# Accepted Manuscript

Evolving graph construction for successive recommendation in event-based social networks

Shenghao Liu, Bang Wang, Minghua Xu, Laurence T. Yang

PII: S0167-739X(18)32332-X
DOI: https://doi.org/10.1016/j.future.2019.02.036
Reference: FUTURE 4789

To appear in: *Future Generation Computer Systems*

Received date : 30 September 2018
Revised date : 16 January 2019
Accepted date : 19 February 2019

# Evolving Graph Construction for Successive Recommendation in Event-based Social Networks

Shenghao Liu, Bang Wang, Minghua Xu and Laurence T. Yang

*Abstract*—**Personalized recommendation can help individual users to quickly reserve their interested events, which makes it indispensable in event-based social networks (EBSNs). However, as each EBSN is often with large amount of entities and each upcoming event is normally with non-repetitive uniqueness, how to deal with such challenges is crucial to the success of event recommendation. In this paper, we propose an *evolving graph-based successive recommendation* (EGSR) algorithm to address such challenges: The basic idea is to exploit the *random walk with restart* (RWR) on a recommendation graph for ranking the upcoming events. In EGSR, we employ a sliding window mechanism to construct evolving graphs for successively recommending new events for each user. We propose a graph entropy-based contribution measure for adjusting the window length and for weighting the history information. In EGSR, we also apply a topic analysis technique for analyzing event text description. We then propose to establish each user an interest model and to compute the similarities in between event content and user interest as edges' weights for each recommendation graph. In successive recommendation, the number of upcoming events may experience great variations in different times. For a fair comparison, we also propose a set of cumulative evaluation metrics based on the traditional recommendation performance metrics. Experiments have been conducted based on the crawled one year data from a real EBSN for two cities. Results have validated the superiority of the proposed EGSR algorithm over the peer ones in terms of better recommendation performance and reduced computation complexity.**

***Index Terms*—Evolving graph construction, successive recommendation, random walk with restart, graph entropy, event-based social networks**

## I. Introduction

With the fast development of *Internet of Things* (IoT), recent years have witnessed the emergence of a new computing paradigm, called *Cybermatics*, which have been continuously integrating diverse *Cyber, Physical and Social Systems* and promoting numerous new applications everyday [1]–[4]. For example, given the wide adoption of smartphones, people can arrange their daily life more convenient and expand their social circles [5]–[7]. While with the population of *event-based social networks* (EBSNs), people can easily reserve their interested events through their smartphones [8]–[10]. However, due to the proliferation of online events, how to accurately recommend individual users their mostly interested ones becomes a challenging task. Although some EBSNs, such as Meetup and Douban Event [1], provide a search function for users to find their preferred events with key words, how to accurately match user preferences with appropriate events is still very difficult, especially for most users being unable to clearly express their interests. In response to the pressing demands, a good event recommendation system is much required for EBSNs.

Event recommendation in EBSNs often faces the *cold start* problem [11], [12]. Compared with the general item recommendation, like recommending books and movies, events are usually with the property of non-repetitive uniqueness [13]. Furthermore, an upcoming event generally cannot be actually 'consumed' and evaluated, though it may be reserved by some users, before its commencement. To deal with such challenges, we can exploit the history events that a user had once attended to establish an interest model for him. Among many event properties, like the launching time and place, we believe that the event text description could provide more intrinsical information for reflecting users' interests. So it is necessary to analyze event text description, which can be done by enjoying some recently developed topical analysis techniques [14], [15].

Furthermore, a typical EBSN normally includes diverse entities, like events, users, groups, subjects, tags and etc., and numerous relations in between entities. Traditional recommendation algorithms, like the CB (content-based recommendation) and CF (collaborative filtering) algorithm, only pay attention to a few part of these relations, which may ignore some useful information and lead to unsatisfactory recommendation performance [16]–[20]. Recently, graph-based algorithms have been proposed to address such issues, which first construct a recommendation graph to represent all available entities and their relations [21]. After graph construction, a *random walk with restart* (RWR) algorithm can be employed to rank the nodes, whose basic idea is to transform the recommendation task into a node convergency probability computation problem [22], [23]. However, including all history entities and their relations for the graph construction incurs the problem of increased computation complexity and storage requirement. It may also introduce unnecessary noises for the random walk, if without discriminating different entities and relations.

Shenghao Liu and Bang Wang are with the School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, China. Email: {shenghao_liu, wangbang}@hust.edu.cn.

Minghua Xu is with the School of Journalism and Information Communication, Huazhong University of Science and Technology (HUST), Wuhan, China. Email: xuminghua@hust.edu.cn.

Laurence T. yang is with the Department of Computer Science, St.Francis Xavier University, Antigonish, Canada. Email: ltyang@gmail.com.

This work is supported in part by National Natural Science Foundation of China (Grant No: 61771209) and National Social Science Foundation of China (Grant No: 14CXW018). The correspondent author is Minghua Xu.

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

In this paper, we propose an *Evolving Graph-based Successive Recommendation* (EGSR) algorithm to enjoy the advantages of graph-based algorithms and topic analysis techniques. The basic idea of EGSR is to construct evolving graphs each with topical similarity weighted edges based on the most recently available system information. In particular, we first divide the timeline into consecutive slots each with equal length. We then employ a sliding window which moves forward one slot per step and use only the system information in the sliding window for graph construction. Instead of using a fixed window length, we propose a graph entropy-based slot contribution measurement to adjust the window length and to weight the history slots per moving step. We apply a topic analysis tool to obtain the content feature for each event from its text description. An interest model is then established for each user as the weighted event feature based on his attended events in the sliding window. We compute the similarities in between event features and user interests as edges' weights for each recommendation graph. Furthermore, we propose a sequential version of the stochastic gradient descent algorithm [24] to train the transition parameters for each recommendation graph. Note that although only one recommendation graph is constructed per sliding window, event recommendation can be made for each individual user on his access to the system by setting this user as the query node when executing the RWR on the graph. In practical EBSNs, the number of upcoming events may experience great variations in different slots. We then propose a set of new cumulative evaluation metrics for fair comparison of successive recommendation by different algorithms. Finally, we compare our EGSR algorithm with other peer algorithms by experimenting two real datasets crawled from Douban Event for two typical cities: Beijing and Shanghai. Experiment results show that the proposed EGSR scheme can achieve better recommendation results.

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

## II. Related Work

In this section, we mainly review the most related work on the graph-based recommendation algorithms, text content analysis for recommendation and graph entropy studies.

Graph-based recommendation algorithms have been proposed to model different kinds of entities and their rich relations by constructing a heterogenous graph, where nodes stand for entities and edges stand for their relations [25]–[39]. Among these graph-based algorithms, the RWR technique has been widely employed to obtain the convergency probabilities for ranking nodes [28]–[39]. For example, Pham et al. [28] construct a recommendation graph containing different types of all available entities and their relations, which can be used not only for group recommendation but also for other entities recommendation in EBSNs. Mo et al. [30] also construct a heterogeneous graph yet with a new reverse RWR for event recommendation to solve the dangling nodes problem. Bagci et al. [33] propose to extract a subgraph centered at each user with only his neighboring nodes and edges in the recommendation graph and apply the RWR on the subgraph for his recommendation. Liu et al. [38] present two types of recommendation graphs: one containing all the entities and their relations, yet the other containing only users and upcoming events. The RWR is performed on both graphs, yet the final ranking is based on the weighted convergency probabilities of the two graphs. However, in all these schemes the graph construction has not considered the impacts that different entities and their relations may evolve with time. Also they has not related the edges' weights with content analysis that might be more accurately reflect users' interests.

The *Latent Dirichlet Allocation* (LDA) technique which analyzes the latent topic distribution for text has been exploited by some CB and CF recommendation algorithms [40]–[45]. For example, Macedo et al. [41] apply the LDA technique to extract the topic vectors of events for calculating content similarity between users and upcoming events. Wu et al. [42] introduce a multi-level LDA model into the collaborative filtering (CF) algorithm for user recommendation in social networks. Zhao et al. [45] propose a Hashtag-LDA model to assist the collaborate filtering for hashtag recommendation in microblogs. However, to the best of our knowledge, content analysis has not been exploited for the graph-based recommendation algorithms.

Graph entropy has been used for social networks to identify the most interesting and important nodes in a network [46]–[48]. A general framework for defining the node entropy and the entropy of a graph has been introduced based on the topological structure of graph in [49]. Shetty et al. [47] adopt the graph entropy to determine the most prominent yet interesting person in an email dataset as the node that has the highest entropy in such an email network. Eagle et al. [48] develop new metrics based on the graph entropy to observe the correspondence between the communication network diversity and economic development. As the graph entropy has been proven an efficient tool in the field of social network analysis, it might also be useful for event recommendation in EBSNs.

## III. Evolving Graph Construction for Event Recommendation

### A. Overview

In this paper, we adopt the random walk on graph for event recommendation, which consists of the following modules: graph entropy-based history information inclusion and influence weighting, content topic-based preference calculation and similarity weighting, parameter training and random walk-based event recommendation.

Although a graph can detail the complex relations in between diverse entities, its construction may become a burdensome task for an ever-increasing EBSN. Furthermore, performing random walk on a very large graph with all available history information may also become time-consuming and even impractical. Instead of using all the history information for a panoramic graph construction, we propose to construct evolving graphs by using a sliding window with adjustable length to include the most recent information.


Fig. 1. Illustration of the sliding window model. The x-axis represents the timeline which is divided into several equal length slots. In this figure, we use blue boxes to denote slots and use green box to represent the current sliding window; While we let the dotted blue box be the sliding window in the next recommendation slot. In each slot, there are several events to be attended by potential users, as illustrated by the yellow dots. Furthermore, in the sliding window, slots are divided into two parts: one or more history slots and one decision slot. As time goes by, the head of sliding window will extend itself by one new slot, and the tail could keep unchanged or extend one or more slots based on our proposed moving strategy.

Fig. 1 illustrates the emulated procedure for event recommendation in a practical EBSN, where the timeline is divided into consecutive slots each with equal length. The recommendation decision is made at the beginning of each slot, yet event announcements and user reservations could asynchronously reach the EBSN at any time [38]. To capture useful history information, we use a sliding window with length of $T+1$ slots for graph construction, which consists of one *decision slot* and $T$ history slots serving as a *training set*. After having made a decision, the sliding window extends itself to cover the next slot, yet its length is subject to adjust according to our proposed *influence weighting based on graph entropy* strategy, which also computes the weight of each history slot in the sliding window for constructing a new graph. Table I summarizes the used symbols and their notations in this paper .

### B. Window moving and slot weighting strategy

The objective is to first decide the length $T+1$ of the sliding window for including history information and then to compute the influence weight $w_t$ of the $t$th history slot for the current decision slot. For event recommendation, the most important history information include the events and users, which are also the most dynamic entities. Although other entities, like groups, tags and etc. exist in an EBSN, our experiment results suggest that including all entities of an EBSN for graph construction degrades the recommendation performance. So we mainly focus on the event and user entities in this paper.

We use the event commencement time to distribute one event into its corresponding slot. Let $E_d$ and $U_d$ denote the set of events and that of users in the decision slot, respectively. We construct a bipartite graph $\mathcal{G}_d = (E_d \bigcup U_d, L_d)$, where $L_d$ is the set of edges. Note that an edge in $L_d$ only connects a user $u$ and an event $e$, if the user $u$ has reserved the event $e$. Let $\mathbf{B}$ denote the adjacency matrix of $\mathcal{G}_d$. Due to its uniqueness, an event can belong to only one slot, and hence the event sets in different slots are mutually disjoint. For those history slots, we include those users who have reserved at least one event in the sliding window. Contrary to an event, a user can participate different events, so a user can belong to multiple slots. Yet the user sets in different slots could be much different.

TABLE I

DEFINITION AND NOTATIONS IN EGSR

| Symbol | Definition |
|---|---|
| $\mathbf{A}_{EE}$, $\mathbf{A}_{UE}$ | The adjacency matrix of the constructed graph |
| $\mathbf{B}$, $\mathbf{b}_i$. | The adjacency matrix of $\mathcal{G}_d$ and its row vector |
| $C(T)$ | The entropy contribution of history information to decision slot with a window length $T+1$ |
| $\vec{e}$, $\vec{u}$ | The event feature vector and user interest model |
| $E_u^P$, $E_u^N$ | The set of positive and negative events of user $u$ |
| $F(\alpha)$, $\alpha$ | The training objective function and its parameter |
| $\mathcal{G}_d$, $\mathcal{G}$ | The bipartite graph of decision slot and the recommendation graph |
| $H$, $h$ | The nodes' entropy and a node entropy |
| $J(T)$ | The objective function |
| $\mathcal{L}_u$ | The list of events user $u$ has registered |
| $\mathbf{P}_{EE}$, $\mathbf{P}_{UE}$, $\mathbf{P}_{EU}$ | The transition matrix of recommendation graph |
| $\mathbf{q}_u$ | The user query vector |
| $r_{ij}$ | The transition probability of node $v_i$ to node $v_j$ |
| $T$ | The number of history slots in a sliding window |
| $\mathbf{u}^k$, $\mathbf{e}^k$ | The user and the event probability vector |
| $U_d^{new}$, $U_d^{old}$ | The set of new and old users in decision slot |
| $w_t$, $w_{e_{im}}$ | The weight of $t$th slot, the weight of event $e_{im}$ |
| $W_{e_i,e_j}$, $W_{u_i,e_j}$ | The weight of an edge in recommendation graph |
| $\tau$ | The time of each slot |
| $\Psi(\cdot)$, $\sigma(\cdot)$ | The unit step function and the sigmoid function |

In each decision slot, some new users may be the first time accessing the EBSN without attending any event in the $T$ history slots; While some old users may have already attended one or more events in the $T$ history slots. So we can divide users in $U_d$ into two parts: $U_d^{new}$ and $U_d^{old}$. That is, $U_d = U_d^{new} \bigcup U_d^{old}$ and $U_d^{new} \bigcap U_d^{old} = \emptyset$. For each old user, we try to also exploit his previous event attendances to establish his collaborative relations in between old events in the $T$ history slots and new events in the decision slot. For the decision slot, we expect to include as more as possible old users to exploit their history information. For including more old users, we need to extend the length of sliding window to cover longer history slots. On the other hand, increasing the window length would also increase the computation complexity, yet some too old history information may also be outdated for the current decision slot. So we need to choose an appropriate length for the sliding window.

In this paper, we determine the sliding window length $T+1$ and compute slot weight $w_t$ based on the graph entropy, which has been widely used to capture the structural information quantity of a graph [48]. We apply the graph entropy to compute the old users' contribution to the decision slot. For each node in a graph, the node entropy is computed from its

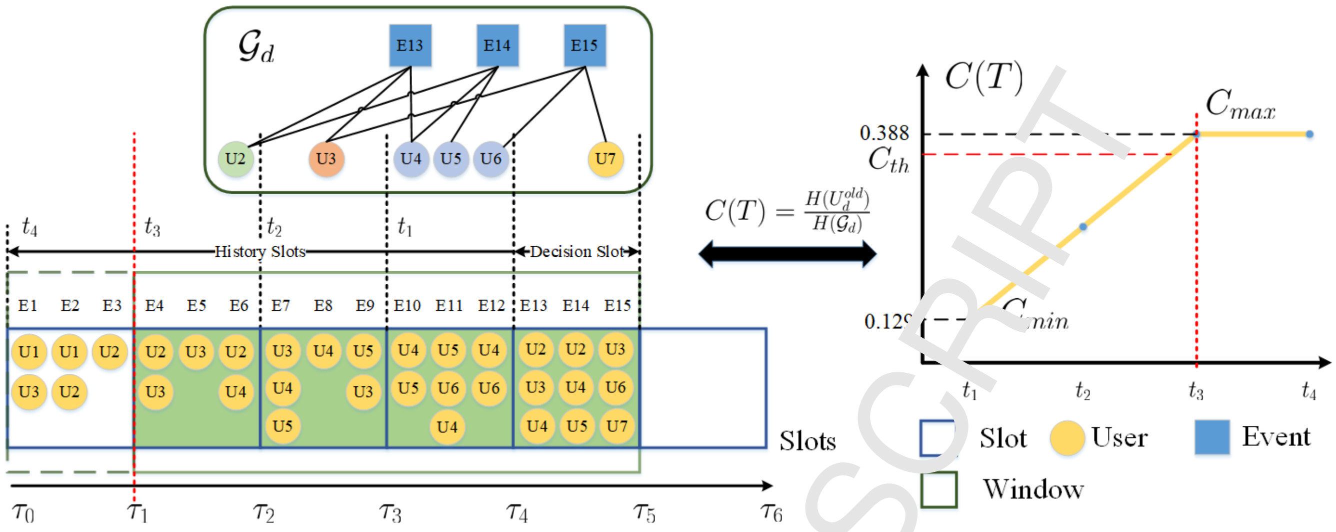



Fig. 2. Illustration of the computation of the sliding window length. On the left of the figure, the green box represents the sliding window and $\mathcal{G}_d$ denotes the bipartite graph of the decision slot, where blue squares are events and yellow dots are users. In $\mathcal{G}_d$, those users who had attended previous events in the history slots are called old users, where the old users who belong to different history slots are shown as dots with different colors. With the different choices of sliding window length $T+1$, the number of old users $|U_d^{old}|$ could be different in the graph $\mathcal{G}_d$, which could result in different entropy contributions $C(T)$. On the right of the figure, we plot $C(T)$ as a function of sliding window length. It can be seen that $C(T)$ is an increasing function of the window length. Furthermore, according to the threshold $C_{th}$, we can find the suitable length of sliding window for the current decision slot.

topological diversity information [48]:

$$h(v_i) = -\sum_{j=1}^{|\mathbf{B}|} r_{ij} \log(r_{ij}), \quad (1)$$

where $r_{ij}$ is the transition probability of node $v_i$ to node $v_j$ in the graph. For the bipartite graph $\mathcal{G}_d$, we compute $r_{ij}$ from the adjacency matrix $B$ by:

$$r_{ij} = 1/|\mathbf{b}_{i\cdot}|, \quad (2)$$

where $\mathbf{b}_{i\cdot}$ is the $i$th row vector of $\mathbf{B}$. The graph entropy of $\mathcal{G}_d$ is computed as the summation of all nodes' entropy:

$$H(\mathcal{G}_d) = \sum_{i=1}^{|\mathbf{B}|} h(v_i). \quad (3)$$

Furthermore, we compute the old user entropy $H(U_d^{old})$ by

$$H(U_d^{old}) = \sum_{i=1}^{|U_d^{old}|} h(v_i), \; v_i \in U_d^{old}. \quad (4)$$

The set of old users $U_d^{old}$ is dependent on the choice of sliding window length $T+1$. In general, the larger $T$, the larger $U_d^{old}$. For a given window length $T+1$, we define $C(T)$ as the entropy contribution of old users to $\mathcal{G}_d$ by

$$C(T) = \frac{H(U_d^{old})}{H(\mathcal{G}_d)}. \quad (5)$$

*Lemma 1:* $C(T)$ is a non-decreasing function of $T$.

*Proof:* See the Appendix . ■

For a non-decreasing $C(T)$, let $C_{min}$ denote its minimum value when $T=1$, and $C_{max}$ the maximum value for the largest allowable $T_{max}$. Furthermore, we define $C_{th}$ as a threshold to indicate the desired portion of entropy contribution by old users:

$$C_{th} = 0.9(C_{max} - C_{min}) + C_{min}. \quad (6)$$

Recall that $\mathcal{G}_d$ is different in each decision slot . So our objective for choosing a suitable window length is to let the old users' entropy contribution $C(T)$ as close as possible to the threshold $C_{th}$. Specifically, we find the most suitable sliding window length for each decision slot by:

$$\arg\min_{T} \; J(T) \equiv |C(T) - C_{th}|. \quad (7)$$

*Lemma 2:* Eq. (7) exists a unique solution.

*Proof:* See the Appendix. ■

After deciding the sliding window size $T+1$ for a decision slot, we then compute the weights of its previous slots, which will be used to discount how the old event attendances would impact on the choice of new event participation. As different users may have attended different history events, we compute the slot weight as a collective measure for all old users and history events based on the increment contribution of graph entropies. For $t = 1, ..., T$ history slots, we compute the weight $w_t$ for the $t$th slot by

$$w_t = C(t) - C(t-1), \; t = 1, 2, ..., T, \quad (8)$$

where we set $C(0) = 0$.

Fig. 2 illustrates how to find an appropriate sliding window length based on old users' entropy contribution to the current decision slot, where the function $C(T)$ is computed based on the user-event information on the left figure. It can be seen that $C(T)$ is an increasing function of window length and in the given example, the sliding window length is chosen as three history slots plus one decision slot.

### C. Graph Construction

For each decision slot, we construct a recommendation graph $\mathcal{G}$ on which the random walk will be performed. Let

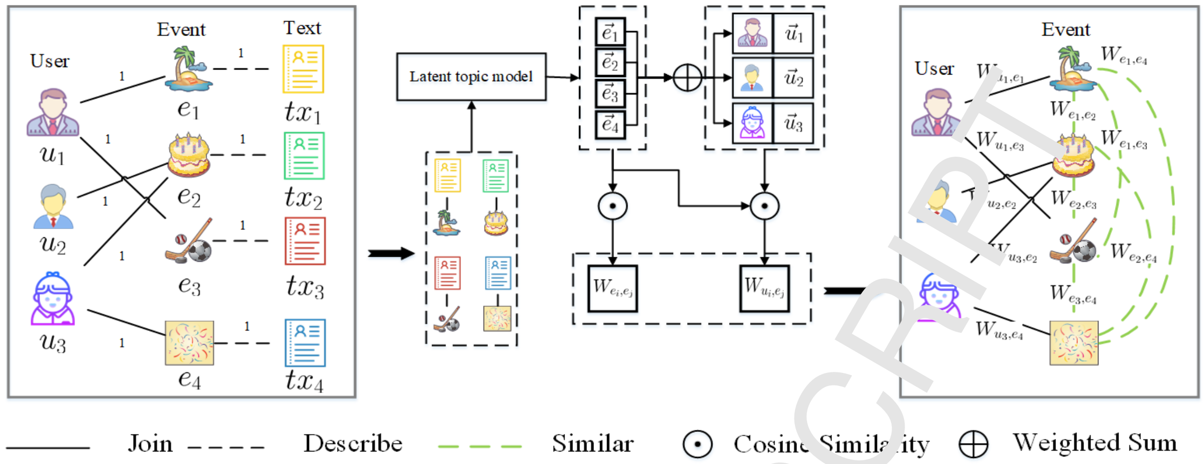



Fig. 3. Illustration of the graph model for random walk. On the left of figure, the original relations in different entities are introduced, which contains users, events and text descriptions of events. The latent topic model is applied to obtain the event feature vector $\vec{e}_j$ by analyzing the text description of each event. Furthermore, each user makes use of the feature vectors of those events that he had attended to compute his interest model $\vec{u}_i$. Based on these feature vectors, the content similarity between entities could be computed. Finally, a new graph is constructed on the right of figure, which contains new event-event edges constructed by the content similarity in between events, as illustrated by the green dotted edges. Furthermore, the weights of edges in between connected users and events are also computed based on the similarity between user interest model and event content feature.

$U$ and $E$, respectively, denote the set of available users and events in the sliding window. The graph $\mathcal{G}$ includes only the user nodes $U$ and event nodes $E$. Besides users and events, we also take event text description into consideration. We note that for almost all EBSNs like Douban Event, an event is often announced together with some text description about the event categories, selling points and other characteristics. Text description has not been well explored for graph-based event recommendation algorithms.

In this paper, we exploit text descriptions to compute user interest model and graph edge weights. We apply the widely used LDA model [14] to analyze each event text description. After having trained a LDA model, we can input an event text description to the LDA model, and obtain an output of a $K$-dimensional vector as the event feature, denoted by $\vec{e}$. Each element in $\vec{e}$ is the probability of belonging to a latent topic.

For two events, $e_i, e_j \in E$, we compute the cosine similarity of their event features as their edge weight:

$$W_{e_i,e_j} = \cos(\vec{e}_i, \vec{e}_j). \tag{9}$$

We construct a weighted adjacency matrix $\mathbf{A}_{EE}$ with each element $\mathbf{A}_{EE}(i,j) = W_{e_i,e_j}$ for describing the relations in between all events in $E$. Note that $\mathbf{A}_{EE}$ is a full matrix, which means each event node connecting with all the other event nodes.

For a user $u_i \in U$, let $\mathcal{L}_{u_i} = \{(e_{i1}, w_{e_{i1}}), ...., (e_{iM}, w_{e_{iM}})\}$ denote the list of events that the user $u_i$ has attended or reserved, where $w_{e_{im}}$ is the weight of the event $e_{im}$. For an event $e_{im}$ in the $t$-th slot of the sliding window, we set $w_{e_{im}} = w_t$ according to Eq. (8). From $\mathcal{L}_{u_i}$, we compute an interest model for the user $u_i$ as the weighted summation of event features:

$$\vec{u}_i = \sum_{e_{im} \in \mathcal{L}_{u_i}} w_{e_{im}} \vec{e}_{im}, \tag{10}$$

where $\vec{e}_{im}$ is event feature of $e_{im}$.

We next construct a weighted adjacency matrix $\mathbf{A}_{UE}$ as follows. If a user $u_i$ has not attended and has not reserved an event $e_j$, then no edge exists in between the user node $u_i$ and event node $e_j$ and $\mathbf{A}_{UE}(i,j) = 0$. If the user $u_i$ has attended or reserved the event $e_i$, then an edge exists in between $u_i$ and $e_j$, and the edge weight is computed as the cosine similarity between the user interest model and the event feature:

$$W_{u_i,e_j} = \cos(\vec{u}_i, \vec{e}_j). \tag{11}$$

Therefore, if an edge exists in between $u_i$ and $e_j$, we set $\mathbf{A}_{UE}(i,j) = W_{u_i,e_j}$. Note that the graph $\mathcal{G}$ is an undirected graph with edges fully specified by the two adjacency matrices $\mathbf{A}_{EE}$ and $\mathbf{A}_{UE}$.

Fig. 3 illustrates the construction of the recommendation graph for random walk. Notice that this recommendation graph contains two types of entities, namely, users and events, and two types of weighted edges, namely, an edge in between a user and an event and an edge in between two events. Random walk will be carried out in such a graph for each query user to obtain his recommendation list.

### D. *Random Walk with Restart on Graph*

We apply the random walk with restart on the graph $\mathcal{G}$ to compute an event recommendation list for a user, which is implemented by using a multivariate Markov chain to obtain the node convergency probabilities. To this end, we first obtain the event-event transition matrix $\mathbf{P}_{EE}$ by row-normalizing the weighted adjacency matrix $\mathbf{A}_{EE}$. Similarly, we obtain the user-event transition matrix $\mathbf{P}_{UE}$ from $\mathbf{A}_{UE}$; While we obtain the event-user transition matrix $\mathbf{P}_{EU}$ by column-normalizing $\mathbf{A}_{UE}$.

To obtain the convergency probabilities, the *random walk with restart* (RWR) algorithm is to iteratively compute the following equations:

$$\mathbf{u}^{(k+1)} = \alpha_{EU}\mathbf{e}^{(k)}\mathbf{P}_{EU} + (1-\alpha_{EU})\mathbf{q}_u \tag{12}$$

$$\mathbf{e}^{(k+1)} = \alpha_{UE}\mathbf{u}^{(k)}\mathbf{P}_{UE} + (1-\alpha_{UE})\mathbf{e}^{(k)}\mathbf{P}_{EE} \tag{13}$$

In the above equations, $\mathbf{q}_u$ is the *user query vector*. If we want to obtain the convergency probabilities for the user $u_i$, we set $\mathbf{q}_u(i) = 1$, and $\mathbf{q}_u(j) = 0$ for $i \neq j$. $\mathbf{u}^k$ and $\mathbf{e}^k$ are the user and event probability vector, respectively, in the $k$th iteration. The probability vectors $\mathbf{u}^{(0)}$ and $\mathbf{e}^{(0)}$ are randomly initialized. The parameters $\alpha_{UE}$ and $\alpha_{EU}$ control the transition weight from one type node to another type node. For example, in Eq. (13) event nodes get $\alpha_{UE}$ probability from user nodes, $(1-\alpha_{UE})$ probability from other event nodes.

The iteration terminates until the pairwise difference in between two iteration probability vectors is smaller than a predefined threshold. It has been proven in [28] that if the constructed graph is a connected one, then the iterations can converge. We note that the constructed graph $\mathcal{G}$ is a connected one. After the iteration termination, each user $u$ obtains a vector of event convergency probabilities for $N$ upcoming events in the decision slot, denoted by

$$\mathbf{p}_u = (p_u(e_1), ..., p_u(e_N)). \tag{14}$$

Each element $p_u(e_j)$ can be considered as the similarity score between $u$ and $e_j \in E_d$. We then sort the $\mathbf{p}_u$ according to the decreasing value of $p_u(e_j)$ to obtain the recommendation list $L_u$ for each user.

### *E. Parameter training*

In Eqs. (12) and (13), the parameters $\alpha_{UE}$ and $\alpha_{EU}$ control the transition weights from one type node to another type node. As the user-event pairs could be much different in different slots, we propose to sequentially train the two parameters one slot by one slot in a single sliding window. In particular, for each history slot $t_k, k = 1, .., K$ in a sliding window, we use the newly added user-event pairs from $t_{k-1}$ to $t_k$ to train new parameters $\alpha_{UE}^{t_k}$ and $\alpha_{EU}^{t_k}$ based on the previous slot parameters $\alpha_{UE}^{t_{k-1}}$ and $\alpha_{EU}^{t_{k-1}}$, respectively. We set $\alpha_{UE}^{t_1} = \alpha_{EU}^{t_1} = 0.5$ for $t_1$. Note that with this sequentially training, when the sliding window extends to the next decision slot, we only need to update the two parameters based on the current decision slot that has newly become a history slot.

Take one slot parameter training for example. Let $U$ and $E$ denote the user set and event set in this slot, respectively. For a user $u \in U$, let $E_u^P \subseteq E$ denote the set of *positive events* that the user $u$ has actually attended; and let $E_u^N \subseteq E$ denote the *negative events* that the user $u$ has not attended. Note that $E_u^P \bigcup E_u^N = E$ and $E_u^P \bigcap E_u^N = \emptyset$. The objective is to train parameters such that for each user $u \in U$, the probabilities of events in $E_u^P$ are higher than those in $E_u^N$. This can be regarded as a typical classification problem. So we adopt the AUC (Area Under the ROC Curve) as the training objective:

$$\arg\max_{\alpha} F(\alpha) = \sum_{u\in U} \frac{\sum\limits_{e_i\in E_u^P}\sum\limits_{e_j\in E_u^N}\Psi(p_u(e_i)-p_u(e_j))}{|E_u^P||E_u^N|}, \tag{15}$$

where $p_u(e_i)$ denotes the convergency probability of event $e_i$ in $\mathbf{p}_u$. $\Psi(\cdot)$ is a unit step function: It equals to 1, if $p_u(e_i) - p_u(e_j) > 0$; Otherwise, it equals to 0. Due to the discontinuities of the unit step function, a sigmoid function is often used instead in the training, $\sigma(x) = \frac{1}{1+e^{-x}}$. So the objective function becomes:

$$\arg\max_{\alpha} F(\alpha) = \sum_{u\in U} \frac{\sum\limits_{e_i\in E_u^P}\sum\limits_{e_j\in E_u^N}\sigma(p_u(e_i)-p_u(e_j))}{|E_u^P||E_u^N|}, \tag{16}$$

We apply the *stochastic gradient descent* (SGD) algorithm to find appropriate parameters. As an incremental gradient descent algorithm, the SGD is more efficient to deal with incremental training data, which can learn the parameters from the newly added training data instead of retraining all the available training data. For each parameter training user $u$, the derivative of objective function is calculated and the parameters $\alpha$ are updated as follows:

$$\alpha \leftarrow \alpha + \eta\frac{\partial F_u(\alpha)}{\partial \alpha}, \tag{17}$$

where $F_u(\alpha)$ is the objective function for parameter training for user $u$ and $\eta$ is learning rate, which is set as 0.01. Then we calculate the partial derivatives of $F_u(\cdot)$ w.r.t. $\alpha$ as:

$$\frac{\partial F_u(\alpha)}{\partial \alpha} = \frac{\sum\limits_{e_i\in E_u^P}\sum\limits_{e_j\in E_u^N}\frac{\partial\sigma(\mu_{ij})}{\partial\mu_{ij}}\left(\frac{\partial p_u(e_i)}{\partial\alpha}-\frac{\partial p_u(e_j)}{\partial\alpha}\right)}{|E_u^P||E_u^N|}, \tag{18}$$

where $\mu_{ij} = p_u(e_i) - p_u(e_j)$. For each derivative $\partial p_u(e_i)/\partial\alpha$, it is calculated by Eqs. (12) and (13). The derivatives w.r.t. parameters $\alpha_{UE}$ and $\alpha_{EU}$, respectively, are as follows:

$$\frac{\partial \mathbf{p}_u}{\partial \alpha_{UE}} = \mathbf{u}^c P_{UE} - \mathbf{e}^c P_{EE} \tag{19}$$

$$\frac{\partial \mathbf{p}_u}{\partial \alpha_{EU}} = \alpha_{UE}(\mathbf{e}^c P_{EU} - \mathbf{q}_u) P_{UE} \tag{20}$$

where $\mathbf{u}^c$ and $\mathbf{e}^c$ denote the user and event convergency probability vectors after the RWR terminates for the query user $u$, respectively. Note that since the two parameters $\alpha_{UE}$ and $\alpha_{EU}$ are independent, so we can train them separately. The training process finishes after all users in $U$ have been used for the parameter training.

## 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 15225 events and 68926 users from May 1st, 2016 to May 1st, 2017, among which in total 208976 user-event pairs are used to compose the Beijing dataset. For Shanghai, we obtained 14194 events and 94103 users from May 1st, 2016 to May 1st, 2017, among which in total 196837 user-event pairs are used to compose the Shanghai dataset. Table. II summarizes the statistics of the two datasets.

Due to the privacy policy, we are not able to obtain the details about the users' accesses to Douban Event, such as the

TABLE II
STATISTICS OF DATASET

| | | | |
|---|---|---|---|
| Beijing | User | Event | UE-Pair |
| | 68926 | 15225 | 208976 |
| | Avg.$U^{test}_{month}$ | max.$U^{test}_{month}$ | min.$U^{test}_{month}$ |
| | 503 | 1044 | 189 |
| | Avg.$U^{test}_{week}$ | max.$U^{test}_{week}$ | min.$U^{test}_{week}$ |
| | 126 | 261 | 47 |
| Shanghai | User | Event | UE-Pair |
| | 94103 | 14194 | 196837 |
| | Avg.$U^{test}_{month}$ | max.$U^{test}_{month}$ | min.$U^{test}_{month}$ |
| | 384 | 626 | 143 |
| | Avg.$U^{test}_{week}$ | max.$U^{test}_{week}$ | min.$U^{test}_{week}$ |
| | 96 | 156 | 35 |

time of each access and the corresponding action. So we have to suppose that users confirm their reservations just before the event commencement time. The dataset of each city is divided according to consecutive slots with equal length. For a decision slot $\tau$ with its sliding window length of $T+1$, a user that has actually attended at least four events is selected to compose the test user set $U^{test}_{\tau}$. Accordingly, the recommendation list is set to four for all test users, that is, $|L_u| = 4$ for all $u \in U^{test}_{\tau}$. All the events that any user $u \in U^{test}_{\tau}$ has attended are used to compose the test event set $E^{test}_{\tau}$. Notice that generally $|E^{test}_{\tau}|$ is much larger than $|U^{test}_{\tau}|$.

In this paper, we set the slot length as one month so as to ensure that the test users are not too few in each slot. In Table. II, we present some statistics of test users in one month and in one week. In the table, $U^{test}_{month}$ and $U^{test}_{week}$ denote the test user of one month and the test user of one week, respectively. In the Beijing dataset, there are in average 503 test user per month and in average 126 test user per week. Yet the minimum number of test users is 189 in one month and 47 in one week. In the Shanghai dataset, the average number of test users per month and per week is 384 and 96, respectively. Yet the minimum number of test users is 143 in one month and 35 in one week. Obviously, if we choose the slot length as one week, the small number of test users could not enough justify the recommendation results. So we choose the slot length as one month for graph construction, parameter training and performance comparison. Yet we note that for an individual user, recommendation can be made at his access to the EBSN by simply setting this user as the query user and executing the RWR algorithm on the recommendation graph. Finally, we set September 201[illegible] as the first decision slot and we have in total nine decision slots for performance evaluation.

### B. Comparison Schemes

We compare the proposed scheme, called EGSR (*evolving graph based successive recommendation*) with some representative peer schemes, including the content-based filtering and graph-based random walk. In our proposed EGSR scheme, we have used the LDA tool for analyzing event text description and establishing user interest model. Yet our interest model for one user is based on the weighted sum of the event features from the events that the user had attended in the history slots; While the weights are obtained from our algorithm for sliding window length determination. For a given training dataset with $E_t$ as its event set, we can also establish an interest model for each user. For a user $u_i$, let $\mathcal{L}_{u_i}(E_t) = \{e_{i1}, ..., e_{iM}\}$ denote the list of events in $E_t$ that the user $u_i$ has actually attended. We compute the user interest model by

$$\vec{u}_i(E_t) = \sum_{e_{im} \in \mathcal{L}_{u_i}(E_t)} \vec{e}_{im}, \tag{21}$$

where $\vec{e}_{im}$ is the LDA feature of event $e_{im}$.

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

- CB: This is the classic content-based recommendation [50]. We compute each user interest model by Eq. (21) with the training dataset covering all available history slots. In each decision slot, we compute the cosine similarity between the user interest model and the topic feature of each upcoming event and generate the recommendation list according to the decreasing value of the similarities.
- HG: It applies the random walk on a heterogeneous graph for event recommendation. The heterogeneous graph contains not only user nodes and event nodes, but also online group nodes and subject nodes [36]. The subject nodes are generated by clustering property tags of events and groups. For each decision slot, the training dataset covers all of its available history slots.
- BG: It applies the random walk on a bipartite graph for event recommendation [51]. For each decision slot, the bipartite graph contains only user nodes and event nodes from its previous four history slots. An edge only connects one user node and one event node, if the user has reserved the event. Furthermore, in the graph adjacency matrix, all edges are with the same weight.
- wBG: It applies the random walk on a weighted bipartite graph, which shares the same graph structure as that in the BG scheme, yet with different edge weights. For each decision slot, we compute each user interest model by Eq. (21) with the training dataset covering the previous four history slots. The weight of an edge is compute as the cosine similarity of user interest model and the corresponding event feature.
- wBGa: It adopts the same procedure as that of the wBG scheme, yet with the only difference of using all of its available history slots as the training dataset for each decision slot. Note that the graph structure in wBGa is generally much complex than that in wBG, as with the time elapse, more old users and events would be included in the wBGa graph.

In the above schemes, the BG and wBG use a fixed window length with four training months; While the CB, HG and wBGa schemes use all the previous months for the training set. Our EGSR scheme adaptively adjusts the sliding window length, So the length of sliding window differs across different decision slots. Note that at the first decision slot $\tau_1$, we use

TABLE III
COMPARISON OF SLIDING WINDOW LENGTH AND USER-EVENT PAIRS IN DIFFERENT DECISION SLOTS.

| (Window Length; UE-Pairs) | | $\tau_1$ | $\tau_2$ | $\tau_3$ | $\tau_4$ | $\tau_5$ | $\tau_6$ | $\tau_7$ | $\tau_8$ | $\tau_9$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Beijing | Adaptive months | (5; 98843) | (4; 82514) | (4; 78471) | (4; 72668) | (5; 81805) | (5; 66242) | (5; 6[illegible]57) | (5; 71668) | (5; 61977) |
| | Fixed months | (5; 98843) | (5; 97730) | (5; 98375) | (5; 95731) | (5; 81805) | (5; 66242) | (5; 68957) | ([illegible]; 71668) | (5; 61977) |
| | All months | (5; 98843) | (6; 113855) | (7; 129716) | (8; 146976) | (9; 156113) | (10; 165085) | (1[illegible]; 182[illegible]12) | (12; 201384) | (13; 208953) |
| Shanghai | Adaptive months | (5; 87479) | (4; 80845) | (4; 80098) | (4; 77099) | (4; 65790) | (5; 72905) | ([illegible]; 65[illegible]95) | (6; 78560) | (6; 70333) |
| | Fixed months | (5; 87479) | (5; 97862) | (5; 99913) | (5; 97472) | (5; 86490) | (5; 72905) | (5; 65[illegible]5) | (5; 59492) | (5; 52959) |
| | All months | (5; 87479) | (6; 107436) | (7; 126504) | (8; 143878) | (9; 153269) | (10; 160384) | (11; [illegible]72731) | (12; 185996) | (13; 196837) |

all the history slots to confirm that every slot would be used in the experiments. Table III compares the statistics of using different numbers of training months. It can be seen that the EGSR involves fewer user-event pairs for graph construction in most cases.

### C. Experiment results

In this paper, we firstly adopt four traditional evaluation metrics for recommendation: P@n (Precision at position $n$), MAP (Mean Average Precision), Recall and F1. For a user $u_i$ $(i = 1, ..., M)$ in the test set, let $L_i$ denote his recommendation list and $N$ be the list length. Let $\mathcal{H}_i$ denote the set of events that the user $u_i$ has actually attended, which are called his positive events.

Both P@n and MAP are used to measure the hit rate with taking top $n$ position of positive events into consideration. P@n is defined as follows:

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

where $\mathbb{I}(\cdot)$ is an indicator function and $L_i^{(j)}$ the $j$th event in the user $u_i$'s recommendation list.

MAP is the mean of the *average precision* (AP) scores over all test users, where AP is calculated by:

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

where $L_i^{(n)}$ denotes the $n$th recommended event in the list $L_i$. $|\mathcal{H}_i|$ represents the number of events that had been actually attended by the $u_i$ in the test set. Thus, MAP is defined by

$$MAP = \frac{\sum_{u_i \in U^{test}} AP_i}{|U_\tau^{test}|}, \tag{24}$$

Recall reflects the proportion of events that users have actually attended in the top-$n$ place. Take user $u_i$ for example, his recall $R_i(L)$ is defined by

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

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 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.

$$F1 = \frac{2PR}{P + R}, \tag{26}$$

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

Tables IV and V compare the experiment results of Beijing and Shanghai, respectively, for the nine successive decision slots. From both tables, we first observe that the CB scheme performs the worst in terms of all performance metrics and in almost all recommendations. This is not unexpected as it only exploits the users' history participation information for recommending new events, without considering potential relations like the topical similarity in between events and the common interests in between users. The HG scheme, on the other hand, includes all the available entities for graph construction, trying to establish all potential relations among different entities. However, its performance is also not good enough, and in most cases, it plays the second worst or even the worst among the six schemes. This could be due to its indiscrimination about the different importance of these entities and their relations. For example, the entities of online groups and event subjects might not be able to precisely reflect a user real interest, as the online groups may not be directly translated into offline event attendances; While event subjects only provide rough event categorizations, which may not be able to capture the main characteristics for each single event, like that done by the latent topic distribution analysis. Compared with the HG scheme, the BG scheme only includes the user and event entities for graph construction, however, its performance is better than that of the HG scheme. This collaborates our conjecture that using the most related entities for graph construction could be better than using all available entities.

From Tables IV and V, we can observe that the proposed scheme EGSR can outperform the other peer schemes in almost all the decision slots. For example, it achieves the best P@1 results among all the schemes in the Beijing dataset. We also note that in some decision slots, the wBG or wBGa scheme performs the best for some performance metrics. Nonetheless, the best results are achieved only by the EGSR, wBG and wBGa schemes: See the bolded results appearing only in the last three columns for each performance metric. Recall that the three schemes only establish relations in between users and events. Furthermore, they all apply the LDA tool for event content analysis to extract latent topic feature and construct weighted edges based on the topic-related similarities. Such results collaborate our conjectures that using the most important entities and using topic-related edge weights for graph construction can lead to better recommendation results. On the other hand, although the EGSR performs the best in almost all cases, it is sometimes not better than the wBG or wBGa in some slots. This could be attributed to that the numbers of test users and events are much different in different decision slots.

We next propose a set of new performance metrics to enable fair performance comparison for successive recommendations, which takes into considerations of test dataset size. We first define a slot coefficient based on the hit rate of *random recommendation*, which randomly selects $K$ events from all available $N_\tau$ events in the $\tau$th decision slot. The *average hit rate* (AHR) $\gamma_{\tau,u}$ of such a random recommendation for the user $u$ in the $\tau$th slot thus can be computed by

$$\gamma_{\tau,u} = \sum_{k=1}^{K} \frac{k}{K} \times \frac{\binom{N^P_{\tau,u}}{k}\binom{N_\tau - N^P_{\tau,u}}{K-k}}{\binom{N_\tau}{K}}. \tag{27}$$

where $N^P_{\tau,u}$ is the number of positive events that the user $u$ has actually attended in the $\tau$th slot, and $K$ is the length of recommendation list, which is set to 4 in our experiments. For each random selection of $K$ events, if there are $k$ positive events, then the hit rate is $\frac{k}{K}$. So $\gamma_{\tau,u}$ computes the AHR of the random recommendation for the user $u$. The mean AHR over all test users can be computed by:

$$\gamma_\tau = \frac{\sum_{u \in U^{test}_\tau} \gamma_{\tau,u}}{|U^{test}_\tau|} \tag{28}$$

where $U^{test}_\tau$ represents the set of test user in $\tau$th slot.

As $N^P_{\tau,u}$ differs across different test users, the computation of $\gamma_{\tau,u}$ becomes user-dependent. On the other hand, as we select the test users as those who have attended at least $K$ events, so we replace $N^P_{\tau,u}$ by the length of recommendation list to reduce the computation complexity. That is, we set $N^P_{\tau,u} = K$ for all users to compute the worst case of hit rate for all test users. Based on Eqs. (27) and (28), we then compute the slot coefficient $\bar{\gamma}_\tau$ as the worst case of mean AHR over all test users:

$$\bar{\gamma}_\tau = \sum_{k=1}^{K} \frac{k}{K} \times \frac{\binom{K}{k}\binom{N_\tau - K}{K-k}}{\binom{N_\tau}{K}}. \tag{29}$$

For the relation between $\bar{\gamma}_\tau$ and $N_\tau$, we have the following lemma:

*Lemma 3:* $\bar{\gamma}_\tau$ is a decreasing function of $N_\tau$.

*Proof:* See the appendix. ■

Lemma 3 states that for the random selection the larger the number of test events, the smaller the average hit rate. In other words, even for this random recommendation, it is likely that its average hit rate could be very high due to a small number of test events. Therefore, to reduce the impact of event number variations in different slots, we propose a slot performance weight as a decreasing function of $\bar{\gamma}_\tau$:

$$f(\bar{\gamma}_\tau) = -\log_2 \bar{\gamma}_\tau. \tag{30}$$

For successive recommendation, besides the variations of test dataset, the training dataset can also be much different in different decision slots. For different recommendation algorithms, as they enable different choices of history slots for composing a training dataset, it is also necessary to compare their cumulative performance till the current decision slot. To do so, we propose a new cumulative metric $cX_\tau$ based on the weighted average of the traditional performance metric $X_\tau$:

$$cX_\tau = \frac{\sum_{j=1}^{\tau} f(\bar{\gamma}_j) \times X_j}{\sum_{j=1}^{\tau} f(\bar{\gamma}_j)}. \tag{31}$$

For example, cP@n is the cumulative version of P@n; While cMAP is the cumulative version of MAP.

TABLE VI
RANDOM SELECTION: THE NUMBER OF TEST EVENTS $N_\tau$, THE MEAN AHR $\bar{\gamma}_\tau$ AND THE SLOT PERFORMANCE WEIGHT $f(\bar{\gamma}_\tau)$.

| | Beijing | | | Shanghai | | |
|---|---|---|---|---|---|---|
| Slot | $N_\tau$ | $\bar{\gamma}_\tau$ | $f(\bar{\gamma}_\tau)$ | $N_\tau$ | $\bar{\gamma}_\tau$ | $f(\bar{\gamma}_\tau)$ |
| $\tau_1$ | 1400 | 0.00286 | 8.451 | 1222 | 0.00327 | 8.255 |
| $\tau_2$ | 1014 | 0.00394 | 7.986 | 1130 | 0.00354 | 8.142 |
| $\tau_3$ | 1083 | 0.00369 | 8.081 | 1088 | 0.00368 | 8.087 |
| $\tau_4$ | 1203 | 0.00333 | 8.232 | 1149 | 0.00348 | 8.166 |
| $\tau_5$ | 850 | 0.00471 | 7.731 | 706 | 0.00567 | 7.464 |
| $\tau_6$ | 897 | 0.00446 | 7.809 | 785 | 0.00510 | 7.617 |
| $\tau_7$ | 1248 | 0.00321 | 8.285 | 1152 | 0.00347 | 8.170 |
| $\tau_8$ | 1224 | 0.00327 | 8.257 | 1089 | 0.00367 | 8.089 |
| $\tau_9$ | 620 | 0.00645 | 7.276 | 1017 | 0.00393 | 7.990 |

Table VI provides the number of test events $N_\tau$, the mean AHR $\bar{\gamma}_\tau$ and the slot performance weight $f(\bar{\gamma}_\tau)$ in each decision slot of our experiments. The slot weights are used to compute the cumulative performance metrics. Figs. 4 and 5 compare the cumulative performance results of the six algorithms for Beijing and Shanghai, respectively. At first, we can observe that the cumulative performance results become less variable for different decision slots, as they have applied the weighted average to remove user-event variations. For the new cumulative metrics, we can observe that both the CB and HG schemes perform much worse than the other four schemes in the two datasets. This again validates the advantages of applying random walk on a graph constructed by using two core entities of an EBSN. On the other hand, we observe that the proposed EGSR scheme performs the best in terms of all cumulative metrics in the Beijing dataset. For the Shanghai dataset, the EGSR is only slightly worse in the first decision slot, which is not much unexpected as the user interest model in the first decision slot may not be accurate enough in the first place. As time goes by, the proposed EGSR can have used more history information to obtain a more accurate interest model for more users, so it can outperform the other schemes in all the subsequent decision slots in the Shanghai dataset.

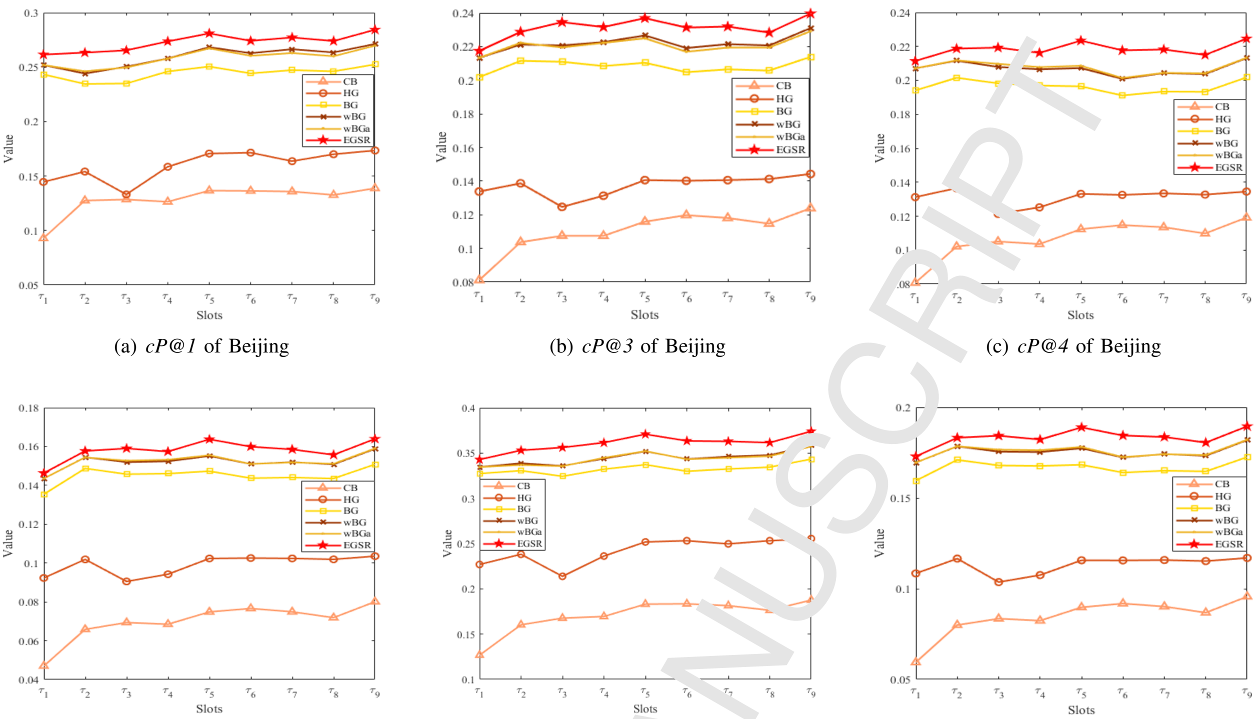
As a short summary of our experiments, the performance of the proposed EGSR scheme outperforms the state-of-the-art schemes in terms of all metrics and in most cases for the two datasets. This first suggests that when constructing a graph

TABLE IV
RECOMMENDATION PREDICTION PERFORMANCE COMPARISON BY SIX TRADITIONAL EVALUATION METRICS OF BEIJING

| | P@1 | | | | | | P@3 | | | | | | P@4 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CB | HG | BG | wBG | wBGa | EGSR | CB | HG | BG | wBG | wBGa | EGSR | CB | HG | BG | wBG | wBGa | EGSR |
| $\tau_1$ | 0.0929 | 0.1446 | 0.2433 | 0.2519 | 0.2519 | **0.2615** | 0.0811 | 0.1338 | 0.2018 | 0.2133 | 0.2133 | **0.2174** | 0.0807 | 0.1312 | 0.194 | 0.2071 | 0.2071 | **0.2114** |
| $\tau_2$ | 0.1642 | 0.1642 | 0.2253 | 0.2358 | 0.2400 | **0.2653** | 0.1277 | 0.1439 | 0.2218 | 0.2295 | 0.2316 | **0.2407** | 0.1247 | [illegible] | 0.2095 | 0.2163 | 0.2168 | **0.2263** |
| $\tau_3$ | 0.1305 | 0.0905 | 0.2358 | 0.2632 | 0.2568 | **0.2695** | 0.1151 | 0.0961 | 0.2098 | 0.2196 | 0.2140 | **0.2463** | 0.1111 | [illegible] | 0.1916 | 0.2005 | 0.2053 | **0.2205** |
| $\tau_4$ | 0.1202 | 0.2340 | 0.2788 | 0.2804 | 0.2821 | **0.2981** | 0.1074 | 0.1512 | 0.2009 | 0.2286 | **0.2302** | 0.2228 | [illegible] | 0.1366 | 0.1931 | 0.2023 | 0.2023 | **0.2071** |
| $\tau_5$ | 0.1799 | 0.2222 | 0.2698 | 0.3122 | 0.3069 | **0.3122** | 0.1517 | 0.1799 | 0.2187 | 0.2434 | 0.2363 | **0.2593** | 0.1495 | [illegible] | 0.1944 | 0.2103 | 0.2116 | **0.2540** |
| $\tau_6$ | 0.1351 | 0.1757 | 0.2117 | 0.2342 | 0.2252 | **0.2387** | 0.1396 | 0.1381 | 0.1757 | 0.1802 | 0.1757 | **0.2027** | [illegible] | 0.1295 | 0.1633 | 0.1678 | 0.1644 | **0.1881** |
| $\tau_7$ | 0.1324 | 0.1180 | 0.2647 | 0.2878 | 0.2777 | **0.2950** | 0.1084 | 0.1429 | 0.2163 | 0.2350 | 0.2336 | **[illegible]** | 0.1061 | 0.1385 | 0.2068 | **0.2241** | 0.2219 | 0.2212 |
| $\tau_8$ | 0.1109 | 0.2137 | 0.2382 | 0.2431 | 0.2398 | **0.2512** | 0.0914 | 0.1457 | 0.2007 | 0.2153 | **0.2197** | [illegible] | [illegible] | 0.1277 | 0.1921 | 0.1998 | **0.2023** | 0.1925 |
| $\tau_9$ | 0.1937 | 0.2042 | 0.3089 | 0.3403 | 0.3560 | **0.3770** | 0.2059 | 0.1710 | 0.2862 | 0.3229 | 0.3159 | **[illegible]** | [illegible] | 0.1505 | 0.2775 | 0.2971 | 0.2945 | **0.3102** |
| | Recall | | | | | | MAP | | | | | | F1 | | | | | |
| | CB | HG | BG | wBG | wBGa | EGSR | CB | HG | BG | wBG | wBGa | EGSR | CB | HG | BG | wBG | wBGa | EGSR |
| $\tau_1$ | 0.0470 | 0.0923 | 0.1354 | 0.1435 | 0.1435 | **0.1463** | 0.1268 | 0.2269 | 0.3273 | 0.3345 | 0.3345 | **0.3428** | 0.0594 | 0.1084 | 0.1595 | 0.1695 | 0.1695 | **0.1729** |
| $\tau_2$ | 0.0858 | 0.1120 | 0.1628 | 0.1660 | 0.1659 | **0.1698** | 0.1957 | 0.2501 | 0.3340 | 0.3432 | 0.3393 | **[illegible]** | 0.1017 | 0.1252 | 0.1832 | 0.1879 | 0.1880 | **0.1941** |
| $\tau_3$ | 0.0765 | 0.0674 | 0.1397 | 0.1469 | 0.1492 | **0.1617** | 0.1827 | 0.1643 | 0.3116 | [illegible] | 0.3329 | **0.3626** | 0.0906 | 0.0773 | 0.1616 | 0.1696 | 0.1728 | **0.1866** |
| $\tau_4$ | 0.0659 | 0.1053 | 0.1469 | 0.1536 | **0.1548** | 0.1526 | 0.1749 | 0.3029 | 0.3556 | 0.3677 | [illegible] | **0.3774** | 0.0791 | 0.1190 | 0.1669 | 0.1747 | 0.1754 | **0.1757** |
| $\tau_5$ | 0.1017 | 0.1367 | 0.1526 | 0.1667 | 0.1652 | **0.1904** | 0.2409 | 0.3178 | 0.3582 | [illegible] | 0.3835 | **0.4103** | 0.1210 | 0.1502 | 0.1710 | 0.1860 | 0.1856 | **0.2177** |
| $\tau_6$ | 0.0858 | 0.1037 | 0.1250 | 0.1297 | 0.1275 | **0.1402** | 0.1852 | 0.2601 | 0.2920 | 0.2988 | 0.2965 | **0.3247** | 0.1025 | 0.1152 | 0.1416 | 0.1463 | 0.1436 | **0.1606** |
| $\tau_7$ | 0.0648 | 0.1013 | 0.1460 | **0.1573** | 0.1566 | 0.1502 | 0.1696 | 0.2295 | [illegible] | **[illegible]** | 0.3544 | 0.3602 | 0.0804 | 0.1170 | 0.1712 | **0.1849** | 0.1836 | 0.1789 |
| $\tau_8$ | 0.0513 | 0.0986 | 0.1400 | 0.1433 | **0.1466** | 0.1362 | 0.1414 | 0.2769 | 0.3484 | 0.3580 | **0.3593** | 0.3520 | 0.0638 | 0.1113 | 0.1619 | 0.1669 | **0.1700** | 0.1595 |
| $\tau_9$ | 0.1531 | 0.1180 | 0.2157 | 0.2298 | 0.2292 | **0.2379** | 0.2880 | 0.2760 | [illegible] | [illegible] | 0.4516 | **0.4857** | 0.1745 | 0.1323 | 0.2427 | 0.2591 | 0.2578 | **0.2693** |

TABLE V
RECOMMENDATION PREDICTION PERFORMANCE COMPARISON BY SIX TRADITIONAL EVALUATION METRICS OF SHANGHAI

| | P@1 | | | | | | P@3 | | | | | | P@4 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CB | HG | BG | wBG | wBGa | EGSR | CB | HG | BG | wBG | wBGa | EGSR | CB | HG | BG | wBG | wBGa | EGSR |
| $\tau_1$ | 0.0942 | 0.1022 | 0.1901 | 0.1869 | 0.1869 | **[illegible]** | 0.0895 | 0.0985 | 0.1667 | 0.1704 | **0.1704** | 0.1651 | 0.0843 | 0.0970 | 0.1518 | 0.1593 | **0.1593** | 0.1550 |
| $\tau_2$ | 0.1261 | 0.1243 | 0.2541 | 0.2450 | 0.2450 | **[illegible]** | 0.1237 | 0.1123 | 0.1934 | 0.1952 | 0.1952 | **0.2168** | 0.1234 | 0.1068 | 0.1725 | 0.1865 | 0.1842 | **0.1982** |
| $\tau_3$ | 0.1095 | 0.1652 | 0.2513 | 0.2693 | 0.2765 | **0.2765** | 0.0946 | 0.1245 | 0.1861 | 0.2017 | 0.2059 | **0.2101** | 0.0965 | 0.1068 | 0.1732 | 0.1858 | 0.1827 | **0.1881** |
| $\tau_4$ | 0.1466 | 0.0957 | 0.2098 | 0.2057 | 0.2057 | **0.2159** | 0.1181 | 0.1018 | 0.1663 | 0.1724 | 0.1752 | **0.1874** | 0.1090 | 0.1008 | 0.1609 | 0.1650 | 0.1645 | **0.1787** |
| $\tau_5$ | 0.1975 | 0.1173 | 0.2778 | 0.2840 | **[illegible]** | [illegible] | 0.1584 | 0.1173 | 0.2572 | **0.2675** | 0.2654 | 0.2572 | 0.1451 | 0.1265 | 0.2454 | **0.2500** | 0.2392 | 0.2392 |
| $\tau_6$ | 0.1818 | 0.1818 | 0.2098 | 0.2168 | **[illegible]** | 0.1888 | 0.1445 | 0.1515 | 0.1795 | 0.1841 | 0.1841 | **0.1958** | 0.1416 | 0.1399 | 0.1661 | 0.1713 | 0.1643 | **0.1783** |
| $\tau_7$ | 0.1552 | 0.1372 | 0.2058 | 0.2094 | 0.2238 | **[illegible]** | 0.1288 | 0.1035 | 0.1685 | 0.1709 | **0.2010** | 0.1949 | 0.1218 | 0.1011 | 0.1543 | 0.1570 | 0.1742 | **0.1805** |
| $\tau_8$ | 0.1399 | 0.1119 | 0.2273 | [illegible] | [illegible] | **[illegible]** | 0.1329 | 0.1026 | 0.1795 | 0.1911 | 0.1911 | **0.2133** | 0.1224 | 0.0953 | 0.1748 | 0.1836 | 0.1757 | **0.2002** |
| $\tau_9$ | 0.1159 | 0.1232 | 0.2101 | 0.2174 | [illegible] | **0.2826** | 0.1039 | 0.1002 | 0.1703 | 0.1679 | 0.1824 | **0.2041** | 0.1024 | 0.1014 | 0.1540 | 0.1685 | 0.1703 | **0.1875** |
| | Recall | | | | | | MAP | | | | | | F1 | | | | | |
| | CB | HG | BG | wBG | wBGa | EGSR | CB | HG | BG | wBG | wBGa | EGSR | CB | HG | BG | wBG | wBGa | EGSR |
| $\tau_1$ | 0.0559 | 0.0714 | [illegible] | [illegible] | **0.1105** | 0.1061 | 0.1378 | 0.1781 | 0.2601 | 0.2662 | **0.2662** | 0.2660 | 0.0672 | 0.0823 | 0.1250 | 0.1305 | **0.1305** | 0.1260 |
| $\tau_2$ | 0.0886 | 0.0859 | [illegible] | [illegible] | 0.1419 | **0.1476** | 0.1730 | 0.1914 | 0.3180 | 0.3252 | 0.3219 | **0.3372** | 0.1032 | 0.0952 | 0.1513 | 0.1622 | 0.1603 | **0.1692** |
| $\tau_3$ | 0.0690 | 0.0828 | [illegible] | [illegible] | 0.1368 | **0.1419** | 0.1505 | 0.2178 | 0.3159 | 0.3304 | 0.3327 | **0.3363** | 0.0804 | 0.0933 | 0.1488 | 0.1597 | 0.1564 | **0.1618** |
| $\tau_4$ | 0.0802 | [illegible] | [illegible] | 0.1285 | 0.1274 | **0.1356** | 0.1839 | 0.1593 | 0.2630 | 0.2653 | 0.2683 | **0.2798** | 0.0924 | 0.0911 | 0.1404 | 0.1445 | 0.1436 | **0.1542** |
| $\tau_5$ | 0.1004 | 0.1000 | [illegible] | **0.1905** | 0.1863 | 0.1820 | 0.2318 | 0.2003 | 0.3639 | 0.3716 | **0.3747** | 0.3525 | 0.1186 | 0.1117 | 0.2106 | **0.2162** | 0.2095 | 0.2067 |
| $\tau_6$ | 0.0966 | 0.1108 | [illegible] | 0.1288 | 0.1226 | **0.1291** | 0.2119 | 0.2667 | 0.2741 | **0.2766** | 0.2733 | 0.2723 | 0.1149 | 0.1236 | 0.1455 | 0.1470 | 0.1404 | **0.1498** |
| $\tau_7$ | 0.0832 | 0.0749 | 0.1115 | 0.1118 | 0.1248 | **0.1264** | 0.1984 | 0.1779 | 0.2532 | 0.2554 | **0.2875** | 0.2865 | 0.0989 | 0.0860 | 0.1295 | 0.1306 | 0.1454 | **0.1487** |
| $\tau_8$ | 0.0898 | 0.0757 | 0.1352 | 0.1414 | 0.1357 | **0.1518** | 0.1852 | 0.1793 | 0.2999 | 0.3124 | 0.3137 | **0.3365** | 0.1036 | 0.0844 | 0.1525 | 0.1597 | 0.1531 | **0.1727** |
| $\tau_9$ | 0.0788 | 0.0797 | 0.1206 | 0.1311 | 0.1316 | **0.1447** | 0.1531 | 0.1932 | 0.2916 | 0.3001 | 0.3082 | **0.3621** | 0.0890 | 0.0893 | 0.1353 | 0.1475 | 0.1485 | **0.1634** |




(a) *cP@1* of Beijing (b) *cP@3* of Beijing (c) *cP@4* of Beijing

(d) *cRecall* of Beijing (e) *cMAP* of Beijing (f) *cF1* of Beijing

Fig. 4. Beijing: Experiment results of *cP@n*, *cP@3*, *cP@4*, *cRecall*, *cMAP*, *cF1* in different recommendation slots.

for RWR-based recommendation, it might be better to use the most related entities and the most recent information for graph construction, rather than using all available entities and all history information. In addition, experiment results also suggest that using weighted edge by applying the latent topic model is effective for achieving better recommendation results.

## V. Conclusion

In this paper, we have proposed the EGSR scheme which applies the RWR on a graph for successive event recommendation in EBSNs. For its practical implementation in large ENSBs, the EGSR scheme exploits a sliding window to include only the most recent information and the core entities for composing a recommendation graph. Furthermore, based on the topic analysis for event text description, it assigns edges' weights based on the similarity computation in between event features and user interests. Experiments from a real EBSN, Douban Event, have validated its superiority over the peer schemes in terms of better recommendation results.

In our experiments, we have noticed that the constructed graph could still be in a very large scale, even given our efforts of applying a sliding window. Furthermore, we have also noticed that the random walk paths for each recommendation could be rather repetitive due to the redundant graph structure. In our future work, we shall investigate some approaches for graph partition and graph embedding to further improve the efficiency and effectiveness of RWR-based recommendation.

## Appendix

*Proof:* Lemma 1: We firstly rewrite Eq.(5) as follows:

$$C(t) = \frac{H(U_d^{old_t})}{H(\mathcal{G}_d)} \tag{32}$$

where $U_d^{old_t}$ is the set of old users when the length of window is $t$. As $H(\mathcal{G}_d)$ is a constant, so

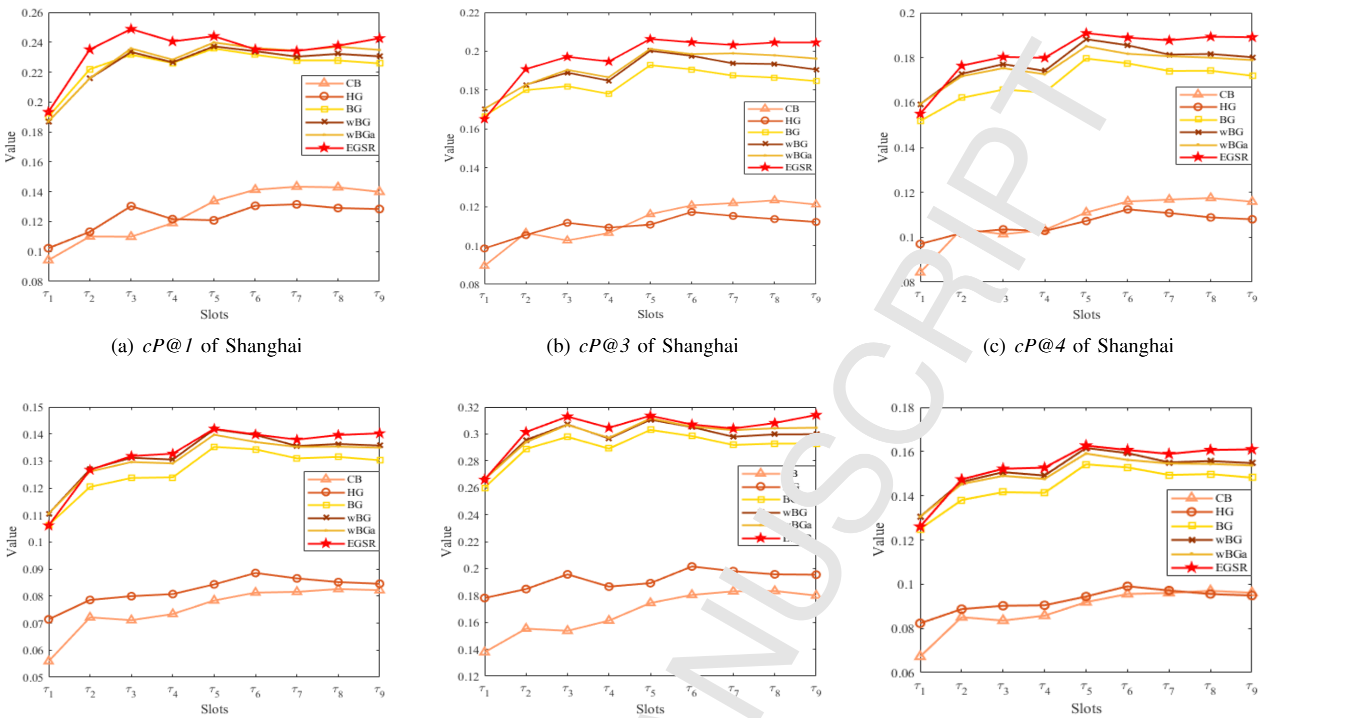
$$C(t) \propto H(U_d^{old_t}) = \sum_{i=1}^{|U_d^{old_t}|} h(u_i), \tag{33}$$

As each user entropy $h(u_i)$ is a constant, the only variable is the number of old users. And it is obvious that the $U_d^{old_{t_i}}$ is the subset of $U_d^{old_{t_j}}$, if $j > i$, which can be proved as follows:

$$\begin{aligned} U_d^{old_{t_j}} &= U_d \cap U_h^{t_j} \\ &= U_d \cap U_h^{t_i} + U_d \cap \sum\nolimits_{k=i}^{j} U_s^{t_k} \\ &= U_d^{old_{t_i}} + U_d \cap \sum\nolimits_{k=i}^{j} U_s^{t_k}, i < j \end{aligned} \tag{34}$$

where $U_h^{t_i}$ is the set of users in history slots from the 1th to the $t_i$th slot and $U_s^{t_k}$ denotes the set of users in the $t_k$th history slot. So if $i < j$, $C(t_i) < C(t_j)$, which means $C(t)$ monotonically increasing with the increment of $t$. ■

*Proof:* Lemma 2: We rewrite the objective function Eq.(7)




(a) *cP@1* of Shanghai (b) *cP@3* of Shanghai (c) *cP@4* of Shanghai

(d) *cRecall* of Shanghai (e) *cMAP* of Shanghai (f) *cF1* of Shanghai

Fig. 5. Shanghai: Experiment results of *cP@n*, *cP@3*, *cP@4*, *cRecall*, *cMAP*, *cF1* in different recommendation slots.

as follows:

$$J(T) = |C(T) - C_{th}| = \begin{cases} C(T) - C_{th}, C(T) > C_{th} \\ C_{th} - C(T), C(T) < C_{th} \end{cases} \quad (35)$$

where $C_{th}$ is a constant.

Deriving the objective function $J(T)$ to $T$, the derivation is defined as:

$$\frac{\partial J(T)}{\partial T} = \begin{cases} \frac{\partial C(T)}{\partial T}, C(T) > C_{th} \\ -\frac{\partial C(T)}{\partial T}, C(T) < C_{th} \end{cases} \quad (36)$$

We have proved that $C(T)$ monotonically increases with the increment of $T$ in Lemma.1, which results in $\frac{\partial C(T)}{\partial T} > 0$. Then we can find a number $\theta$ which lets $C(\theta) = C_{th}$. So if $T < \theta$, $J(T)$ decreases with the increment of $T$. Otherwise, if $T > \theta$, $J(T)$ increases with the increment of $T$. And there is an unique minimum extreme point $C(\theta)$ of $C(T)$. ∎

*Proof:* Lemma 3: we rewrite the objective function Eq. (29) as follows:

$$\bar{\gamma}_\tau = \sum_{k=1}^{K} \frac{k}{K} \times \frac{\binom{K}{k}\binom{N_\tau - K}{K-k}}{\binom{N_\tau}{K}}, \quad (37)$$

Let $N_\tau$ be the independent variable of this function. Then we define a function $D(N_\tau)$ as

$$D(N_\tau) = \frac{\binom{K}{k}\binom{N_\tau - K}{K-k}}{\binom{N_\tau}{K}}, \quad (38)$$

So the function of $D(N_\tau)/D(N_\tau - 1)$ can be simplified as

$$\begin{aligned} \frac{D(N_\tau)}{D(N_\tau - 1)} &= \frac{\binom{K}{k}\binom{N_\tau - K}{K-k}\binom{N_\tau - 1}{K}}{\binom{N_\tau}{K}\binom{K}{k}\binom{N_\tau - K - 1}{K-k}} \\ &= \frac{[(N_\tau - K)!]^2 (N_\tau - 1)!(N_\tau - 2K + k - 1)!}{[(N_\tau - K - 1)!]^2 N_\tau!(N_\tau - 2K + k)!} \\ &= \frac{N_\tau^2 - 2KN_\tau + K^2}{N_\tau^2 - 2KN_\tau + N_\tau k}, \end{aligned} \quad (39)$$

In Eq (39), owing to $N_\tau \gg K$, so $0 < D(N_\tau)/D(N_\tau - 1) < 1$, which means that $D(N_\tau)$ increase as $N_\tau$ decrease. Then we rewrite Eq.(37) as follows:

$$\bar{\gamma}_\tau = \sum_{k=1}^{K} \frac{k}{K} \times D(N_\tau), \quad (40)$$

This function shows that $\bar{\gamma}_\tau$ has the same monotonicity as $D(N_\tau)$, which means $\bar{\gamma}_\tau$ also monotonically decreases with the increment of $N_\tau$. ∎

## References

[1] J. Zeng, L. T. Yang, and J. Ma, "A system-level modeling and design for cyber-physical-social systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 2, p. 35, 2016.

[2] H. Ning, H. Liu, J. Ma, L. T. Yang, and R. Huang, "Cybermatics: Cyber–physical–social–thinking hyperspace based science and technology," *Future generation computer systems*, vol. 56, pp. 504–522, 2016.

[3] Q. Zhang, L. T. Yang, Z. Chen, P. Li, and F. Bu, "An adaptive droupout deep computation model for industrial iot big data learning with crowdsourcing to cloud computing," *IEEE Transactions on Industrial Informatics*, 2018.

[4] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A double deep q-learning model for energy-efficient edge scheduling," *IEEE Transactions on Services Computing*, 2018.

[5] D. Zhang, D. Zhang, H. Xiong, L. T. Yang, and V. Gauthier, "Nextcell: predicting location using social interplay from cell phone traces," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 452–463, 2015.

[6] J. Wu, M. Dong, K. Ota, J. Li, and Z. Guan, "Fcss: Fog computing based content-aware filtering for security services in information centric social networks," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2018.

[7] B. Feng, Q. Fu, M. Dong, D. Guo, and Q. Li, "Multistage and elastic spam detection in mobile social networks through deep learning," *IEEE Network*, vol. 32, no. 4, pp. 15–21, 2018.

[8] H. Peng, M. Bao, J. Li, M. Z. A. Bhuiyan, Y. Liu, Y. He, and E. Yang, "Incremental term representation learning for social network analysis," *Future Generation Computer Systems*, vol. 86, pp. 1503–1512, 2018.

[9] W. Jiang, G. Wang, M. Z. A. Bhuiyan, and J. Wu, "Understanding graph-based trust evaluation in online social networks: Methodologies and challenges," *ACM Computing Surveys (CSUR)*, vol. 49, no. 1, pp. 10:1–10:35, 2016.

[10] M. A. Rahman, V. Mezhuyev, M. Z. A. Bhuiyan, S. N. Sadat, S. A. B. Zakaria, and N. Refat, "Reliable decision making of accepting friend request on online social networks," *IEEE Access*, vol. 6, pp. 9484–9491, 2018.

[11] J. Bao, Y. Zheng, D. Wilkie, and M. F. Mokbel, "A survey on recommendations in location-based social networks," *ACM Transaction on Intelligent Systems and Technology*, 2013.

[12] P. Kefalas, P. Symeonidis, and Y. Manolopoulos, "A graph-based taxonomy of recommendation algorithms and systems in lbsns," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 604–622, 2016.

[13] Y. Jhamb and Y. Fang, "A dual-perspective latent factor model for group-aware social event recommendation," *Information Processing & Management*, vol. 53, no. 3, pp. 559–576, 2017.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[15] K. Xu, Y. Cai, H. Min, X. Zheng, T. Wong *et al.*, "Uis-lda: a user recommendation based on social connections and interests of users in uni-directional social networks," in *WI '17 Proceedings of the International Conference on Web Intelligence*. ACM, 2017, pp. 260–265.

[16] X. Liu, P. Yin, M. Z. A. Bhuiyan, and G. Wang, "The strength of dithering in recommender system," in *2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*. IEEE, 2017, pp. 71–78.

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

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

[19] F. Xia, Z. Chen, W. Wang, J. Li, and L. T. Yang, "Mvcwalker: Random walk-based most valuable collaborators recommendation exploiting academic factors," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 364–375, 2014.

[20] 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.

[21] B. Shams and S. Haratizadeh, "Graph-based collaborative ranking," *Expert Systems with Applications*, vol. 67, pp. 59–70, 2017.

[22] H. Cheng, P. N. Tan, J. Sticklen, and W. F. Punch, "Recommendation via query centered random walk on k-partite graph," in *7th IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 457–462.

[23] X. Deng, G. Li, M. Dong, and K. Ota, "Finding overlapping communities based on markov chain and link clustering," *Peer-to-Peer Networking and Applications*, vol. 10, no. 2, pp. 411–420, 2017.

[24] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A tensor-train deep computation model for industry informatics big data feature learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3197–3204, 2018.

[25] Q. Yuan, G. Cong, and A. Sun, "Graph-based point-of-interest recommendation with geographical and temporal influences," in *CIKM '14 Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 659–668.

[26] C. Shi, B. Hu, X. Zhao, and P. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering (Early Access)*, 2018.

[27] C. Musto, P. Basile, P. Lops, M. de Gemmis, and G. Semeraro, "Introducing linked open data in graph-based recommender systems," *Information Processing and Management*, vol. 53, no. 2, pp. 405–435, 2017.

[28] T.-A. N. Pham, X. Li, G. Cong, and Z. Zhang, "A general graph-based model for recommendation in event-based social networks," in *31st IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2015, pp. 567–578.

[29] B. Li, B. Wang, Y. Mo, and L. T. Yang, "A novel random walk and scale control method for event recommendation," in *The 13th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC)*. IEEE, 2016, pp. 228–235.

[30] 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 Generation Computer Systems*, vol. 79, pp. 383–395, 2018.

[31] S. Liu, B. Wang, and M. Xu, "Event recommendation based on graph random walking and history preference reranking," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 861–864.

[32] X. He, M. Gao, M.-Y. Kan, and D. Wang, "Birank: Towards ranking on bipartite graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 57–71, 2017.

[33] H. Bagci and P. Karagoz, "Random walk based context-aware activity recommendation for location based social networks," in *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*. IEEE, 2015, pp. 1–9.

[34] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 635–644.

[35] T. A. Pham, X. Li, G. Cong, and Z. Zhang, "A general recommendation model for heterogeneous networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3140–3153, 2016.

[36] C. Guo and X. Liu, "Automatic feature generation on heterogeneous graph for music recommendation," in *The 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 807–810.

[37] H. Bagci and P. Karagoz, "Context-aware friend recommendation for location based social networks using random walk," in *WWW '16 Companion Proceedings of the 25th international conference companion on world wide web*. International World Wide Web Conferences Steering Committee, 2016, pp. 531–536.

[38] S. Liu, B. Wang, and M. Xu, "Serge: Successive event recommendation based on graph entropy for event-based social networks," *IEEE Access*, vol. 6, pp. 3020–3030, 2018.

[39] J. Song, X. Luo, J. Gao, C. Zhou, H. Wei, and J. X. Yu, "Uniwalk: Unidirectional random walk based scalable simrank computation over large graph," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 992–1006, 2017.

[40] J. Wang and J.-j. Huang, "Lda-rr: A recommendation method based on ratings and reviews," *Computer Science*, vol. 2, p. 045, 2017.

[41] A. Q. Macedo, L. B. Marinho, and R. L. Santos, "Context-aware event recommendation in event-based social networks," in *RecSys '15 Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 123–130.

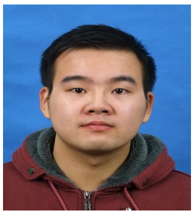
[42] L. Wu, D. Wang, X. Zhang, S. Liu, L. Zhang, and C. W. Chen, "Mllda: Multi-level lda for modelling users on content curation social networks," *Neurocomputing*, vol. 236, pp. 73–81, 2017.

[43] N. Lee, E. Kim, and O. Kwon, "Combining tf-idf and lda to generate flexible communication for recommendation services by a humanoid robot," *Multimedia Tools and Applications*, vol. 77, no. 4, pp. 5043–5058, 2018.

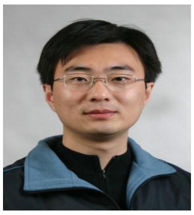
[44] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen, "Adapting to user interest drift for poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 10, pp. 2566–2581, 2016.

[45] F. Zhao, Y. Zhu, H. Jin, and L. T. Yang, "A personalized hashtag recommendation approach using lda-based topic model in microblog environment," *Future Generation Computer Systems*, vol. 65, pp. 196–206, 2016.

[46] M. Dehmer and A. Mowshowitz, “A history of graph entropy measures,” *Information Sciences*, vol. 181, no. 1, pp. 57–78, 2011.
[47] J. Shetty and J. Adibi, “Discovering important nodes through graph entropy the case of enron email database,” in *LinkKDD ’05 Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 74–81.
[48] N. Eagle, M. Macy, and R. Claxton, “Network diversity and economic development,” *Science*, vol. 328, no. 5981, pp. 1029–1031, 2010.
[49] M. Dehmer, “Information processing in complex networks: Graph entropy and information functionals,” *Applied Mathematics and Computation*, vol. 201, no. 1-2, pp. 82–94, 2008.
[50] M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The adaptive web*. Springer, 2007, pp. 325–341.
[51] X. Li and H. Chen, “Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach,” *Decision Support Systems*, vol. 54, no. 2, pp. 880–890, 2013.



**Shenghao Liu** obtained his B.S. from the School of Electronic Information and Communications in Huazhong University of Science and Technology (HUST), Wuhan, China in 2016. He is currently pursuing his Ph.D. degree in the same school. His research focuses on 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 indoor localization, recommendation systems, and social computing.

**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 Social Science Department, National University of Singapore, in 2005 and 2010, respectively. She is currently an Associate Professor with the School of Journalism and Information Communication, HUST. Her research interests include the theory and practice of information diffusion in social networks, recommendation systems and applications, and information communication theories.

**Laurence T. Yang** received his BE degree in Computer Science and Technology from Tsinghua University, China and his Ph.D. degree in Computer Science from University of Victoria, Canada. He is a professor in the School of Computer Science and Technology in 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, cyber-physical-social systems. His research has been supported by the National Sciences and Engineering Research Council, and the Canada Foundation for Innovation.

Highlights:

- Construct evolving graphs based on the mostly recent network information for successive event recommendation
- Propose a graph entropy-based contribution measure to adjust sliding window length and to compute weights for history information
- Propose using content analysis to establish user interest model and compute graph edges' weights
- Conduct experiments based on real EBSN datasets to confirm the superiority of the proposed scheme

ACCEPTED MANUSCRIPT