IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON CYBER-PHYSICAL-SOCIAL COMPUTING AND NETWORKING

Received November 14, 2017, accepted December 18, 2017, date of publication December 25, 2017,
date of current version February 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2786679

SERGE: Successive Event Recommendation Based on Graph Entropy for Event-Based Social Networks

SHENGHAO LIU[1], BANG WANG[1], AND MINGHUA XU[2]
[1]School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China
[2]School of Journalism and Information Communication, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Minghua Xu (xuminghua@hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61771209 and in part by the National Social Science Foundation of China under Grant 14CXW018.

ABSTRACT With the fast development of many *event-based social networks* (EBSNs), event recommendation, which is to recommend a list of upcoming events to a user according to his preference, has attracted a lot of attentions in both academia and industry. In this paper, we propose a *successive event recommendation based on graph entropy* (SERGE) to deal with the new event cold start problem by exploiting diverse relations as well as asynchronous feedbacks in EBSNs. The SERGE creates recommendation lists at discrete times during each publication period. At the beginning, it constructs a *primary graph* (PG) based on the entities and their relations in an EBSN and computes the user-event similarity scores by applying a *random walk with restart* (RWR) algorithm on PG. At each recommendation time, it then constructs a *feedback graph* (FG) based on the up-to-date user feedbacks on event reservations and applies the RWR again on FG to compute new user-event similarity scores. We then propose to weight the two sets of similarity scores with the graph entropies of both PG and FG and create the final recommendation lists accordingly. We have crawled two datasets from a real EBSN for two cities, Beijing and Shanghai in China. Experimental results validate the effectiveness and superiority of the proposed SERGE scheme over the peer schemes.

INDEX TERMS Successive event recommendation, random walk with restart, graph entropy, cold start problem, event-based social networks.

I. INTRODUCTION

Recently, *event-based social networks* (EBSNs) have been widely developed as a convenient cyber-platform to announce various *offline social events* that are hosted by some people and organization for other persons to participate, like concert, hiking and etc., for recruiting interested participants [1]–[5]. With the help of EBSNs, a user can search and reserve his preferred event online and attend his reserved event offline. However, due to the proliferation of online events, searching interested ones becomes burdensome for EBSN users. For example, Meetup currently has 16 million users with more than 300,000 events announced per month [2]. Although some EBSNs service users with the interface to retrieve and rank the events by manually setting search conditions, how to clearly express diverse preferences and quickly pinpoint preferred events are still difficult for most of users. In response to the pressing demands, a good *event recommendation system* is much required for EBSNs.

Event recommendation in EBSNs, though shares some similarities with general item recommendation, like recommendation for books, clothes, friends and etc. [6]–[9], has many distinct features that make it a difficult task and a hot research topic in recent years [10], [11]. First, an offline event possesses unique temporal and spatial characteristics. Generally, an event could not be actually *'consumed'* and evaluated before its commencement, which raises the issue of new event *cold start* problem. In practice, an EBNS often provides a reservation or RSVP interface for a potential participant to reserve his interested events. Also, an offline event normally takes place in some particular locations: While some users like to participate in events near their homes, but some others tend to attend events in their favorite regions [10]. How to process and explore such spatiotemporal factors are of many usefulness in event recommendation.

Event recommendation in EBSNs should also take into consideration of various social relations among users.

Many EBSNs, like **Meetup** and **Douban Event**,[1] allow users to join online groups, track event organizers, and follow other users, by which a complicated online social network is also formed. Generally, users in an online group share some common interests. On the other hand, users who have attended a same offline event are likely to establish some social relations and even become friends. Users with close relations in both the online and offline social networks could have similar preferences towards future events. How to extract and exploit such social relations are of great importance in event recommendation.

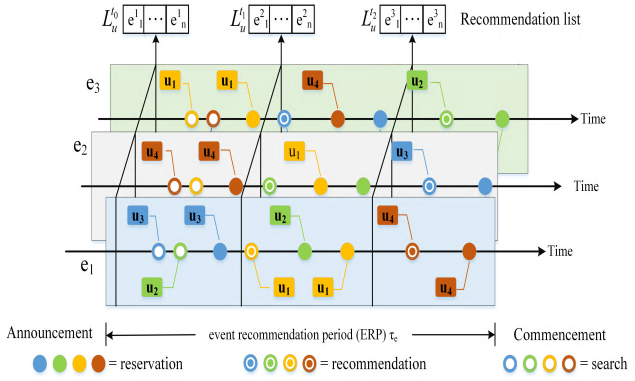


FIGURE 1. Illustration of system recommendations, user accesses and actions. Each event recommendation period τ_e is the time span between its announcement and commencement. The system publishes a batch of new upcoming events at a fixed time interval and creates successive recommendation lists $L_u^{t_k}$ for each user u at different recommendation time t_k in between two publications. Users can access the EBSN at any time; While in each of his access, besides searching events, a user can reserve an event no matter whether the event has been recommended or not.

Event recommendation in EBSNs may experience a different processing procedure, compared with general item recommendations. Fig. 1 illustrates our envisioned recommendation process. For an upcoming event e, an *event recommendation period* (ERP) τ_e can be identified as the time span between its announcement and its commencement. During an ERP τ_e, a user u may not access the EBSN, or access the EBSN one or more times in one day or multiple days. In each of his accesses, the upcoming event e could be recommended to the user u. Whether or not being recommended, the user u could reserve the event e in his access. Notice that even if the event e is not recommended, the user u can still reserve e via his own search. Yet after the reservation, the event e is obviously no longer needed to be recommended. The objective of event recommendation is to create a recommendation list $L_u^{t_k} = (e_1^k, \ldots, e_N^k)$ for user u in his kth access during an ERP, which could contain different events in his different accesses. Generally, $L_u^{t_k}$ contains the top-N recommended events in the order of his reservation likelihood from the highest to the lowest. Notice that between the kth and $(k+1)$th access of user u in τ_e, some other users may have reserved the event e. As these reservations could be made by the friends of user u, they can be regarded as *feedbacks* related to user u when computing $L_u^{t_{k+1}}$. To well exploit the asynchronous users' feedbacks, the platform can modify and update its recommendation lists to a user at his different access times during the EBP. Therefore, successive event recommendation considering feedbacks should be expected for EBSNs.

In this paper, we propose a *successive event recommendation based on graph entropy* (SERGE) for the famous EBSN in China, **Douban Event**. Besides users and events, we first extract the factors that could indicate users' preferences, including the online groups, tags, hosts as well as various event attributes. After some data preprocessing, we next construct a *primary graph* (PG) to capture the characteristics of the extracted entities and their relations. We then apply the *random walk with restart* (RWR) to compute the similarity scores $\boldsymbol{sim}_{t_0}(u)$ between the user u and upcoming events. We notice that in a real EBSN, event announcements and user accesses can happen at any time during a day, which would impact on the structure of the constructed primary graph. For practical implementations, we propose a *coordinated configuration* for the system operator to deal with the problem of asynchronous information update, where the primary graph is constructed and updated at a fixed time interval, say for example, once per day. Since a user who accesses the EBSN in between the two configurations can reserve an event, we also propose to construct a *feedback graph* (FG) which contains only users and events to capture such dynamic relations. We then apply the RWR again on FG to obtain a new set of similarity scores $\boldsymbol{sim}_{t_k}(u)$. Notice that the PG and FG have different structures. The PG provides a more complete description of the EBSN, yet the FG reflects the most updated user-event relations. To strike a balance between the two recommendation results, we propose to use graph entropy for PG and FG to weight the two sets of similarity scores and to compute the final recommendation similarity scores $\boldsymbol{SIM}_{t_k}(u)$ for each user. The recommendation lists are then obtained by ranking events according to their similarity scores in $\boldsymbol{SIM}_{t_k}(u)$. We have crawled two data sets from **Douban Event** for two typical cities: Beijing and Shanghai. Our experiments on the two cities show that the proposed SERGE scheme can achieve better recommendation results, compared with the peer schemes.

Our contributions are summarized as follows:

- Propose the SERGE scheme to deal with the cold start problem and the asynchronous feedback problem;
- Use two graph structures to capture the most important and the most updated factors as well as their relations;
- Compute the graph entropies of the two graphs to weight their respective recommendation results;
- Experiment the SERGE scheme on the real EBSN datasets and validate its superiority over peer schemes.

The rest of the paper is structured as follows: Section II briefly reviews the related work. The proposed SERGE scheme is presented in Section III and experimented in Section IV. The paper is concluded in Section V with some discussions.

[1]Meetup: www.meetup.com; Douban Event: www.douban.com

II. RELATED WORK

We review event recommendation in EBSNs from two main approaches in the literature: classic algorithms and graph-based algorithms. We also review some graph entropy studies.

Classic recommendation algorithms, like *content-based* recommendation (CB), *collaborative filtering* (CF) and their combinations have been proposed to match user preferences with event features in different ways. In CB recommendation, user preferences are normally extracted from the events in which he had participated before. For example, Macedo *et al.* [12] propose a contextual model in which the *term frequency-inverse document frequency* (TF-IDF) has been used to characterize each event. Gu *et al.* [13] propose a context-aware matrix factorization algorithm by using the *Latent Dirichlet Allocation* (LDA) to create event features from their text announcements. In CF recommendation, similarities between users are used to recommend events to a user if his friends or other users with similar preferences had also attended the events [14]–[18]. Li *et al.* [17] propose a CF algorithm that calculates the similarity between users based on three aspects, including user influences on topics, regions and organizers. However, although CF performs well in most cases, its greatest shortcoming is that CF cannot deal with the new event cold start problem owing to the lack of rating records. Some hybrid algorithms combining both CB and CF have also been proposed [19], [20]. Khrouf and Troncy [19] propose a weighted hybridization using a linear combination of recommendation scores which are calculated through CB and CF algorithm, respectively. Hsieh *et al.* [20] propose a user-centric recommendation model that considers both content information and user social relations.

In graph-based algorithms, a graph is constructed to represent the relations of different entities in an EBSN, in which nodes stand for entities and edges for their relations [21]–[27]. The event recommender problem is then converted into a node proximity problem, where the event nodes are ranked with their proximity values to a user node. Some algorithms have been proposed to calculate node proximities, including path-dependent algorithms and random walk with restart (RWR) algorithms. For example, in the RWR algorithms [23]–[28], a Markov chain as well as its transition matrix are first constructed based on the graph. Setting an initial state for all nodes, the nodes' probability distribution iterates through transition matrix and converge to a steady distribution that are used as node proximities.

Most of existing algorithms have not take into consideration of the information quantity in an EBSN. Especially in graph-based algorithms, different graphs can be constructed, yet each could provide different information quantities for their diverse topological structures. In the studies of complex networks, graph entropy has been widely used to compute the information quantity that a graph can provide [29]–[31]. For example, Dehmer [29] introduces a general framework for defining the entropy of a graph based on a local information graph and on information functionals derived from the topological structure of graph. Eagle *et al.* [30] define the node entropy based on its topological diversity and compute the graph entropy as the summation of all nodes' entropies.

In [23], we have applied a graph model for one-time event recommendation and shown that the graph model can well capture different entities and their relations in an EBSN. In this paper, we apply this graph model in our primary graph only. However, much unlike our previous work, the proposed SERGE scheme is a kind of hybrid graph-based successive recommendation algorithms, yet with the novelties of constructing different graphs at different times and weighting their results based on graph entropy.

III. SERGE: SUCCESSIVE EVENT RECOMMENDATION BASED ON GRAPH ENTROPY

A. THE SERGE SCHEME OVERVIEW

In this paper, we propose a *successive event recommendation based on graph entropy* (SERGE) for the very popular Chinese EBSN, Douban Event. We first extract and process the related factors to construct graphs. We note that although the factors and graphs are for Douban Event, the idea and implementation of the proposed SERGE scheme can be easily extended to other EBSNs with slight modifications.

As illustrated in Fig. 1, the working process of SERGE is as follows: An EBSN can collect new events at any time, yet it publishes a batch of new events at a fixed time interval, say for example, at midnight each day. At the publication time t_0, the EBSN can create an initial recommendation list for each user. During the two publications, the EBSN creates K new recommendation lists $t_1, \ldots, t_K$ to respect the fact of asynchronous user accesses and their feedbacks in between two publications. At t_0, the SERGE scheme constructs the primary graph based on the available history information before t_0 to compute a similarity score $\mathbf{sim}_{t_0}(u)$ between user u and upcoming events; While at each t_k $(1 \leq k \leq K)$, it constructs the feedback graph based on the updated user-event relations to compute a new similarity score $\mathbf{sim}_{t_k}(u)$. The SERGE then creates successive recommendation lists $L_{t_k}(u)$ for user u at each recommendation time t_k based on the ranking of the weighted two scores, $\boldsymbol{SIM}_{t_k}(u) = \alpha_{t_k} \boldsymbol{sim}_{t_0}(u) + (1 - \alpha_{t_k}) \boldsymbol{sim}_{t_k}(u)$. Algorithm 1 presents the procedures of the proposed SERGE scheme.

B. DATA PREPROCESSING

In Douban Event, the essential entities include users U, events E, groups G, hosts H, tags T. Furthermore, an event is also described by the following attributes: event time E_m, event location E_l, event cost E_c and event type E_t. These in total nine entities can be used as nodes for graph construction. However, some entities take real numbers, e.g., E_m and E_c; while some other entities may take too many discrete values, e.g., T and E_l. Therefore, we need to first preprocess these entities for reducing graph complexity.

The basic idea of our preprocessing is to use segmentation or aggregation to reduce the parameter value space. For entity event time E_m, we divide the continuous time line

Algorithm 1 The SERGE Scheme

if it is the publication time t_0 **then**
  Obtain the entities and relations in the EBSN
  Build the primary graph PG
  Apply RWR on PG to get $\boldsymbol{sim}_{t_0}(u)$
  Compute the PG entropy $H_{PG}(t_0)$
else
  Update the user-event relation based on feedbacks
  Build the feedback graph FG_{t_k}
  Apply RWR on FG_{t_k} to get $\boldsymbol{sim}_{t_k}(u)$
  Compute the FG entropy $H_{FG}(t_k)$
  Compute weighting coefficient $\alpha_{t_k} = \frac{H_{PG}(t_0)}{H_{PG}(t_0)+H_{FG}(t_k)}$
  Compute $\boldsymbol{SIM}_{t_k}(u) = \alpha_{t_k}\boldsymbol{sim}_{t_0}(u) + (1-\alpha_{t_k})\boldsymbol{sim}_{t_k}(u)$
end if

into seven week days plus one another 'Everyday', i.e., from Monday to Sunday and Everyday, as we argue that people daily life often takes some periodic feature. For entity event cost E_c, we partition its value into five ranges for normal expense habits, i.e., free charge, $1 \sim 200$, $201 \sim 500$, $501 \sim 1000$, and above 1000 Chinese Yuan. For event locations E_l, we use fewer *administrative regions* E_r, each to represent for one event location. Most of event locations also include the administrative region. If an event location does not contain the region information, we use the *nearest neighbor* algorithm to include it into the region with the shortest Euclidean distance to the region center. For tags T, we cluster them into fewer *subjects* by using the *unweighted pair-group method with arithmetic mean* (UPGMA). In each iteration, we group the most similar two clusters or tags into a new one. The iteration terminates, until the required number of clusters has achieved. Note that the intersection of any two tag clusters (i.e., subjects) is an empty set.

C. GRAPH CONSTRUCTION

The primary graph is constructed based on the system information available at t_0, which is a mixed graph containing both directed and undirected edges; The feedback graph is constructed at each t_k to respect the newest user-event relations updated before t_k. The PG structure is more complicated than that of FG, yet the FG contains the most up-to-date information.

1) PRIMARY GRAPH CONSTRUCTION

Most existing graph-based event recommendation algorithms have proposed to construct a *heterogeneous graph* to use all available entities as nodes and their explicit relations as edges [22]. In such a heterogeneous graph, no edge exists in between two entities of the same type. For example, no edge exists in between two events; Instead, an edge only exists to connect an entity from one entity type and another from the other entity type. In [23], we have proposed a new graph structure different from the heterogeneous graph, which can achieve better random walk results. In SERGE, we exploit our previous findings to construct the primary graph.

In the primary graph construction, we allow edges existing in between two events by converting the *explicit relations* in between events and their attributes into the *implicit relations* in between events. The conversion is processed as follows: Let $\mathbf{A}_{EE_c}$, $\mathbf{A}_{EE_m}$, $\mathbf{A}_{EE_r}$ and $\mathbf{A}_{EE_t}$ denote the adjacency matrices of events and event attribute nodes. Let $\mathbf{A}_E$ denote the concatenation matrix of these matrices, where each row represents the attribute vector of one event node. We compute the cosine similarity between two event nodes by

$$sim(E_i, E_j) = \cos(\overrightarrow{A}_i, \overrightarrow{A}_j), \tag{1}$$

where $\overrightarrow{A}_i$ is the ith row vector of $\mathbf{A}_E$. For each event E_i, we select its top K most similar events to establish K directed implicit edges each from E_i to one of its similar event nodes. In this paper, we set $K = 100$. Note that since the sets of similar events may be different of two different events, so we use directed implicit edges. As illustrated in Fig. 2, we convert the local heterogeneous graph of events and their attributes into a single inter-connected directed event graph.

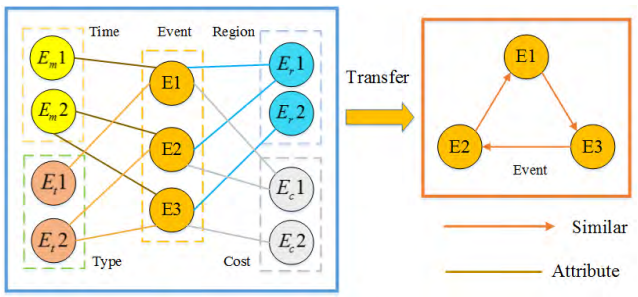


FIGURE 2. Illustration of local event graph construction. The left blue box illustrates a local graph of event nodes and their attribute nodes; While the right orange box illustrates its conversion to a local event graph. We use event attributes to compute the cosine similarity in between two events. For each event, we select its top K most similar events to establish K directed edges each from the event to one of its similar events.

After the local event graph conversion, we construct the primary graph from five entity types, namely, users U, events E, groups G, hosts H and subjects S. The primary graph contains two types of edges: the directed implicit edge in between two events, and undirected explicit edge in between two entities from two different entity types. For example, if a user $U1$ joins an online group $G1$, then an undirected edge connects $U1$ and $G1$; If $U1$ has reserved or attended an offline event $E1$, then an undirected edge exists between them. Note that after the tag clustering, a group, a host or an event can be stamped by one or more subjects, given its tags appearing in how many subjects. The left part of Fig. 3 illustrated a constructed primary graph, where the colored edges in between two entities represent their relations.

For the constructed primary graph, let $\boldsymbol{A}_{MN}$ denote the adjacency matrix of type M nodes and type N nodes, where $\boldsymbol{A}_{MN}(m, n) = 1$ indicates that an explicit or

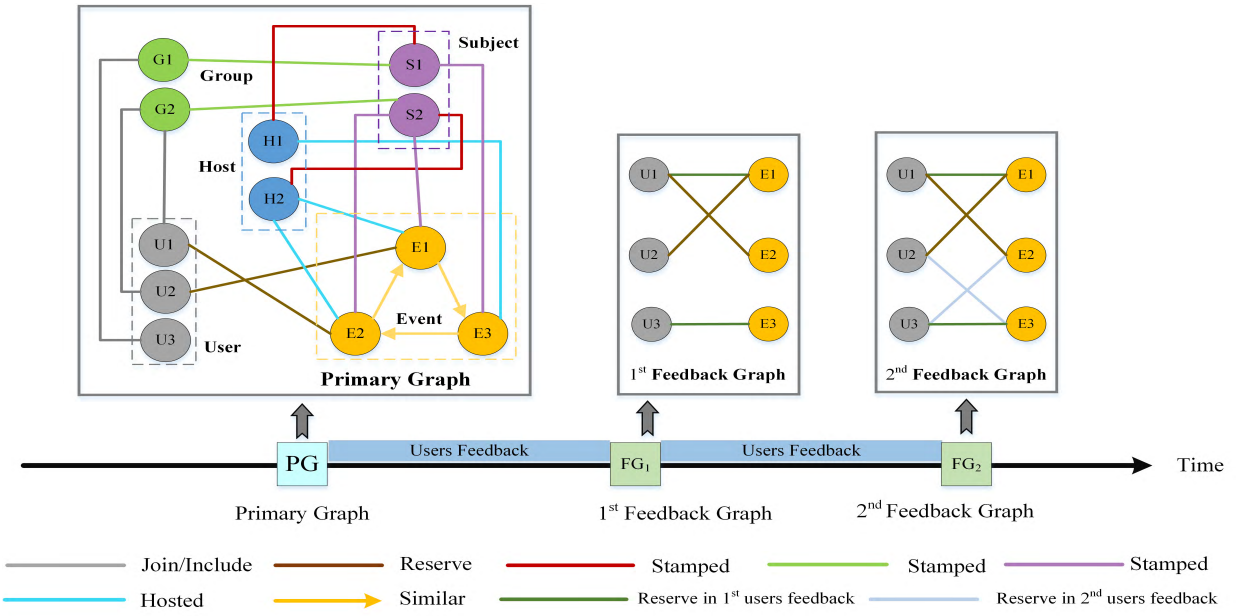


FIGURE 3. Illustration of graph construction in the proposed SERGE scheme. The primary graph is constructed at each event publication time, where different colored nodes represent different entity types in an EBSN, including users *U*, events *E*, groups *G*, hosts *H* and subjects *S*. Note that only the edges in between event nodes are directed; While others are undirected. The feedback graph is constructed at each recommendation time, which only contains user nodes and event nodes. An edge in between a user and an event indicates that the user has reserved the event before that recommendation time.

implicit relation exists between the node n and node m; Otherwise, $\boldsymbol{A}_{MN}(m,n)=0$.

2) FEEDBACK GRAPH CONSTRUCTION

In most of the existing graph-based recommendation schemes [32], the whole graph would be reconstructed to respect each user's update in the EBSN, like joining an online group or reserving an upcoming event. Accordingly, new recommendation lists are computed based on the new graph. However, as the users' feedbacks normally involve with only small changes of the whole graph yet with more up-to-date information, we propose to construct a standalone feedback graph to respect the most related information updated in between users and events during the two recommendation times.

In the feedback graph construction, we only consider two node types, namely, users U and events E, and their relations. Note that the FG does not consider inter-event relations, because the inter-event relations are not the up-to-date information which have been fixed since events have been published. As illustrated in Fig. 3, from t_0 to t_2, some feedbacks regarding whether a user newly reserves an event has been received, like user $U1$ reserving event $E1$, $U2$ reserving $E3$, and $U3$ reserving $E2$ and $E3$. As shown in Fig. 3, the feedback graph is constructed, where the edges $(U1, E2)$ and $(U2, E1)$ are inherited from the previously known user-event relations and other edges are based on the newly updated user-event relations.

For the constructed feedback graph, let $\boldsymbol{A}_{UE_{t_k}}$ and $\boldsymbol{A}_{EU_{t_k}}$ be the adjacency matrix of user and event nodes at time t_k, respectively, They are based on the $\boldsymbol{A}_{UE_{t_0}}$ and $\boldsymbol{A}_{EU_{t_0}}$ at t_0 and updated by users' feedbacks at t_k. For example, if user u_1 confirms to participate in event e_1 during the time period $[t_0, t_k)$, $\boldsymbol{A}_{UE_{t_0}}(u_1, e_1)$ will be updated from 0 to 1 to generate the adjacency matrix $\boldsymbol{A}_{UE_{t_k}}$.

D. RANDOM WALK WITH RESTART ON GRAPH

We use a multivariate Markov chain to transform the event recommendation task into a node convergency probability computation problem. A transition matrix $\boldsymbol{P}_{MN}$ is obtained by row-normalizing the adjacency matrix $\boldsymbol{A}_{MN}$.

1) RWR ON PRIMARY GRAPH

For the primary graph, we let $\boldsymbol{P}_{UE_{t_0}}$ and $\boldsymbol{P}_{EU_{t_0}}$ be the transition matrices of the available user-event relations at time t_0. We define the *user query vector* as $\boldsymbol{q}_u$. For each user u_j, $\boldsymbol{q}_u(i)=1$, if $i=j$; Otherwise, $\boldsymbol{q}_u(i)=0$.

We randomly initialize the probability vector of users, events, groups, hosts and subjects as $\boldsymbol{u}^{(0)}, \boldsymbol{e}^{(0)}, \boldsymbol{g}^{(0)}, \boldsymbol{h}^{(0)}, \boldsymbol{s}^{(0)}$. To obtain the convergency probabilities, the *random walk with restart* (RWR) algorithm is to iteratively compute the following equations:

$$\begin{aligned}\boldsymbol{u}^{(j+1)} &= \alpha_{EU}\boldsymbol{e}^{(j)}\boldsymbol{P}_{EU_{t_0}} + \alpha_{GU}\boldsymbol{g}^{(j)}\boldsymbol{P}_{GU} \\ &\quad + (1-\alpha_{EU}-\alpha_{GU})\boldsymbol{q}_u \qquad (2)\\ \boldsymbol{e}^{(j+1)} &= \alpha_{UE}\boldsymbol{u}^{(j)}\boldsymbol{P}_{UE_{t_0}} + \alpha_{HE}\boldsymbol{h}^{(j)}\boldsymbol{P}_{HE} + \alpha_{SE}\boldsymbol{s}^{(j)}\boldsymbol{P}_{SE} \\ &\quad + (1-\alpha_{UE}-\alpha_{HE}-\alpha_{SE})\boldsymbol{e}^{(j)}\boldsymbol{P}_{EE} \qquad (3)\\ \boldsymbol{h}^{(j+1)} &= \alpha_{EH}\boldsymbol{e}^{(j)}\boldsymbol{P}_{EH} + (1-\alpha_{EH})\boldsymbol{s}^{(j)}\boldsymbol{P}_{SH} \qquad (4)\\ \boldsymbol{g}^{(j+1)} &= \alpha_{UG}\boldsymbol{u}^{(j)}\boldsymbol{P}_{UG} + (1-\alpha_{UG})\boldsymbol{s}^{(j)}\boldsymbol{P}_{SG} \qquad (5)\\ \boldsymbol{s}^{(j+1)} &= \alpha_{GS}\boldsymbol{g}^{(j)}\boldsymbol{P}_{GS} + \alpha_{HS}\boldsymbol{h}^{(j)}\boldsymbol{P}_{HS} \\ &\quad + (1-\alpha_{HS}-\alpha_{GS})\boldsymbol{e}^{(j)}\boldsymbol{P}_{ES} \qquad (6)\end{aligned}$$

Here, $\boldsymbol{u}^{(j)}, \boldsymbol{e}^{(j)}, \boldsymbol{g}^{(j)}, \boldsymbol{h}^{(j)}, \boldsymbol{s}^{(j)}$ are probability vectors representing the probability that user, event, group, host and subject nodes are visited in the jth iteration ($j = 0, 1, ..$), respectively. And α_{MN} denotes the transition weight from one type node to another type node. For example, in Eq. (2) user nodes get α_{EU} probability from event nodes, α_{GU} probability from group nodes, and return to the candidate user node with $(1 - \alpha_{EU} - \alpha_{GU})$ probability. Since the primary graph is of large scale, we do not try to train the weights for the computation complexity considerations. Instead, we set that the weights of each factor of transition probability are equal. For example, in Eq. (2) we set $\alpha_{EU} = \alpha_{GU} = (1 - \alpha_{EU} - \alpha_{GU}) = 1/3$. The iteration terminates until the pairwise difference in between two iteration probability vectors is small than a predefined threshold. It has been proven in [21] that if the constructed graph is a connected one, then the iterations can converge. We note that the constructed primary graph from our data set is a connected one. After the iteration termination, each user u obtains a vector of event convergency probabilities for M upcoming events, denoted by

$$\boldsymbol{sim}_{t_0}(u) = (sim_{t_0}(u, e_1), \ldots, sim_{t_0}(u, e_M)), \tag{7}$$

where each element $sim_{t_0}(u, e_j)$ can be considered as the similarity score between u and e_j at t_0, computed from the primary graph.

2) RWR ON FEEDBACK GRAPH

For the feedback graph at t_k, the transition matrix $\boldsymbol{P}_{UE_{t_k}}$ and $\boldsymbol{P}_{EU_{t_k}}$ are obtained by row-normalizing the adjacency matrix $A_{UE_{t_k}}$ and $A_{EU_{t_k}}$, respectively. Notice that a feedback graph only consists of edges each connecting one user and one event. To obtain the convergency probabilities, we apply the RWR algorithm again by iteratively computing the following equations:

$$\boldsymbol{u}^{(j+1)} = \alpha_{EU}\boldsymbol{e}^{(j)}\boldsymbol{P}_{EU_{t_k}} + (1 - \alpha_{EU})\boldsymbol{q}_u \tag{8}$$

$$\boldsymbol{e}^{(j+1)} = \boldsymbol{u}^{(j)}\boldsymbol{P}_{UE_{t_k}} \tag{9}$$

Here we set $\alpha_{EU} = 1/2$. After the iteration termination, each user u obtains a vector of event convergency probabilities, denoted by

$$\boldsymbol{sim}_{t_k}(u) = (sim_{t_k}(u, e_1), \ldots, sim_{t_k}(u, e_M)), \tag{10}$$

where each element $sim_{t_k}(u, e_j)$ can be considered as the similarity score between u and e_j at t_k computed from the feedback graph.

E. RECOMMENDATION BASED ON GRAPH ENTROPY

The similarity score $\boldsymbol{sim}_{t_0}$ is computed from the primary graph based on all the available information at t_0 about all entities and their relations in an EBSN. If we create recommendation lists from $\boldsymbol{sim}_{t_0}$, we might be able to deal with the cold start problem by exploiting the indirect relations between users and events. The similarity score $\boldsymbol{sim}_{t_k}$ is computed from the feedback graph based only on the most up-to-date user-event relations at t_k. If we create recommendation lists from $\boldsymbol{sim}_{t_k}$, we might be able to exploit the newest feedbacks. In order to make full use of all the information from both graphs, we propose to weight the two similarity scores to create recommendation lists. Except the time t_0, the weighted similarity score at t_k ($i = 1, 2, ..$) is computed by

$$\boldsymbol{SIM}_{t_k}(u) = \alpha_{t_k}\boldsymbol{sim}_{t_0}(u) + (1 - \alpha_{t_k})\boldsymbol{sim}_{t_k}(u), \tag{11}$$

where α_{t_k} is the weighting coefficient at t_k. The recommendation list L_{t_k} is then obtained based on the ranking of the similarity score $\boldsymbol{SIM}_{t_k}$ for each user.

As the primary graph and feedback graph contain different amount of information and with different structures, we propose to compute α_{t_k} based on the *graph entropy*, which has been widely used in complex networks to capture the structural information quantity of a graph [29]. The weighting coefficient α_{t_k} is computed by

$$\alpha_{t_k} = \frac{H_{PG}(t_0)}{H_{PG}(t_0) + H_{FG}(t_k)}, \tag{12}$$

where $H_{PG}(t_0)$ and $H_{FG}(t_k)$ represent the quantity of information in the primary graph at time t_0 and that in the feedback graph at time t_k, respectively. We argue that the more information a graph contains, the higher weight it should take. In this paper, we adopt the graph entropy computation in [30]. It first computes the node entropy in a graph based on its topological diversity information:

$$h(i) = -\sum\nolimits_{j=1}^{N} r_{ij}\log(r_{ij}) \tag{13}$$

where r_{ij} is the transition probability of node n_i to node n_j in the graph. Note that when computing $h(i)$, we only consider the topological structure of whole graph without differentiating node types. That is, each user, event, group, host and subject are treated simply as a single node in this whole graph. So unlike using multiple adjacent matrices $\boldsymbol{A}_{MN}$ to describe the connections in between different node types, we use a single adjacent matrix $\boldsymbol{B}$ to describe the topological structure of the whole primary graph. In $\boldsymbol{B}$, an element $b_{ij} = 1$ indicates an edge from node i to j; Otherwise, $b_{ij} = 0$ in the whole primary graph. Based on $\boldsymbol{B}$, $r_{ij} = 1/\boldsymbol{b}_{i\cdot}$, where $\boldsymbol{b}_{i\cdot}$ is the ith row vector of $\boldsymbol{B}$. Then the graph entropy of the primary graph is the summation of all nodes' entropy

$$H_{PG} = \sum_{i=1}^{N_{PG}} h(i), \tag{14}$$

where N_{PG} is the total number of nodes in the primary graph. The entropy of feedback graphs is computed in the same way.

We use Fig. 3 to illustrate entropy computation. For PG and 2^{nd} FG, we take one common node $U1$ to illustrate its node entropy computation. In PG, $U1$ connects to one group node and one event node, so the entropy $h_{PG}(U1) = -(0.5\log\ (0.5) + 0.5\log\ (0.5)) = 1$. In 2^{nd} FG, although $U1$ dose not connect to any group node, its entropy equals to $h_{2^{nd}FG}(U1) = -(0.5\log\ (0.5) + 0.5\log\ (0.5)) = 1$ owing to the connection to a new event node $E1$. In the same way, all nodes' entropy can be computed and the graph entropy is the

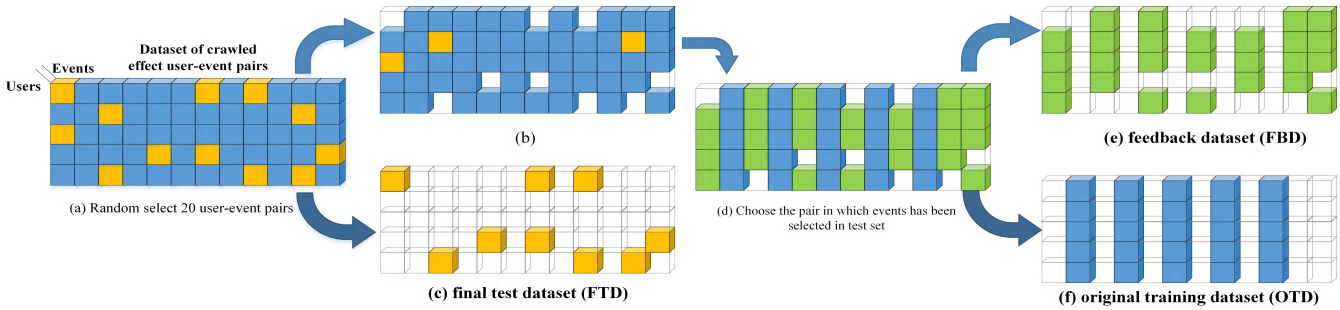


FIGURE 4. Illustration of dataset arrangement. Each colored block indicates a user-event pair. From the crawled user-event pairs, 20 percent of them are randomly selected and among these 20 percent user-event pairs, only those users have reserved at least three events are chosen as the final test dataset (c), colored by orange. The others are further split into the original training dataset (f), colored by blue, and the feedback dataset (e), colored by green.

summation of all nodes' entropy. In Fig. 3, the graph entropy of the PG and 2^{nd} FG equal to 16.32 and 6, respectively.

IV. EXPERIMENT RESULTS

A. EXPERIMENT DATASETS

We have crawled datasets from Douban Event for two main cities, Beijing and Shanghai, in China. For Beijing, we obtained 6982 events and 88963 users from Jul 1st, 2015 to Dec 31st, among which in total 80153 effective user-event pairs are used to compose Beijing dataset. For Shanghai, we obtained 6427 events and 75829 users from Sep 1st, 2015 to Dec 31st, among which in total 67822 effective user-event pairs are used to compose Shanghai dataset. Table 1 summarizes the statistics of the two datasets.

TABLE 1. Statistics of dataset.

Data	User	Event	Group	Host	Tag	UE-Pair
Beijing	88963	6982	253	2008	6509	80153
Shanghai	75829	6427	253	1770	5138	67822

As we were not able to obtain the details about the users' accesses to Douban Event, such as the time of each access and action, we decide to divide the original datasets (OD) into three parts to emulate users' feedbacks during the recommendation process. The three datasets, namely, *original training dataset* (OTD), *feedback dataset* (FBD) and *final test dataset* (FTD), are composed as follows. As illustrated in Fig. 4 (a), we first randomly extract 20% user-event pairs (colored as orange), from which the users who have reserved at least 3 events as well as their reserved events to form the FTD as illustrated by Fig. 4 (c). Those user-event pairs in OD but not in FTD, called unselected user-event pairs as illustrated Fig. 4 (b), are then used to compose the OTD and FBD. As illustrated by Fig. 4 (d), from unselected user-event pairs, those user-event pairs whose events have also been selected in FTD are selected to compose the FBD, as illustrated by Fig. 4 (e). All the other unselected user-event pairs compose the OTD, as illustrated by Fig. 4 (f). Furthermore, we randomly divide the FBD into ten equal subsets, each representing the user feedback at the ten recommendation times. Note that all the feedback subsets available before the recommendation t_k can also be used for the recommendation at t_k.

B. EVALUATION METRICS

We adopt four traditional evaluation metrics: P@n (Precision at Position n), MAP (Mean Average Precision), Recall and F1. For a user u_i ($i = 1, \ldots, M$) in the final test set, let L_i denote his recommendation list and N the list length. Let $\mathcal{H}_i$ denotes the set of events that user u_i has actually attended.

Both P@n and MAP are commonly used in ranking problems. P@n computes the percentage of correctly recommended events in top n positions of a recommendation list, ignoring events ranked lower than n, to assess the recommendation effectiveness.

$$P@n = \frac{\sum_{i=1}^{M}\sum_{j=1}^{n}\mathbb{I}(L_i^{(j)} \in \mathcal{H}_i)}{M \times n}, \tag{15}$$

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,

$$AP_i = \frac{\sum_{n=1}^{N} P@n \cdot \mathbb{I}(L_i^{(n)} \in \mathcal{H}_i)}{|\mathcal{H}_i|}, \tag{16}$$

where N 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 actually attended by u_i in the final test set. The MAP is obtained by averaging AP_i over all users.

In practice, users are often concerned only with the top part of the recommendation list. Recall evaluates how many a user's actually attended events ranking in the top-n places. For a user u_i, his recall $R_i(L)$ is defined by

$$R_i(L) = \frac{d_i(L)}{|\mathcal{H}_i|} \tag{17}$$

where $d_i(L)$ indicates the number of u_i's attended events in the top-n places of the recommendation list L_i, and $|\mathcal{H}_i|$ the

total number of u_i's attended events. The mean recall, denoted by *Recall* is obtained by averaging the individual recall over all users with at least one relevant event.

The $F1$ metric is used to evaluate the joint effectiveness of the Recall and Precision:

$$F1 = \frac{2PR}{P+R}, \tag{18}$$

where P and R are Precision and Recall metric, respectively.

TABLE 2. The graph entropy and the weight coefficient at different recommendation times.

	Beijing		Shanghai	
Time	Entropy ($\times 10^3$)	α_{t_k}	Entropy ($\times 10^3$)	α_{t_k}
t_0	$H(PG_{t_0}) = 83.2$	1	$H(PG_{t_0}) = 69.3$	1
t_1	$H(FG_{t_1}) = 9.01$	0.90	$H(FG_{t_1}) = 8.33$	0.89
t_2	$H(FG_{t_2}) = 10.8$	0.89	$H(FG_{t_2}) = 9.75$	0.88
t_3	$H(FG_{t_3}) = 12.4$	0.87	$H(FG_{t_3}) = 11.0$	0.86
t_4	$H(FG_{t_4}) = 13.8$	0.86	$H(FG_{t_4}) = 12.2$	0.85
t_5	$H(FG_{t_5}) = 15.2$	0.84	$H(FG_{t_5}) = 13.7$	0.84
t_6	$H(FG_{t_6}) = 16.5$	0.83	$H(FG_{t_6}) = 14.6$	0.83
t_7	$H(FG_{t_7}) = 17.9$	0.82	$H(FG_{t_7}) = 15.7$	0.81
t_8	$H(FG_{t_8}) = 19.2$	0.81	$H(FG_{t_8}) = 16.8$	0.80
t_9	$H(FG_{t_9}) = 20.4$	0.80	$H(FG_{t_9}) = 17.9$	0.79
t_{10}	$H(FG_{t_{10}}) = 21.7$	0.79	$H(FG_{t_{10}}) = 18.9$	0.79

C. EXPERIMENT RESULTS

For the SERGE scheme, Table 2 summarizes the intermediate results of the graph entropies (H_{PG} and H_{FG}) and weighting coefficient (α) at different recommendation times for both Beijing and Shanghai in one-fold of experiments. Other folds have similar values. It is not unexpected to observe that H_{PG} of the primary graph is larger than H_{FG} of those feedback graphs due to its complex topological structure. Yet with the increase of more feedbacks, the graph entropy H_{FG} increases, and accordingly, the value of weighting coefficient α_{t_k} decreases. This well reflects the design objective of the proposed SERGE algorithm: At t_0, the upcoming events are regarded as a cold start problem, as no users have made reservations. So we mainly rely on the history information to construct the primary graph consisting of all available entities and their relations without prioritization. Later on, as feedbacks about user reservations are available due to the asynchronous user accesses and actions on the EBSN, we not only exploit such the newest and mostly related information about the relations of users and upcoming events for reliving the cold start problem, but also gradually prioritize their importance for creating the final recommendation lists.

We next compare the proposed SERGE scheme with the following state-of-the-art schemes:

- **CB**: It is the content-based recommendation which applies a user history preference feature to compute the cosine similarity with the event attribute vector. The preference feature is computed from the events that the user has attended before.
- **AllPG**: At each recommendation time t_k, the system builds a graph with the same structure of the primary graph yet with all the system information as well as the newest feedbacks before t_k. It then applies the RWR algorithm on this graph to create recommendation lists.
- **AllPG + CB**: This algorithm can be divided into two parts. The first part is the same as the AllPG at each t_k to create a candidate recommendation list. The second part then applies the CB algorithm above based on a user history preference feature as well as the most up-to-date user feedback to compute new cosine similarities for the events in the candidate list. Finally we will combine the result of AllPG and the result of CB to rerank the candidate recommendation list.
- **OnlyFG**: At each t_k, it creates recommendation lists based on RWR on a feedback graph constructed in the same way as that in our **SERGE** algorithm.
- **OnlyFG + EE**: It works the same process as the OnlyFG algorithm, except that the feedback graph at each t_k also includes the implicit relations among events.

We use the five-fold cross validation to obtain the averaged results for these schemes.

Fig. 5 presents the experiment results for Beijing. In all results, the x-axis is the recommendation times from t_0 to t_{10}. We first observe that the proposed **SERGE** scheme outperforms the other peer schemes in terms of all performance metrics and at almost all recommendation times, yet with only a few of slight degradations. It is also observed that the **CB** performs the worst among all schemes at almost all recommendation times. This is because the **CB** does not take into consideration of the various online and offline social relations in an EBSN; While other graph-based schemes exploit such relations by constructing various graphs as well as applying the random walk algorithm on graphs.

From Fig. 5, we observe that the **AllPG + CB** slightly outperforms the **SERGE** at the recommendation time t_0. Recall that at t_0 the cold start problem exists, as no user has reserved any upcoming event. So at t_0, the **SERGE** scheme is the same as the **AllPG** with the same experiment results. On the other hand, at t_0 the **AllPG + CB** actually makes a sequential recommendation by using the **CB** scheme on top of the **AllPG** scheme. Such repeated exploitation of the history user-event relations leads to its slight improvement over the **SERGE** at t_0, which could be attributed to its prioritization of different information types by sequentially combining two algorithms. This can also be observed by comparing only the **AllPG** and **AllPG + CB** at other recommendation times. However, as observing the results for other recommendation times t_k ($k = 1, \ldots, 10$), the **AllPG + CB** performs much worse than the **SERGE**, which indicates that the naive exploitation of the newest feedbacks in the **AllPG + CB** is inferior to that of **SERGE**.

From Fig. 5, when having a close look of the **AllPG**, it is a bit surprising to find that it performs much worse than the **SERGE** in all performance metrics at all recommendation times, except at t_0. Recall that the **AllPG** exploits all the available system information including the newest user feedbacks before t_k to construct a primary graph at t_k; While

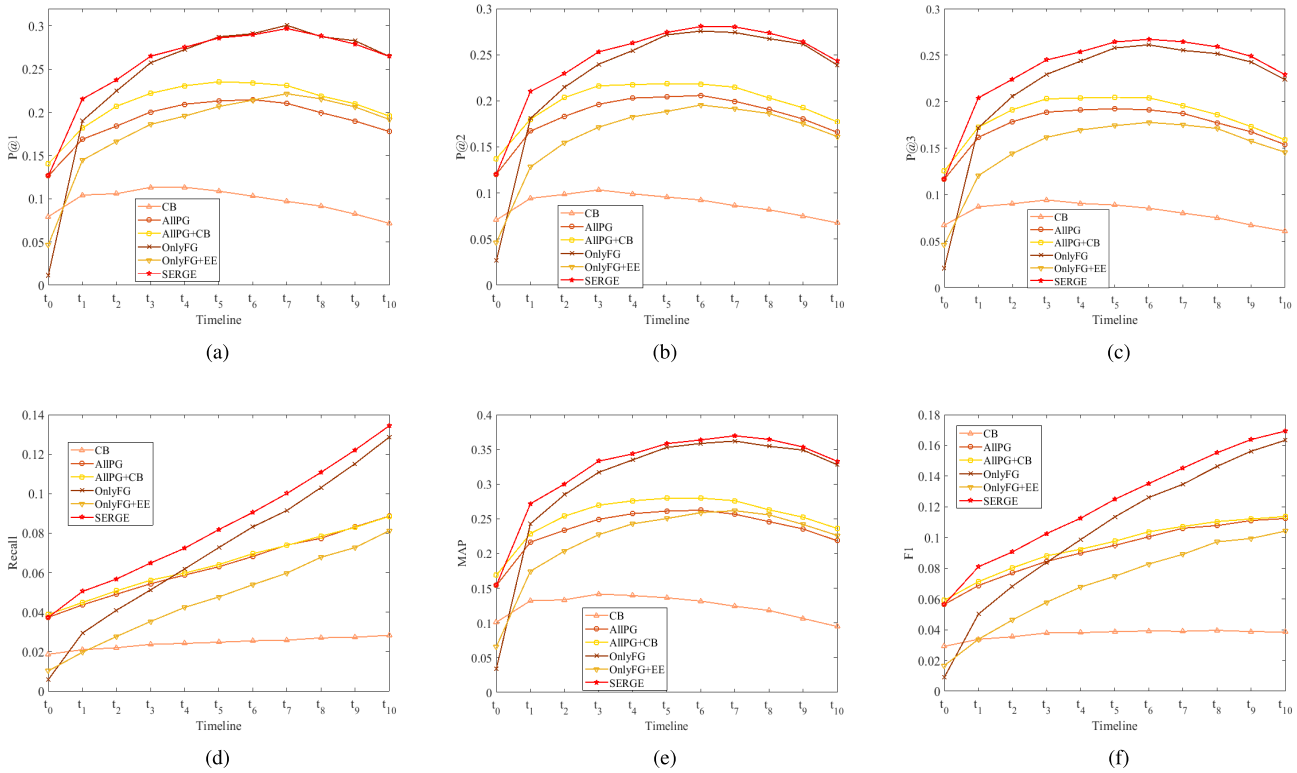


FIGURE 5. Beijing: Experiment results of P@1, P@2, P@3, Recall, MAP and F1 at different recommendation times. (a) P@1 of Beijing. (b) P@2 of Beijing. (c) P@3 of Beijing. (d) Recall of Beijing. (e) MAP of Beijing. (f) F1 of Beijing.

the SERGE constructs a feedback graph only consisting of the newest user-event relations at t_k. Such surprises, however, might be explained partially by observing the OnlyFG scheme in Fig. 5, where it performs the second-best among all schemes in most recommendation times and occasionally outperforms the SERGE in terms of a slightly higher value of P@1 at t_7 and t_9. Recall that the OnlyFG constructs only the feedback graph based on the newest user feedbacks at each recommendation time. This indicates that it focuses on the offline user social relations, other than considering all other online relations with a primary graph. For the Beijing dataset, it might imply that many Beijing users are kind of gregarious people with much offline social passion. Therefore, the offline social relations play a more important part when recommending events. However, this is not the case in the experiment results of the Shanghai dataset.

Fig. 6 presents the experiment results for Shanghai. Besides some similar observations as those in Beijing, we observe that the SERGE performs closely with the AllPG and AllPG + CB in terms of all performance metrics at many recommendation times, though sometimes slightly better, sometimes slightly worse. It is worth of noting that both the AllPG and AllPG + CB require to reconstruct a primary graph at each recommendation time. However, such primary graph construction normally asks for more computation power and memory as well as more computation time in practical systems.

From Fig. 6, it can also observe that the OnlyFG performs much worse than these three schemes. As discussed above, the OnlyFG ignores many online relations, like user-group, group-subject, host-subject and etc., while making recommendations merely based on the feedback graphs. However, the poor performance of OnlyFG in the Shanghai dataset indicates that focusing only such offline social relations are far from enough to make correct recommendations; While considering the online relations leads to better performance in the SERGE, AllPG and AllPG + CB. In contrast with such results in Beijing, our observations might suggest that many Shanghai users are not kind of offline social-enthusiasts, but caring more on individual requirements and history interests.

From both Fig. 5 and Fig. 6, when comparising the OnlyFG and OnlyFG + EE, we can observe that in most cases the OnlyFG performs better than the OnlyFG + EE. This suggests that when considering user feedbacks, it might be better to only focus on how such feedbacks would impact on offline user-event relations, other than complicating the situation with online event-event similarities.

In a short summary, among all schemes, the proposed SERGE can achieve the best or close to the best performance in terms of P@n, Recall, MAP and F1 at most of the recommendation times. As it constructs two types of graph to represent different sources of available system information at different times and uses the graph entropy-based weighting method for combining two recommendation approaches, both

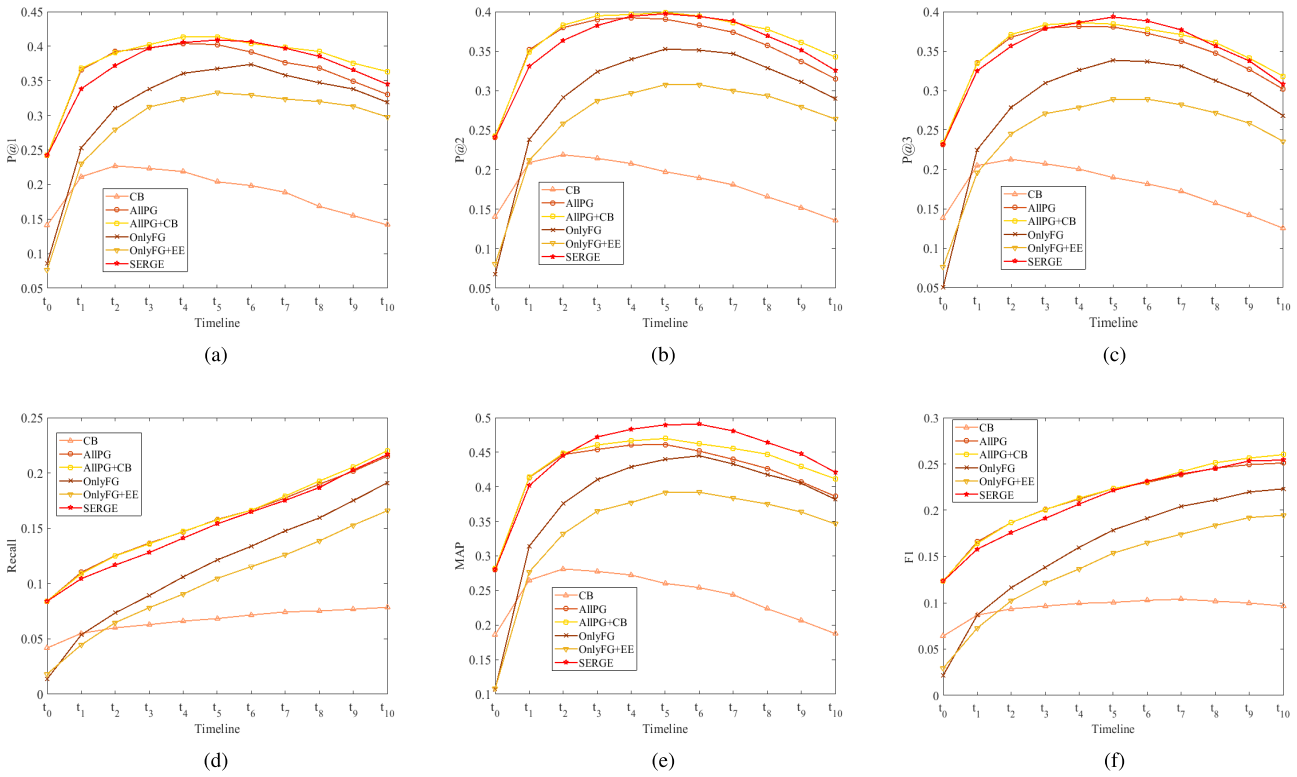


FIGURE 6. Shanghai: Experiment results of P@1, P@2, P@3, Recall, MAP and F1 at different recommendation times. (a) P@1 of Shanghai. (b) P@2 of Shanghai. (c) P@3 of Shanghai. (d) Recall of Shanghai. (e) MAP of Shanghai. (f) F1 of Shanghai.

the stationary network factors and the dynamic user feedbacks that could impact on the recommendation performance in an EBSN can be well exploited and balanced in SERGE to make better recommendation. It also embodies the merit of easier implementation for practical recommendation systems.

V. CONCLUDING REMARKS

In this paper, we have proposed the SERGE scheme for successive event recommendation in an EBSN. The SERGE constructs a primary graph to exploit diverse relations among the available entities of an EBSN, and constructs a feedback graph based on user feedbacks of the newest event reservations. The random walk with restart algorithm is applied on both graphs to obtain two sets of user-event similarity scores, yet a weighting approach based on two graph entropies is used to create the final recommendation list. Experiments on real EBSN datasets have validated the superiority of the proposed SERGE scheme over the peer ones.

In this work, we have computed the similarity in between two events only based on their event attributes, without digging into the semantic content in each event announcement. Indeed, using topic analysis for announcements could extract a more subtle feature for describing each event. Besides, semantic clustering of tags could help to generate more conceptual subjects for matching with event topical features. We note that the proposed SERGE has partially reduced the computation costs by walking on the large scale primary graph at a lower frequency. However, the primary graph might still be very large, demanding lots of storage and computation costs, which should be carefully addressed in practical implementations. We shall further take these considerations in our future work.

REFERENCES

[1] G. Liao, Y. Zhao, S. Xie, and P. S. Yu, "An effective latent networks fusion based model for event recommendation in offline ephemeral social networks," in *Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, 2013, pp. 1655–1660.

[2] W. Zhang, J. Wang, and W. Feng, "Combining latent factor model with location features for event-based group recommendation," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 910–918.

[3] H. Ding, C. Yu, G. Li, and Y. Liu, "Event participation recommendation in event-based social networks," in *Proc. Int. Conf. Soc. Informat.*, 2016, pp. 361–375.

[4] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han, "Event-based social networks: Linking the online and offline social worlds," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1032–1040.

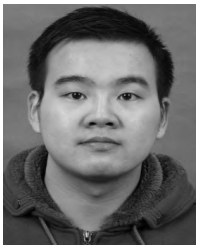
[5] F. Hao, S. Li, G. Min, H. C. Kim, S. S. Yau, and L. T. Yang, "An efficient approach to generating location-sensitive recommendations in ad-hoc social network environments," *IEEE Trans. Serv. Comput.*, vol. 8, no. 3, pp. 520–533, May 2015.

[6] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Phys. Rep.*, vol. 519, no. 1, pp. 1–49, 2012.

[7] F. Zhang, "A personalized time-sequence-based book recommendation algorithm for digital libraries," *IEEE Access*, vol. 4, pp. 2714–2720, 2016.

[8] F. Xia, Z. Chen, W. Wang, J. Li, and L. T. Yang, "MVCWalker: Random walk-based most valuable collaborators recommendation exploiting academic factors," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 364–375, Sep. 2014.

[9] L. Hu, Y. Wang, Z. Xie, and F. Wang, "Semantic preference-based personalized recommendation on heterogeneous information network," *IEEE Access*, vol. 5, pp. 19773–19781, 2017.
[10] J. Bao, Y. Zheng, D. Wilkie, and M. F. Mokbel, "Recommendations in location-based social networks: A survey," *GeoInformatica*, vol. 19, no. 3, pp. 525–565, 2015. [Online]. Available: https://link.springer.com/article/10.1007/s10707-014-0220-8
[11] P. Kefalas, P. Symeonidis, and Y. Manolopoulos, "A graph-based taxonomy of recommendation algorithms and systems in LBSNs," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 604–622, Mar. 2016.
[12] A. Q. Macedo, L. B. Marinho, and R. L. T. Santos, "Context-aware event recommendation in event-based social networks," in *Proc. 9th ACM Conf. Recommender Syst.*, 2015, pp. 123–130.
[13] Y. Gu, J. Song, W. Liu, L. Zou, and Y. Yao, "Context aware matrix factorization for event recommendation in event-based social networks," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Oct. 2016, pp. 248–255.
[14] E. M. Daly and W. Geyer, "Effective event discovery: Using location and social information for scoping event recommendations," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 277–280.
[15] M. Sklar, B. Shaw, and A. Hogue, "Recommending interesting events in real-time with foursquare check-ins," in *Proc. 6th ACM Conf. Recommender Syst.*, 2012, pp. 311–312.
[16] E. Minkov, B. Charrow, J. Ledlie, S. Teller, and T. Jaakkola, "Collaborative future event recommendation," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 819–828.
[17] X. Li, X. Cheng, S. Su, S. Li, and J. Yang, "A hybrid collaborative filtering model for social influence prediction in event-based social networks," *Neurocomputing*, vol. 230, pp. 197–209, Mar. 2017.
[18] Z. Liu, W. Qu, H. Li, and C. Xie, "A hybrid collaborative filtering recommendation mechanism for P2P networks," *Future Generat. Comput. Syst.*, vol. 26, no. 8, pp. 1409–1417, 2010.
[19] H. Khrouf and R. Troncy, "Hybrid event recommendation using linked data and user diversity," in *Proc. 7th ACM Conf. Recommender Syst.*, 2013, pp. 185–192.
[20] C.-K. Hsieh, L. Yang, H. Wei, M. Naaman, and D. Estrin, "Immersive recommendation: News and event recommendations using personal digital traces," in *Proc. 25th ACM Int. Conf. World Wide Web*, 2016, pp. 51–62.
[21] T.-A. N. Pham, X. Li, G. Cong, and Z. Zhang, "A general graph-based model for recommendation in event-based social networks," in *Proc. 31st IEEE Int. Conf. Data Eng.*, Apr. 2015, pp. 567–578.
[22] B. Li, B. Wang, Y. Mo, and L. T. Yang, "A novel random walk and scale control method for event recommendation," in *Proc. 13th IEEE Int. Conf. Ubiquitous Intell. Comput. (UIC)*, Jul. 2016, pp. 228–235.
[23] S. Liu, B. Wang, and M. Xu, "Event recommendation based on graph random walking and history preference reranking," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 861–864.
[24] L. Backstrom and J. Leskovec, "Supervised random walks: Predicting and recommending links in social networks," in *Proc. 4th ACM Int. Conf. Web Search Data Mining*, 2011, pp. 635–644.
[25] Q. Yuan, G. Cong, and A. Sun, "Graph-based point-of-interest recommendation with geographical and temporal influences," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, 2014, pp. 659–668.
[26] S. Lee, S.-I. Song, M. Kahng, D. Lee, and S.-G. Lee, "Random walk based entity ranking on graph for multidimensional recommendation," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 93–100.
[27] Y. Mo, B. Li, B. Wang, L. T. Yang, and M. Xu, "Event recommendation in social networks based on reverse random walk and participant scale control," *Future Generat. Comput. Syst.*, vol. 79, pp. 383–395, Feb. 2017.
[28] C. Guo and X. Liu, "Automatic feature generation on heterogeneous graph for music recommendation," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2015, pp. 807–810.
[29] M. Dehmer, "Information processing in complex networks: Graph entropy and information functionals," *Appl. Math. Comput.*, vol. 201, nos. 1–2, pp. 82–94, 2008.
[30] N. Eagle, M. Macy, and R. Claxton, "Network diversity and economic development," *Science*, vol. 328, no. 5981, pp. 1029–1031, 2010.
[31] M. Dehmer and A. Mowshowitz, "A history of graph entropy measures," *Inf. Sci.*, vol. 181, no. 1, pp. 57–78, 2011.
[32] H. Bagci and P. Karagoz, "Random walk based context-aware activity recommendation for location based social networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2015, pp. 1–9.



SHENGHAO LIU received the B.S. degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China, in 2016, where he is currently pursuing the Ph.D. degree. His research focuses on recommendation systems and applications.

BANG WANG received the B.S. and M.S. degrees from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology (HUST), Wuhan, China, in 1996 and 2000, respectively, and the Ph.D. degree from the Electrical and Computer Engineering Department, National University of Singapore, in 2004. He is currently a Professor with the School of Electronic Information and Communications, HUST. He has authored or co-authored over 100 technical papers in international conferences and journals. His research interests include wireless networking issues, recommendation systems, and social networks.

MINGHUA XU received the B.A. degree from the School of Journalism and Information Communication, Huazhong University of Science and Technology (HUST), in 2002, and the M.A. and Ph.D. degrees from the Social Science Department, National University of Singapore (NUS), in 2005 and 2010, respectively. She is currently an Associate Professor with the School of Journalism and Information Communication, HUST. She has authored or co-authored over 30 peer-reviewed conference and journal papers. Her research interests mainly include theory and practice of information diffusion in social networks, recommendation systems and applications, and information communication theories.