its proximity matrix is column normalized, limiting the recommendation of popular events. From the table, we can see that RRWR also performs better than the path-dependent BPP and RBPP methods. This is because they lack the process of probability normalization, leading to the low recommendation scores for less

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

Table 3
Coverage results.

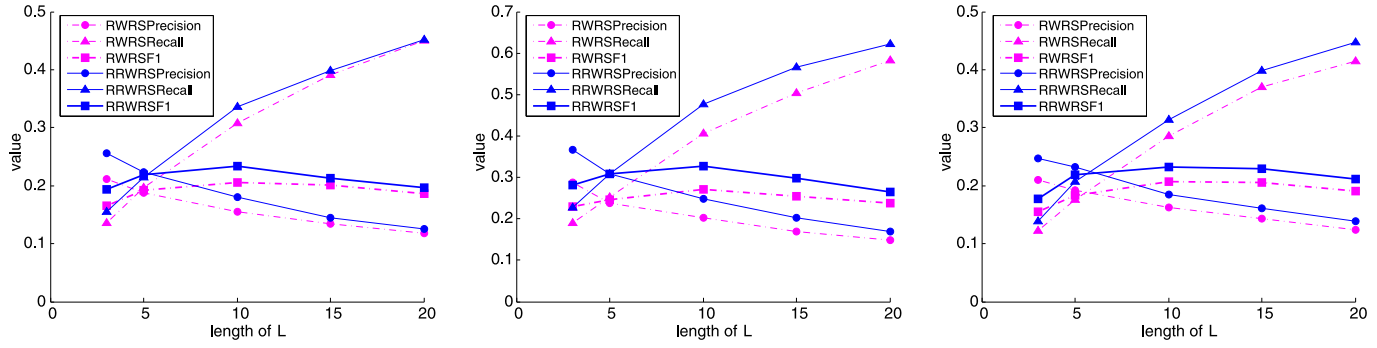
	RWR	RWR-S	RRWR	RRWR-S	BPP	BPP-S	RBPP	RBPP-S
Atlanta	0.0663	0.3693	0.301	0.4361	0.2208	0.4107	0.264	0.4389
Chicago	0.0764	0.3591	0.2955	0.385	0.2124	0.3704	0.2611	0.387
Houston	0.087	0.4294	0.2719	0.4715	0.2062	0.457	0.2402	0.476
San Francisco	0.072	0.3588	0.3148	0.4072	0.2217	0.3789	0.2916	0.4079
Los Angeles	0.12	0.4131	0.3134	0.4767	0.2233	0.4633	0.2713	0.4817

Table 4
S-Pearson results.

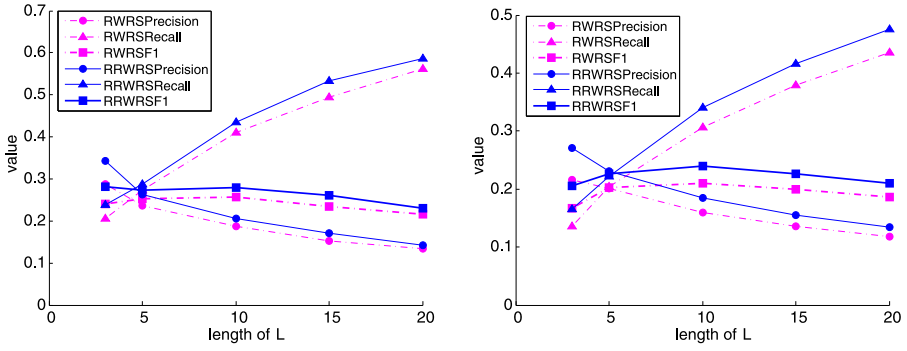
	RWR	RWR-S	RRWR	RRWR-S	BPP	BPP-S	RBPP	RBPP-S
Atlanta	0.1036	0.3048	0.1269	0.2487	0.0788	0.1663	0.0534	0.1467
Chicago	0.1387	0.3379	0.2423	0.3432	0.1239	0.2497	0.1468	0.2338
Houston	0.0691	0.2703	0.1108	0.2064	0.0599	0.1523	0.0559	0.1302
San Francisco	0.1265	0.3739	0.219	0.3422	0.126	0.2325	0.1398	0.2111
Los Angeles	0.0459	0.1694	0.0716	0.1678	0.0261	0.049	0.03	0.0591

Table 5
Two approximation scale control method.

		p@1	p@3	MAP	recall	S-Pearson	coverage	Time(s)
Atlanta	RRWR-S1	0.2973	0.2226	0.3635	0.1602	0.2487	0.4361	8.0132
	RRWR-S2	0.2923	0.24	0.3619	0.1609	0.2296	0.435	931.6402
Chicago	RRWR-S1	0.4229	0.3276	0.5146	0.233	0.3432	0.385	6.6982
	RRWR-S2	0.4038	0.3328	0.4923	0.2247	0.3335	0.3868	482.7547
Houston	RRWR-S1	0.3001	0.2078	0.377	0.1397	0.2064	0.4715	10.8297
	RRWR-S2	0.2959	0.2078	0.3653	0.1406	0.1989	0.4719	1746.1
San Francisco	RRWR-S1	0.3914	0.2925	0.4762	0.2481	0.3422	0.4072	5.2109
	RRWR-S2	0.4172	0.2667	0.4943	0.2352	0.3456	0.4059	146.0007
Los Angeles	RRWR-S1	0.3178	0.2429	0.3932	0.1701	0.1678	0.4767	3.9609
	RRWR-S2	0.3161	0.2579	0.3777	0.1681	0.1715	0.4756	303.9302



(a) Atlanta. (b) Chicago. (c) Houston.



(d) SanFrancisco. (e) LosAngeles.

Fig. 4. Experiment results of Precision, Recall and F_1 against the length of recommendation list.

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

popular events to some extent. As a result, some less popular events might not be recommended.

In addition, we can observe that the scale control algorithm promotes the coverage results evidently for all algorithms. For example, RRWR-S improves about thirty to seventy percents compared with RRWR. This validates the effectiveness of the scale control algorithm which can balance the arrangements of users among candidate events. A valuable recommendation system should offer personalized recommendation to different users and all events should have the equal opportunity to be recommended. Since popular events are easily accessible to most users, recommending them is less valuable than other interesting but less popular events. We believe that high coverage is important to an event recommendation system.

Table 4 presents the S-Pearson results. An algorithm with a high S-Pearson value indicates that most candidate event can recruit proper number of relevant users. We can observe that the Δ-S is better than Δ algorithm. This again validates that the proposed scale control method is good at making reasonable global arrangements for all events.

Fig. 3 plots P@1, P@3 and MAP results of three cities. It can be observed that in all of the three cities, RRWR-S outperforms all the other algorithms, followed by RWR-S. For example, the MAP result of RRWR-S has improved by more than thirty to forty percents over RRWR, and three to four times over RWR. This is because the reverse walk and scale control adopted in RRWR-S help to relieve the negative impacts of dangling nodes. For the path-dependent methods, they cannot thoroughly exploit all connections in between nodes in the graph due to the step limitation. As a result, their performance is not comparable with the proposed reverse random walk method.

Fig. 4 plots the precision, recall and F_1 against the length of recommendation list K. We only consider the RWR-S and RRWR-S algorithm because they perform the best and the second best as discussed before. From the results, we can observe that (1) With the increase of K, the precision is descending; while the recall is increasing, which means more relevant events appear in L. However, the hit ratio is getting worse. (2) The F_1 metric firstly increases with K, at the point of ten, it will achieve the peak. When K continues to increase, the value of F_1 tends to decrease. Therefore, the optimal recommendation list length can be set to ten events from our experiment results.

Finally, we compare the performance of the proposed two scale control algorithms. We only compare them on top of the proposed RRWR method, because the RRWR performs the best as discussed before. We denote RRWR-S1 as the RRWR algorithm with the greedy adjustment, and RRWR-S2 the RRWR algorithm with greedy arrangement scale control. We mainly compare the performance metrics that are impacted by the two scale control algorithms, as presented in Table 5. We can observe that the on all performance metrics the two algorithms perform quite similar, except the running time. The running time of RRWR-S1 is much smaller than that of RRWR-S2. Recall that the RRWR-S2 needs some additional memory to maintain a heap H to store the candidate pairs and a flag for each user–event pair to mark its status in order to avoid repeated access. Therefore, we suggest to adopt the RRWR-S1 for practical event recommendation system. However, since the theoretical approximation ratio of RRWR-S1 has not been obtained, its performance still needs to be explored on some extreme conditions.

7. Conclusion

In this paper, we have proposed a novel event recommendation scheme for event-based social networks. We have proposed to construct a heterogeneous graph to represent the entities and relations in a social network and then to conduct a reverse random walk with restart to obtain the user–event proximity values from the constructed graph. Since in practice events may have participant capacity constraint, we have further proposed two heuristic algorithms for event participant scale control to balance the arrangements of users among upcoming events. Experiment results on Meetup dataset have shown that the proposed RRWR-S algorithm outperforms the state-of-art algorithms in terms of higher AUC values, better recommendation precision and larger recommendation coverage. In our future work, we will explore more advanced methods to deal with tag clustering and to process dangling nodes. Besides, more efforts will be made to optimize the scale control algorithm on extreme conditions.

Acknowledgments

This work is supported in by National Natural Science Foundation of China (Grant No: 61371141) and National Social Science Foundation of China (Grant No: 14CXW018).

Appendix

Proof of Theorem 1. We firstly rewrite Eqs. (2)–(6) as follows:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} M + \mathbf{q} \tag{29}$$

where

$$\mathbf{w}^{(t)} = (\mathbf{u}^{(t)}, \mathbf{e}^{(t)}, \mathbf{g}^{(t)}, \mathbf{s}^{(t)}, \mathbf{r}^{(t)}) \tag{30}$$

where $\mathbf{u}^{(t)}$, $\mathbf{e}^{(t)}$, $\mathbf{g}^{(t)}$, $\mathbf{s}^{(t)}$, $\mathbf{r}^{(t)}$ are row probability vectors of users, events, groups, subjects and regions at iteration t, respectively.

$$\mathbf{q} = (\mathbf{0}, \alpha_{EE}\mathbf{q}_e, \mathbf{0}, \mathbf{0}, \mathbf{0}) \tag{31}$$

The overall transition matrix M is defined as follows:

$$\begin{bmatrix} \mathbf{0} & \alpha_{UE}P_{UE} & \alpha_{UG}P_{UG} & \alpha_{US}P_{US} & \alpha_{UR}P_{UR} \\ \alpha_{EU}P_{EU} & \mathbf{0} & \alpha_{EG}P_{EG} & \mathbf{0} & \alpha_{ER}P_{ER} \\ \alpha_{GU}P_{GU} & \alpha_{GE}P_{GE} & \mathbf{0} & \alpha_{GS}P_{GS} & \mathbf{0} \\ \alpha_{SU}P_{SU} & \mathbf{0} & \alpha_{SG}P_{SG} & \mathbf{0} & \mathbf{0} \\ \alpha_{RU}P_{RU} & \alpha_{RE}P_{RE} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\alpha_{RU} = 1 - \alpha_{EU} - \alpha_{GU} - \alpha_{SU}$$
$$\alpha_{EE} = 1 - \alpha_{UE} - \alpha_{GE} - \alpha_{RE}$$
$$\alpha_{SG} = 1 - \alpha_{UG} - \alpha_{EG}$$
$$\alpha_{GS} = 1 - \alpha_{US}$$
$$\alpha_{ER} = 1 - \alpha_{UR}.$$

Note that $P_{MN}(M, N \in \mathcal{V})$ are stochastic matrices where the sum of each row equals to 1. However, the matrix M is not a stochastic matrix. By extracting the parameters in M, we can obtain the matrix Λ as follows:

$$\Lambda = \begin{bmatrix} \mathbf{0} & \alpha_{UE} & \alpha_{UG} & \alpha_{US} & \alpha_{UR} \\ \alpha_{EU} & \mathbf{0} & \alpha_{EG} & \mathbf{0} & \alpha_{ER} \\ \alpha_{GU} & \alpha_{GE} & \mathbf{0} & \alpha_{GS} & \mathbf{0} \\ \alpha_{SU} & \mathbf{0} & \alpha_{SG} & \mathbf{0} & \mathbf{0} \\ \alpha_{RU} & \alpha_{RE} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{32}$$

Λ^T is an irreducible sub-stochastic matrix where each element is non-negative and every row adds up at most to 1 with at least one row having sum less than 1. The following Lemma 3 prove that the spectral radius (maximum of the absolute eigenvalues) of Λ^T is strictly less than 1. The same holds for Λ due to the same eigenpolynomial. By the way, Λ^T is an irreducible matrix (there is always a path from one node to another). According to Perron–Frobenius Theorem, there exists a positive vector $\mathbf{z} = (z_U, z_E, z_G, z_S, z_R)^T$, where $\Lambda\mathbf{z} = \lambda\mathbf{z}$. In the meanwhile, we note that $P_{MN} \times \mathbf{1}_{|N|} =$

$\mathbf{1}_{|M|}$, where $\mathbf{1}_M$ is a M-dimensional column vector with all ones. Thus we can obtain that $M(z_U\mathbf{1}_{|U|}, z_E\mathbf{1}_{|E|}, z_G\mathbf{1}_{|G|}, z_S\mathbf{1}_{|S|}, z_R\mathbf{1}_{|R|}) = \lambda(z_U\mathbf{1}_{|U|}, z_E\mathbf{1}_{|E|}, z_G\mathbf{1}_{|G|}, z_S\mathbf{1}_{|S|}, z_R\mathbf{1}_{|R|})$. Hence λ is also an eigenvalue of matrix M and the corresponding eigenvector is positive. Since the matrix M is the transition matrix of a connected graph, thus it is a non-negative and irreducible matrix. Based on Perron–Frobenius theorem, the only eigenvectors whose components are all positive are those associated with the spectral radius r. As a result, r is the spectral radius of matrix M. That is $\rho(M) = \lambda < 1$.

Given the initial status of the graph $w^{(0)} = \pi$, we can calculate that, $w^{(1)} = \pi M + q$, $w^{(2)} = \pi M^2 + qM + q, \ldots, w^{(t)} = M^t\pi + \sum_{k=0}^{t-1} M^k q$. As the spectral radius of M is less than 1, when t goes to infinity, $M^t = 0$ and $\sum_{k=0}^{t-1} M^k = (I - M)^{-1}$. So $w^{(t)}$ finally converges to $w^* = (I - M)^{-1}q$. □

Lemma 3. *The spectral radius of an irreducible sub-stochastic matrix is strictly less than* 1.

Proof. Suppose A is an irreducible sub-stochastic matrix and λ is the Perron–Frobenius eigenvalue of A (i.e. $\rho(A) = \lambda$) with v the corresponding eigenvector normalized such that $\|v\|_1 = 1$. By the Perron–Frobenius theorem for irreducible non-negative matrices, the entries of v must be positive. Using this, we have the following.

$$\begin{aligned}|\lambda| &= \|\lambda v\|_1 = \|vA\|_1 \\ &= \|vA\|_1 \\ &= \Sigma_j \Sigma_k v_j A_{jk}.\end{aligned}$$

Let $\epsilon_j = \frac{1}{N}(1 - \sum_{k=1}^{N} A_{jk})$. If we add ϵ_j to each element of the jth row of A, the row sum will become one. Let ϵ be the row vector containing the values of $\{\epsilon_j\}$.

$$\begin{aligned}|\lambda| &= \Sigma_j \Sigma_k v_j (A_{jk} + \epsilon_j - \epsilon_j) \\ &= \Sigma_j \Sigma_k v_j (A_{jk} + \epsilon_j) - \Sigma_j \Sigma_k v_j \epsilon_j \\ &= \|v(A + \epsilon^T\mathbf{1})\|_1 - N(\epsilon \cdot v).\end{aligned}$$

We define $\hat{A} = A + \epsilon^T\mathbf{1}$ and note that it is a proper stochastic matrix. Since v is positive and ϵ is non-negative, we have:

$$\begin{aligned}|\lambda| &= \|v\hat{A}\|_1 - N(\epsilon \cdot v) \\ &= 1 - N(\epsilon \cdot v) \\ &< 1. \quad \square\end{aligned}$$

References

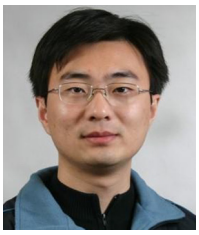
[1] D.T. Nguyen, J.E. Jung, Real-time event detection for online behavioral analysis of big social data, Future Gener. Comput. Syst. 66 (2017) 137–145.
[2] W. Zhang, J. Wang, W. Feng, Combining latent factor model with location features for event-based group recommendation, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 910–918.
[3] L. Lü, M. Medo, C.H. Yeung, Y.-C. Zhang, Z.-K. Zhang, T. Zhou, Recommender systems, Phys. Rep. 519 (1) (2012) 1–49.
[4] Y.-C. Sun, C.C. Chen, A novel social event recommendation method based on social and collaborative friendships, in: Social Informatics, Springer, 2013, pp. 109–118.
[5] H. Yin, Y. Sun, B. Cui, Z. Hu, L. Chen, Lcars: a location-content-aware recommender system, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 221–229.
[6] J. Bao, Y. Zheng, D. Wilkie, M. Mokbel, Recommendations in location-based social networks: a survey, GeoInformatica 19 (3) (2015) 525–565.
[7] Z. Qiao, P. Zhang, Y. Cao, C. Zhou, L. Guo, B. Fang, Combining heterogenous social and geographical information for event recommendation, in: Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014, pp. 145–151.
[8] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, J. Han, Event-based social networks: linking the online and offline social worlds, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 1032–1040.
[9] H. Wang, M. Terrovitis, N. Mamoulis, Location recommendation in location-based social networks using user check-in data, in: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2013, pp. 374–383.
[10] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (1) (2004) 5–53.
[11] M.J. Pazzani, D. Billsus, Content-based recommendation systems, in: The Adaptive Web, Springer, 2007, pp. 325–341.
[12] Z. Liu, W. Qu, H. Li, C. Xie, A hybrid collaborative filtering recommendation mechanism for p2p networks, Future Gener. Comput. Syst. 26 (8) (2010) 1409–1417.
[13] H. Chen, X. Cui, H. Jin, Top-k followee recommendation over microblogging systems by exploiting diverse information sources, Future Gener. Comput. Syst. 55 (2016) 534–543.
[14] G. Liao, Y. Zhao, S. Xie, P.S. Yu, An effective latent networks fusion based model for event recommendation in offline ephemeral social networks, in: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, ACM, 2013, pp. 1655–1660.
[15] C.C. Chen, Y.-C. Sun, Exploring acquaintances of social network site users for effective social event recommendations, Inform. Process. Lett. 116 (3) (2016) 227–236.
[16] V.W. Zheng, Y. Zheng, X. Xie, Q. Yang, Collaborative location and activity recommendations with gps history data, in: Proceedings of the 19th International Conference on World Wide Web, ACM, 2010, pp. 1029–1038.
[17] R. Du, Z. Yu, T. Mei, Z. Wang, Z. Wang, B. Guo, Predicting activity attendance in event-based social networks: Content, context and social influence, in: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 2014, pp. 425–434.
[18] C. Cheng, H. Yang, I. King, M.R. Lyu, Fused matrix factorization with geographical and social influence in location-based social networks, in: Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012, pp. 17–23.
[19] H. Wang, M. Terrovitis, N. Mamoulis, Location recommendation in location-based social networks using user check-in data, in: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2013, pp. 374–383.
[20] D. Quercia, N. Lathia, F. Calabrese, G. Di Lorenzo, J. Crowcroft, Recommending social events from mobile phone location data, in: 2010 IEEE International Conference on Data Mining, IEEE, 2010, pp. 971–976.
[21] B. Xu, H. Zhuge, An angle-based interest model for text recommendation, Future Gener. Comput. Syst. 64 (2016) 211–226.
[22] R. Krestel, P. Fankhauser, W. Nejdl, Latent dirichlet allocation for tag recommendation, in: Proceedings of the 3rd ACM Conference on Recommender Systems, ACM, 2009, pp. 61–68.
[23] A.S.R.W.W. Umbrath, L. Hennig, A hybrid plsa approach for warmer cold start in folksonomy recommendation, in: Proceedings of Recommender Systems and the Social Web, ACM, 2009, pp. 10–13.
[24] F. Zhao, Y. Zhu, H. Jin, L.T. Yang, A personalized hashtag recommendation approach using lda-based topic model in microblog environment, Future Gener. Comput. Syst. 65 (2016) 196–206. Special issue on Big data in the cloud.
[25] Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Personalized recommendation via integrated diffusion on user-item-tag tripartite graphs, Physica A (2010) 179–186.
[26] S. Rendle, L. Balby Marinho, A. Nanopoulos, L. Schmidt-Thieme, Learning optimal ranking with tensor factorization for tag recommendation, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 727–736.
[27] P. Symeonidis, User recommendations based on tensor dimensionality reduction, in: Artificial Intelligence Applications and Innovations III, 2009, pp. 331–340.
[28] S. Lee, S.-i. Song, M. Kahng, D. Lee, S.-g. Lee, Random walk based entity ranking on graph for multidimensional recommendation, in: Proceedings of the Fifth ACM Conference on Recommender Systems, ACM, 2011, pp. 93–100.
[29] L. Backstrom, J. Leskovec, Supervised random walks:predicting and recommending links in social networks, in: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, ACM, 2011, pp. 635–644.
[30] B. Gao, T.-Y. Liu, W. Wei, T. Wang, H. Li, Semi-supervised ranking on very large graphs with rich metadata, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 96–104.
[31] T.A.N. Pham, X. Li, G. Cong, Z. Zhang, A general graph-based model for recommendation in event-based social networks, in: 2015 IEEE 31st International Conference on Data Engineering, IEEE, 2015, pp. 567–578.
[32] J. She, Y. Tong, L. Chen, C.C. Cao, Conflict-aware event-participant arrangement, in: 2015 IEEE 31st International Conference on Data Engineering, IEEE, 2015, pp. 735–746.
[33] J. She, Y. Tong, L. Chen, Utility-aware social event-participant planning, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 1629–1643.
[34] Y. Tong, R. Meng, J. She, On bottleneck-aware arrangement for event-based social networks, in: 2015 31st IEEE International Conference on Data Engineering Workshops, IEEE, 2015, pp. 216–223.

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045

[35] M. Jünger, T.M. Liebling, D. Naddef, G.L. Nemhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, L.A. Wolsey, 50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-art, Springer Science & Business Media, 2009.
[36] Q. Yuan, G. Cong, A. Sun, Graph-based point-of-interest recommendation with geographical and temporal influences, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, ACM, 2014, pp. 659–668.

Yijun Mo obtained the Ph.D. degree in 2008, the M.Phil. degree in 2002 and the B.Eng. (EEE) degree in 1999 from the Department of Electronics and Information Engineering in Huazhong University of Science and Technology (HUST). He is currently working as an associate professor in the same school of HUST. He was a visiting researcher at the Hong Kong University of Science and Technology during 2008. His research interests include the area of wireless network, semantic computing, recommendation systems, social networks, etc.

Bixi Li obtained his B.S. from the Department of Electronics and Information Engineering in Huazhong University of Science and Technology (HUST), Wuhan, China in 2013. He is currently working towards his master degree in the same school. His research interests mainly include recommendation systems and applications.



Bang Wang obtained his B.S. and M.S. from the Department of Electronics and Information Engineering in Huazhong University of Science and Technology (HUST) Wuhan, China in 1996 and 2000, respectively, and his Ph.D. degree in Electrical and Computer Engineering (ECE) Department of National University of Singapore (NUS) in 2004. He is now working as a professor in the School of Electronic Information and Communications, HUST. His research interests include wireless networking issues, recommendation systems and social networks. Dr. Wang had published over 100 technical papers in international conferences and journals.

Laurence T. Yang received the BE degree in Computer Science and Technology from Tsinghua University, China and the Ph.D. degree in Computer Science from University of Victoria, Canada. He is a professor in the School of Computer Science and Technology at Huazhong University of Science and Technology, China, and in the Department of Computer Science, St. Francis Xavier University, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing. His research has been supported by the National Sciences and Engineering Research Council, and the Canada Foundation for Innovation.

Minghua Xu received her B.A. in the School of Journalism and Information Communication of Huazhong University of Science and Technology (HUST) in 2002, and her M.A and Ph.D. in Social Science Department of National University of Singapore (NUS) in 2005 and 2010, respectively. She is now working as an associate professor in the School of Journalism and Information Communication of HUST. Her research interests mainly include theory and practice of information diffusion in social networks, recommendation systems and applications, and information communication theories. Dr. Xu has published over 30 peer reviewed conference and journal papers.

Please cite this article in press as: Y. Mo, et al., Event recommendation in social networks based on reverse random walk and participant scale control, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.045