World Wide Web
DOI 10.1007/s11280-017-0454-0

Recommendations based on a heterogeneous spatio-temporal social network

Pavlos Kefalas[1] · **Panagiotis Symeonidis**[1] · **Yannis Manolopoulos**[1]

Received: 18 September 2015 / Revised: 30 October 2016 / Accepted: 20 March 2017
© Springer Science+Business Media New York 2017

Abstract Recommender systems in location-based social networks (LBSNs), such as Facebook Places and Foursquare, have focused on recommending friends or locations to registered users by combining information derived from *explicit* (i.e. friendship network) and *implicit* (i.e. user-item rating network, user-location network, etc.) sub-networks. However, previous models were static and failed to adequately capture user time-varying preferences. In this paper, we provide a novel recommendation method based on the time dimension as well. We construct a hybrid tripartite (i.e., user, location, session) graph, which incorporates 7 different unipartite and bipartite graphs. Then, we test it with an extended version of the Random Walk with Restart (RWR) algorithm, which randomly walks through the network by using paths of 7 differently weighted edge types (i.e., user-location, user-session, user-user, etc.). We evaluate experimentally our method and compare it against three state-of-the-art algorithms on two real-life datasets; we show a significant prevalence of our method over its competitors.

Keywords Algorithms · Link prediction · Location recommendation · Friend recommendation · Social networks · Big data

✉ Pavlos Kefalas
kefalasp@csd.auth.gr

Panagiotis Symeonidis
symeon@csd.auth.gr

Yannis Manolopoulos
manolopo@csd.auth.gr

[1] Department of Informatics, Aristotle University, Thessaloniki, 54124, Greece

Published online: 19 April 2017

1 Introduction

Users utilize location-based social networks (LBSNs) to share their location with their friends, by incorporating in their posts the longitude and latitude information of their location. In LBSNs, users *explicitly* build a friendship network by adding each other as friends. In addition, users form *implicit* sub-networks through their daily interactions, like commenting on same posts or rating similarly some products/services in places they have co-visited.

Previous works have focused on recommending friends or locations [9, 15] to users by combining information derived from multi-modal and heterogeneous explicit and implicit networks. In particular, there has been extensive research in this area, which mainly focuses on information derived from users' interaction with locations over user-location bipartite network ties. However, such models are static and fail to capture adequately users' preferences as they change over time. That is, time is an important factor in LBNSs, which affects the recommendation accuracy. For example, users periodically perform daily activities in specific locations (e.g. home, work, etc.).

To incorporate the time dimension into their models, Xiang et al. [16] and Yuan et al. [19] proposed the construction of tripartite graphs (i.e., users, locations, sessions) known as *Session-based Temporal Graph (STG)* and *Geographical-Temporal influences Aware Graph (GTAG)*, respectively. Notably, both STG and GTAG do not incorporate edges among nodes of the same set, thus, failing to exploit information from all three unipartite networks (user-user, location-location and session-session). For instance, STG and GTAG do not have links among user nodes. However, intuitively friends tend to visit similar locations at close time points, which means that friendship links could leverage the accuracy of location recommendations. A second problem of STG and GTAG stems from their own structure. That is, GTAG and STG do not connect directly users either with locations or sessions, which results to lower recommendation accuracy when data (i.e., session/location nodes) are sparse.

In this paper, we provide recommendations based on a Heterogeneous Spatio-Temporal graph (HST graph) by incorporating the time dimension into our model. To build this HST graph, we create a new type of an artificial node, called extended session node, which is associated with the co-location of two or more users in a location at a specific time period. Our HST graph incorporates 7 different unipartite or bipartite graphs providing more information in comparison to STG and GTAG. Moreover, we follow a star-schema structure, where users are directly connected with locations and sessions. This structure can be more resistant in cases of sparsity (e.g. when there are not enough session nodes as a result of the fact that users check-into locations at different time periods).

Further, we extend the Random Walk with Restart algorithm (RWR) to run on our HST graph to provide spatio-temporal recommendations. Our extended version of RWR algorithm is called Random Walk with Restart on Heterogeneous Spatio-Temporal (RWR-HST) algorithm. RWR-HST can adequately capture the notion of user-user similarity or the user-location relevance from our HST graph. That is, social drivers which influence the ties formation in communities like homophily, social influence, common friendship, etc. are incorporated by design into the RWR algorithm, as it will be shown later.

The contributions of this paper are summarized as follows:

– We propose the construction of the HST graph, which is a tri-partite graph consisting of 3-disjoint sets of nodes (i.e. sessions, users, locations), and incorporates edges among nodes of the same set, including also three unipartite graphs, i.e. user friendship network, session-session network and location-location network.

- We propose a new variation of the RWR algorithm suitable for friend and location recommendation in multi-modal social networks, called *Random Walk with Restart on Heterogeneous Spatio-Temporal algorithm (RWR-HST)*. Moreover, we provide details about the complexity of the proposed algorithm and discuss other implementation issues in detail (see Section 4.8).
- We compared our method with other state-of-the-art algorithms on two real datasets. It will be shown that our RWR-HST algorithm prevails its predecessors and achieves an significant improvement of 12-29% for friend recommendation and 9-30% for location recommendation against all its competitors, in terms of relative average $F1$ performance (see Section 5.4).

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 presents a formal definition of the examined problem. Section 4 describes the construction of our HST graph, its edge weighting and our proposed algorithm. Experimental results are given in Section 5. Finally, Section 6 concludes the paper.

2 Related work

Time is a crucial factor in LBSNs, since it could leverage the accuracy of friend, location and activity recommendations. Recently, Yuan et al. [18] exploited spatio-temporal characteristics of POIs by using a unified framework consisting of spatial and temporal dimensions.

Gao et al. [5] proposed the *Location Recommendation with Temporal effects* (LRT) algorithm. They argued that the time dimension is crucial in recommendation and introduced a framework to make time-aware recommendations. In the same direction, a time-aware method was proposed by Marinho et al. [10] to improve location recommendations in LBSNs. Ho et al. [7] extracted spatio-temporal information for future events from news articles. Furthermore, Raymond et al. [13] proposed a method to provide location recommendations for users that use buses. Their method was based on users' location histories and spatio-temporal correlations among the locations. By combining collaborative filtering algorithms with link propagation, they were able to predict origins, destinations and arrival times of buses.

In a different direction, Ding et. al [3] presented an algorithm to compute the time weights for different items by assigning a decreasing weight to old data rather than to items recently purchased. In particular, they use a clustering method to distinguish different kinds of items by tracing user's preference dynamics with the use of a decay factor. Similarly, Koren [8] introduced the timeSVD++ algorithm, which captures the lasting and the transient factors by modeling the user preference dynamics through the entire time period. The goal is to distill longer term preferences from the noisy patterns using a matrix factorization model. He showed that in an item-item neighborhood model, the essential relations among the items can be extracted by learning how ratings evolve. Unfortunately, both approaches are out of our scope, since they: (i) concern item rather than Point-Of-Interests (POIs) recommendation, (ii) ignore geographical information, and (iii) ignore social network relations.

The creation of artificial session nodes has been originally proposed by Xiang et al. [16], who designed a framework that models users' long-term and short-term preferences over time. Their model was based on a Session-based Temporal Graph (STG) to incorporate user, location and session information. In addition, Xiang et al. [16] proposed a novel

recommendation algorithm named Injected Preference Fusion (IPF) and extended the personalized Random Walk for temporal recommendation.

Figure 1 shows an example of STG. As shown, there are 2 users, 4 locations and 3 session nodes. User U1 has visited locations L1, L2 and L3, whereas user U2 has visited locations L3 and L4. Notice also that locations L1 and L2 are linked to Session 1 node. This means that both locations (L1 and L2) were co-visited by U1 at the same period T1 (e.g. during the morning of Thursday 19 September 2013). Based on this graph, the user-location bipartite graph denotes the long term preferences of a user, whereas the location-session bipartite graph denotes the short term preferences of a user.

Our work is inspired by the work of Xiang et al. [16]. However, our HST graph has two main differences in comparison with STG:

- The introduction of *session* nodes to connect users (but not locations). That is, user nodes are the heart of a star schema graph and, thus, they are connected via direct links with both location and session nodes
- The graph structure per se is the second difference. It is not only a 3-partite graph that consists of 3-disjoint sets of nodes (i.e. sessions, users, locations). In contrast, it incorporates also edges among nodes of the same set, i.e. three unipartite graphs, which makes it even richer in information.

Recently, Yuan et al. [19] inspired from the STG, proposed the Geographical-Temporal influences Aware Graph (GTAG), which also consists of three entities (i.e., users, sessions and locations). User nodes are connected with session nodes to capture the temporal influences. Moreover, session nodes are connected with location nodes to capture the geographical influence. Notice that, in contrast to our HST graph, GTAG does not capture the underlying similarity among nodes of the same set, i.e. user-user, session-session and location-location. Moreover, they proposed a *Breadth-first Preference Propagation (BPP)* algorithm to walk over GTAG. BPP follows paths based on 3 criteria: (i) there is no repeated node in a path, (ii) the path contains only one visited location and session node of the target user, and (iii) the walk terminates when an unvisited location is met. The main problem of GTAG is its own structure. That is, users are not connected directly with locations, but they are connected through session nodes. Thus, BPP fails to make accurate location recommendations, when there are not enough session nodes (i.e. when users check-in to locations at varying time points).

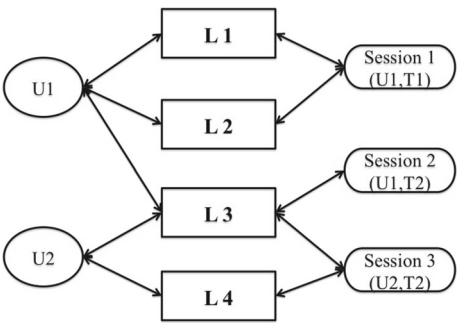

Figure 1 An example of STG

Our approach differs from the GTAG-BPP algorithm at the following points:

- Our framework, builds the graph in a different way, which eventually allows our algorithm to more efficiently exploit the relations among the nodes. In particular, HST connects directly users with locations and users with sessions. This way, the algorithm will not stop if there is no edge between a user node and a session node. In contrast, the algorithm chooses an alternative path, through an edge connecting the user node with either a session or a location node.
- Also, RWR-HST exploits efficiently information from sub-networks such as: (i) friendship, (ii) user-location, and (iii) location-location etc., which GTAT-BPP ignores.
- Finally, HST weighs differently the edges between the nodes.

3 Problem definition

Periodicity and proximity are two main factors in LBSNs. To examine users' check-in periodic behavior, we have incorporated time into our model using *Sessions*, which captures the co-locations among two or more users during a time window. This way, we augment our knowledge about a user through his/hers check-in history. For example, if two users hang out at a bar where a rock band performs live every Friday, then they probably like the same things and they could be friends in the future or they could be interested attending the same venues.

Thus, the problem can be formulated as: given a graph $\mathcal{G}$ which represents the relations among users and locations nodes (i.e. check-in history), we want to recommend new friends and POIs to a target user taking into consideration the auxiliary information derived from the *Session* artificial nodes of our novel Heterogeneous Spatio-Temporal graph (HST graph).

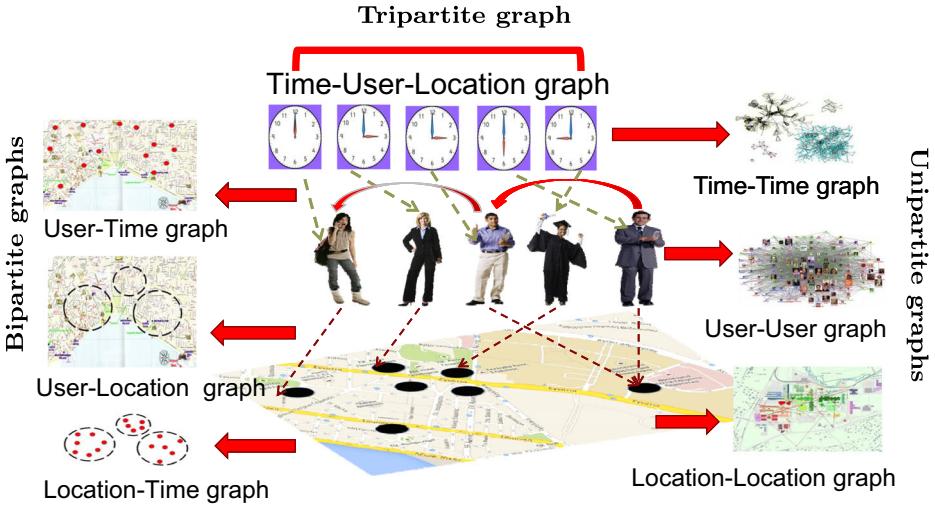

Figure 2 Latent Relations among Time, Users, Locations dimensions of an LBSN and the generated k-partite graphs

4 Background and preliminaries

Here, we introduce the most important notions with the necessary definitions and a motivating example depicted in Figure 2. Also, we provide an analytical description of the basic entities that interact in a LBSN (i.e. users, locations, and time dimension) and discuss the information types that can be extracted from the connections among them. Figure 2 shows the relations among the aforementioned entities. As shown in Figure 2, we have 3 layers (one layer for each entity) and 5 users who have visited some places. For each visit we keep the time of the user's check-in. As also shown in Figure 2, there are 7 graphs of different participating entities (i.e. three unipartite, three bipartite and one tripartite). On the right side of Figure 2, we can see the 3 generated unipartite graphs (Time-Time graph, User-User graph, and Location-Location graph). On the left side of Figure 2, we observe 3 bipartite graphs (User-Time graph, User-Location graph, and Location-Time graph). Finally, on top of Figure 2, we can see the tripartite graph (Time-User-Location graph).

It is necessary to emphasize that the above graph is not a k-partite graph, since there can also exist edges among nodes of the same set (e.g. an edge between a user and another user, i.e. friendship). Henceforth, we denote this special case of a graph as hybrid k-partite graph because it is a k-partite graph that consists of k-disjoint sets of nodes (i.e. time, users, locations), incorporating edges among nodes of the same set as well.

As depicted in Figure 2, our data are in the form of $\langle time, user, location \rangle$ triplets, which are usually modeled by a tripartite graph or a tensor. However, if we had to use a tripartite graph or a tensor for capturing the time dimension as well, then we should create a new node for each different timestamp. This would create a huge tensor or a temporal graph with an enormous number of time nodes introducing severe noise in the model.

Definition 1 (Session node): Based on the above considerations, we choose to create a new type of an artificial node, called *s*ession node, which is associated with the co-location of two or more users (e.g. u_1, ..., u_n) in a location l at a specific time period t. This co-location of two or more users reflects their interest for a place at a specific time.

For example, two users may often visit a specific bar on Friday nights to listen a favorite music band. Thus, the possibility of having both common music interests is very high. That is, two users who visit a specific location at a certain time period have a higher possibility to become friends than those who visit a location but not during the same period.

To create a new session node, in the same direction as in [16], we transform the $\langle user, location, time \rangle$ into $\langle user, location \rangle$ and $\langle user, session \rangle$ by dividing the time into discrete intervals (bins). Then, we associate each bin with corresponding users who have visited a specific place during this bin. Notice that a session node combines a number of users, with a location at a specific time interval. The length of a session can last from one hour, to six hours, or even one day etc. Based on the $\langle user, location \rangle$ and $\langle user, session \rangle$ we create our temporal graph, the HST graph.

4.1 Session node extraction

Users may visit locations all day long. The huge amount of these check-ins, prevent us from understanding their trends and their likes, without before preprocessing the time dimension of this information. To have an abstract and thorough understanding of the users' behavior, we create artificial session nodes based on SQL statements.

Sql Statement 1 SQL query for table creation

```
CREATE TABLE ultime(
        id int NOT NULL,
        UserID int NOT NULL,
        LocationID int NOT NULL,
        tmp datetime DEFAULT NULL
        PRIMARY KEY (id));
```

For our running example, let's assume that we create a table to hold information about users, locations, and the time of their check-ins, as shown in SQL Statement 1. In particular, we have the primary key "id", the field "UserID" for users, the field "LocationID" for locations, and the field "tmp" for the time of the user's check-in.

Sql Statement 2 SQL query for session nodes extraction

```
SELECT A.userID, B.userID, A.Locationid
FROM ultime as A,ultime as B
WHERE A.LocationID = B.LocationID
    AND A.userID <> B.userID
    AND (DATEDIFF(HOUR, A.tmp, B.tmp) / 24=0)
    AND (DATEDIFF(HOUR, A.tmp, B.tmp)
```

We extract the artificial session nodes, by using an SQL statement as shown in SQL Statement 2. This SQL statement finds co-locations between two or more users during the same time period, i.e. a session. In our running example, we set the time window for a co-location of two or more users equal to 6 hours. It is obvious that we can also use other bin lengths (i.e. we can split time into time slots of an hour, a day, a month or a year, depending on the desired session for extraction).

4.2 Constructing the heterogeneous spatio-temporal graph

We define a hybrid 3-partite graph as $\mathcal{G}(S, U, L, \mathcal{E}^{(US)}, \mathcal{E}^{(SU)}, \mathcal{E}^{(UL)}, \mathcal{E}^{(LU)}, \mathcal{E}^{(SS)}, \mathcal{E}^{(UU)}, \mathcal{E}^{(LL)})$, which consists of 3-disjoint sets of nodes ($\mathcal{S}$ for session, $\mathcal{U}$ for user, $\mathcal{L}$ for location). $\mathcal{G}$ is called "hybrid" because it has also edges among nodes of the same set, as shown in Figure 3, where there are edges among users. Similarly, there are edges among sessions and edges among locations. $\mathcal{E}^{(US)}$ represents the edges between nodes in $\mathcal{U}$ and $\mathcal{S}$. Vice versa, $\mathcal{E}^{(SU)}$ represents edges between nodes in $\mathcal{S}$ and $\mathcal{U}$. $\mathcal{E}^{(UL)}$ represents edges between nodes in $\mathcal{U}$ and L, whereas on the other hand $\mathcal{E}^{(LU)}$ represents edges between the nodes in $\mathcal{L}$ and U. $\mathcal{E}^{(SS)}$ represents the edge set linking the nodes in $\mathcal{S}$. $\mathcal{E}^{(UU)}$ represents the edge set linking the nodes in $\mathcal{U}$. Finally, $\mathcal{E}^{(LL)}$ represents the edge set linking the nodes in $\mathcal{L}$. For clarity, in Table 1 we provide a list of all used symbols notations and descriptions.

We assume that the graph $\mathcal{G}$ is directed and weighted. We also assume that the graph $\mathcal{G}$ may have multiple edges connecting two nodes s and u. An example of a hybrid 3-partite HST graph is illustrated in Figure 3, where there are 2 session nodes: session $s_1(l_2|u_1, u_2)$ indicates that user u_1 and user u_2 co-visited location l_1 at the same time period, whereas session $s_2(l_4|u_2, u_3, u_4)$ presents that several users (u_2, u_3 and u_4) co-visited location l_4 at

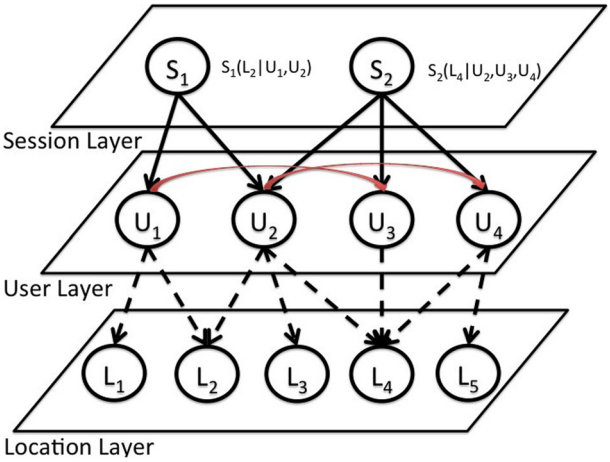

Figure 3 Hybrid 3-partite temporal graph example

the same time period. Notice that, there is a major difference between HST and graphs that consider only the users' relations. In a toy example, someone may want to propose a friend to user u_3. Using only the user-user network of the User Layer it is impossible to provide a friend recommendation because the only available information is the connection with u_1, who is already a friend of u_3. However, by using an auxiliary network (e.g., the user-location

Table 1 Symbols notations and descriptions

Symbol	Description
S	Set of sessions, $S = \{s_1, s_2, ..., s_n\}$
S_u	Set of sessions a user participated
S_l	Set of sessions at a location
s, s'	Some sessions
U	Set of users, $U = \{u_1, u_2, ..., u_n\}$
U_u	Set of users who are friends with user u
U_s	Set of users who participated in a session s
U_l	Set of users who visited a location l
u, u'	Some users
L	Set of locations, $L = \{l_1, l_2, ..., l_n\}$
L_u	Set of locations visited by a user u
l, l'	Some locations
$d_{l,l'}$	Distance between locations l and l'
$\mathcal{E}^{(US)}$	Set of edges linking nodes of U to nodes of S
$\mathcal{E}^{(SU)}$	Set of edges linking nodes of S to nodes of U
$\mathcal{E}^{(UL)}$	Set of edges linking nodes of U to nodes of L
$\mathcal{E}^{(LU)}$	Set of edges linking nodes of L to nodes of U
$\mathcal{E}^{(SS)}$	Set of edges linking the nodes of S
$\mathcal{E}^{(UU)}$	Set of edges linking the nodes of U
$\mathcal{E}^{(LL)}$	Set of edges linking the nodes of L

bipartite network) the recommendation task becomes easier by the fact that u_2 and u_4 have also visited location l_4.

4.3 Edge weighting

In this section, we define the weights between nodes in our HST graph. By incorporating the artificial session nodes into our HST graph, we have the following 7 types of edges, which have to be weighted differently:

- an edge from a session node s to a user node u,
- an edge from a user u to a session s,
- an edge from a user u to a location l,
- an edge from location l to a user u,
- an edge from a user u to another user u',
- an edge from a location l to another location l', and
- an edge from a session s to another session s'.

In the following, we define the edge weights for the 7 different edge types, starting from the edges of the bipartite graphs (session-user and user-location). Firstly, we set the weight $w(s, u)$ of the edge from a session node s to a user node u as:

$$w(s, u) = \frac{1}{|U_s|} \tag{1}$$

where $(|U_s|)$ is the number of users who participated in a session s. Notice that we weight differently an edge that starts from a user u and ends to a session s. In particular, $w(u, s)$ is:

$$w(u, s) = \frac{1}{|S_u|} \tag{2}$$

where $|S_u|$ is the number of sessions in which a user u has participated. That is, the probability of a user to join a session is equally divided on all sessions s/he has participated.

Next, we define the edge weight $w(u, l)$ of the edge from a user node u to a location node l as:

$$w(u, l) = \frac{n_{u,l}}{\sum\limits_{\forall l \in L} n_{u,l}} \tag{3}$$

where $n_{u,l}$ is the number of times a user u visited a location l and $\sum_{\forall l \in L} n_{u,l}$ is the total number of check-ins in all locations by user u. We define the edge weight $w(l, u)$ that starts from location l and ends at a user u as:

$$w(l, u) = \frac{n_{l,u}}{\sum\limits_{\forall u \in U} n_{l,u}} \tag{4}$$

where the $n_{l,u}$ is the number of times a location l is visited by a user u and $\sum_{\forall u \in U} n_{l,u}$ is the total number of check-ins of all users in location l.

We proceed with the edge weighting of the unipartite graphs (user-user, location-location, session-session). First, the edge weight $w(u, u')$ between two user nodes u and u' is defined as the fraction of 1 over the number of users (U_u), who are friends with a user u:

$$w(u, u') = \frac{1}{|U_u|} \tag{5}$$

The edge weight between two location nodes l and l' is defined as:

$$w(l, l') = \left(1 - \frac{(geodist_{l,l'})}{\sum\limits_{\forall l,l' \in L} (geodist_{l,l'})}\right) \tag{6}$$

In this case, we set as link weight the geographical distance between two location nodes l and l'. To obtain all weights, we calculate the distance between all pairs of locations.

Finally, for the edge weighting between two session nodes s and s', we take into consideration both the location and the time dimensions of each session nodes after normalizing both dimensions, by using the following equation:

$$w(s, s') = \left(1 - \frac{(geodist_{s,s'})}{\sum\limits_{\forall s,s' \in S} (geodist_{s,s'})}\right) + \left(1 - \frac{(timediff_{s,s'})}{\sum\limits_{\forall s,s' \in S} (timediff_{s,s'})}\right) \tag{7}$$

where $geodist_{s,s'}$ and $timediff_{s,s'}$ are the geographical distance and the time difference between two session nodes s and s', respectively.

4.4 Construction of the transition probability matrix

Random walk processes on graphs have been extensively used in social network analysis [11, 17]. To apply a random walk on a heterogeneous spatio-temporal graph, we have to construct a transition probability P matrix to configure and set all transition probabilities among the nodes of our HST graph. To represent all possible transitions on the HST graph, the size of the P matrix should be $(|\mathcal{S}|+|\mathcal{U}|+|\mathcal{L}|) \times (|\mathcal{S}|+|\mathcal{U}|+|\mathcal{L}|)$. By combining (1)-(7), we compute the transition probability matrix P which comprises of several sub-matrices that correspond to our HST graph, as follows:

$$\mathbf{P} = \begin{bmatrix} \mathcal{SS} & \mathcal{SU} & 0 \\ \mathcal{US} & \mathcal{UU} & \mathcal{UL} \\ 0 & \mathcal{LU} & \mathcal{LL} \end{bmatrix} \tag{8}$$

where $\mathcal{SS}$ is a $|\mathcal{S}| \times |\mathcal{S}|$ sub-matrix representing the transition probability between session nodes to session nodes, as defined in (7). $\mathcal{UU}$ is a $|\mathcal{U}| \times |\mathcal{U}|$ sub-matrix, which is not symmetric because transition probabilities between two user nodes are defined based on the number of neighbors of each user node (see (5)). $\mathcal{LL}$ is a $|\mathcal{L}| \times |\mathcal{L}|$ sub-matrix representing the transition probability from location nodes to location nodes, as defined in (6). $\mathcal{US}$ sub-matrix holds the transition probabilities from user nodes to session nodes, whereas $\mathcal{SU}$ sub-matrix holds the transition probabilities from session nodes to user nodes. Similarly, $\mathcal{UL}$ sub-matrix holds the transition probabilities from user nodes to location nodes, whereas $\mathcal{LU}$ sub-matrix holds the transition probabilities from location nodes to user nodes. Finally, matrix P is normalized and the sum of edge weights of nodes equals to 1 for all nodes.

4.5 Normalization

In Section 4.3 we described the edge weighting among nodes of our HST graph in both unipartite and bipartite sub-networks. We aimed to assign weights on edges in the interval [0,1]. All these weights will be inserted in a probability transition matrix P, and then we will run our method for capturing the notion of similarity between the nodes of the HST graph. However, in several cases the distribution of the weight values in the interval [0,1]

between the 7 edge types (i.e. session-user, user-user, etc.) differs significantly. For example, consider the case that the most weights in $\mathcal{E}^{(US)}$ are normally distributed between 0 and 0.1, whereas most similarity values of $\mathcal{E}^{(LL)}$ are normally distributed between 0.9 and 1. That is, the weighting values of $\mathcal{E}^{(US)}$ will always be dominated by those of $\mathcal{E}^{(LL)}$.

To avoid this problem, we present a normalization step for the construction of the final transition probability P matrix:

- we compute the mean similarity value m_P of the matrix P.
- we compute the standard deviation value s_P of the matrix P.
- for each (i, j) cell of the P matrix, where $i \neq j$, we apply the transformation:

$$P(i, j) = \frac{P(i, j) - m_P}{s_P} \tag{9}$$

- we scale and translate the derived values back into the interval [0,1]:

$$P(i, j) = \frac{P(i, j) - min_P}{max_P - min_P} \tag{10}$$

  where min_P, max_P are the minimum and maximum values of matrix P after the transformation by (9), respectively.
- finally, we normalize our matrix P according to *Theorem 1.1* presented in [2]. In particular, we convert our transition probability matrix P to a stochastic matrix so that the values of each column of our transition probability matrix P sum up to 1. Thus, given our transition matrix P with positive eigenvalue r and a positive maximal eigenvalue we apply (11):

$$P = D^{-1} \cdot r^{-1} \cdot P(i, j) \cdot D \tag{11}$$

  where $D = diag\{x_1, x_2, ..., x_n\}$ is a diagonal matrix with size equal to the size of the dimensions of the probability matrix P.

4.6 Random walk on the normalized transition probability matrix

The Random Walk with Restart (RWR) algorithm [12, 14] is a variation of the well-known PageRank algorithm. RWR has properties, which can adequately capture the notion of user-user similarity or the user-location relevance for a specific user u of our HST graph. The main advantage of RWR over PageRank is its teleporting characteristic, which obliges the random walker to re-start his walk from the initial node u. As expected, RWR assigns more importance/similarity to the nearby nodes of u. Thus, if two users are close to each other, the probability of becoming friends is larger. Moreover, RWR can capture the notion of similarity among users who share a large number of common friends. For the user-location graph, if two users visit the same locations, then the overall probability for connecting them (via a location node) increases. The same holds for two users via a session node.

RWR considers one random walker starting from an initial user node u and randomly choosing among the available edges with a probability α. In addition, each time the random walker may return back to the initial node with a probability $1 - \alpha$. Therefore, the random walk process can be represented as:

$$S^{(UU)}(t + 1) = \alpha \cdot P \cdot S^{(UU)}(t) + (1 - \alpha) \cdot I \tag{12}$$

where $S^{(UU)}(t)$ and $S^{(UU)}(t+1)$ are the state probability matrices at time t and t+1, respectively. $S^{(UU)}$ is a matrix that represents the link relevance from all HST graph nodes to

the target user u. Parameter a is the prior probability that the random walker will leave its current state. Moreover, I is the identity and P the transition-probability matrix.

4.7 Network contribution adjustment

Here, we incorporate different weighting strategies in our method, which are essential to effectively control the contribution of each sub-network to the final similarity among users. Our main task is to recommend new friends/locations to a target user by exploiting both explicit and implicit sub-networks (i.e., user-user, user-location, user-session, etc.). As discussed in the previous, our HST graph consists of 7 different types of edges. In Section 4.6, we ran RWR without having balanced the ratio among the 7 types of edges. However, if we would like to promote the information derived from the unipartite friendship user-user network and simultaneously reduce the contribution of other sub-networks (e.g. bipartite user-location and user-session networks), then we should have embedded parameters to adjust the contribution of each network.

In this context, the transition probability matrix P transforms to:

$$\mathbf{P} = \begin{bmatrix} \beta \cdot \mathcal{SS} & \frac{\gamma}{2} \cdot \mathcal{SU} & 0 \\ \frac{\gamma}{2} \cdot \mathcal{US} & \delta \cdot \mathcal{UU} & \frac{\epsilon}{2} \cdot \mathcal{UL} \\ 0 & \frac{\epsilon}{2} \cdot \mathcal{LU} & \zeta \cdot \mathcal{LL} \end{bmatrix} \tag{13}$$

where $0 \leq \beta, \gamma, \delta, \epsilon, \zeta \leq 1$ are the trade-off parameters controlling the contribution of each type of the 7 networks to the final similarity. Notice that, for $\mathcal{UL}$ and $\mathcal{LU}$ sub-networks, we use the same parameter. The same stands for $\mathcal{US}$ and $\mathcal{SU}$ sub-networks. The reason is that we have already used a different weighting strategy for each sub-network, as explained in Section 4.3.

It is apparent that the friendship network is very important to provide friend recommendations within the friendship domain. Notably, the contribution of the user-location and user-session networks could be proven helpful, albeit in some cases noisy as well. The trade-off parameters will be examined experimentally and should be adjusted by either learning the dynamics of the networks, or be limited in a specific range according to the recommendation domain. In a real recommendation system, users may be able to self-adjust the contribution of an auxiliary information source to the received recommendations.

4.8 Random walk on the edge weighted HST graph

In this section, we will describe how the walk of a random surfer (i.e. (12)) differs when we vary edge weights for different blocks of the transition probability matrix P (see Section 4.7). Based on the different weighting strategy, we can assign different contribution to each unipartite and bipartite network. Our main purpose is to find out the gain we get when we add auxiliary information to the original friendship network. Next, we enrich our knowledge about a user by incorporating into the initial friendship network a new auxiliary network at each time.

When we take into consideration only the user-user unipartite network, then (12) is transformed to:

$$S^{(UU)}_{Friends}(t+1) = \alpha \cdot \mathcal{UU} \cdot S^{(UU)}_{Friends}(t) + (1-\alpha) \cdot I \tag{14}$$

Equation (14) contains only information derived from the social ties among the users. That is, (14) recommends friends using information only from the friendship network.

When, in addition, we take into consideration the user-location bipartite network, then (14) is augmented as:

$$S^{(UU)}_{Friends-Checkins}(t+1) = \alpha \cdot \left(\left(\frac{\epsilon}{2} \cdot \mathcal{UL}\right) \cdot \left(\frac{\epsilon}{2} \cdot \mathcal{LU}\right) + \delta \cdot \mathcal{UU}\right) \cdot \\ \cdot S^{(UU)}_{Friends-Checkins}(t) + (1-\alpha) \cdot I \tag{15}$$

At this point, notice that in (15), we use a different weighting parameter for $\mathcal{LU}$ and $\mathcal{UL}$ paths. Similarly, if we also take into account the user-session bipartite network, then the later equation is augmented to:

$$S^{(UU)}_{Friends-Sessions}(t+1) = \alpha \cdot \left(\left(\frac{\epsilon}{2} \cdot \mathcal{UL}\right) \cdot \left(\frac{\epsilon}{2} \cdot \mathcal{LU}\right) + (\delta \cdot \mathcal{UU}) + \left(\frac{\gamma}{2} \cdot \mathcal{US}\right) \cdot \left(\frac{\gamma}{2} \cdot \mathcal{SU}\right)\right) \cdot \\ \cdot S^{(UU)}_{Friends-Sessions}(t) + (1-\alpha) \cdot I \tag{16}$$

Finally, if we take into account all 7 networks together (i.e., user-location, location-location, location-user, user-user, user-session, session-session, and session-users), then the later equation is augmented to:

$$S^{(UU)}_{Friends-Sessions-Checkins}(t+1) = \alpha \cdot \left(\left(\frac{\epsilon}{2} \cdot \mathcal{UL}\right) \cdot (\zeta \cdot \mathcal{LL}) \cdot \left(\frac{\epsilon}{2} \cdot \mathcal{LU}\right) + (\delta \cdot \mathcal{UU}) \right. \\ \left. + \left(\frac{\gamma}{2} \cdot \mathcal{US}\right) \cdot (\beta \cdot \mathcal{SS}) \cdot \left(\frac{\gamma}{2} \cdot \mathcal{SU}\right)\right) \cdot S^{(UU)}_{Friends-Sessions-Checkins}(t) + (1-\alpha) \cdot I \tag{17}$$

where $S^{(UU)}(t)$ and $S^{(UU)}(t+1)$ are the state probability matrices at time points t and t+1, respectively. The probability of moving to another node is represented as α, whereas the probability of returning to the initial node is $1-\alpha$. Moreover, I is the identity matrix having the dimensions of the user-user network and $\mathcal{UL}, \mathcal{LU}, \mathcal{UU}, \mathcal{US}, \mathcal{SU}, \mathcal{SS}$, and $\mathcal{LL}$ are the transition-probability block matrices, respectively.

4.9 Recommending locations to a user

To derive location recommendations for a target user u at a time point t, we take into account the check-in history of his k most similar users. More specifically, let $s_1, s_2, \ldots, s_k$ the corresponding final similarity values of the k most similar users $u_1, u_2, \ldots, u_k$ to u (those values have already been calculated with $S^{(UU)}$ similarity matrix, i.e. $s_i = S^{(UU)}(u, u_i)$) and $r_{u_i,t,l}$ the frequency of times that user u_i visited a location l at a time point t. Then, the probability that a user u will check-in a location l at a specific time t is:

$$\widehat{r}_{u,t,l} = \frac{\sum_{i=1}^{k} s_i \cdot r_{u_i,t,l}}{\sum_{i=1}^{k} s_i} \tag{18}$$

If some $r_{u_i,t,l}$ are not defined into the user-location-time $ultime$ table (i.e. the user u_i has not visited location l at time point t), then the corresponding terms into the summation of (18) are deleted. Finally, we sort the predicted locations $\hat{r}_{u,t,l}$ of user u and we suggest the top-N locations, where N is a desired cardinality value.

4.10 The proposed algorithm

The pseudocode of our method is given in Algorithm 1. Our RWR-HST algorithm provides both friend and location recommendations for a target user at a target time point.

Algorithm 1: RANDOM WALK WITH RESTART ON HETEROGENEOUS SPATIO-TEMPORAL GRAPH (RWR-HST)

Input: $u, time$: the target user and the time point that he asks for a recommendation
UU: User-User friendship network
$ultime$: a table that holds the users, locations,
and the time of their check-in
dur: the duration of time for a session
t: number for steps for algorithm convergence,
N: number of recommendations
Output: F: a list of recommended friends to user u
E: a list of recommended locations to user u
Create artificial session nodes using **SQL Statement** 2 based on parameters dur and $time$
Construct the 3-partite HST graph $\mathcal{G}(S, U, L, \mathcal{E}^{(US)}, \mathcal{E}^{(SU)}, \mathcal{E}^{(UL)}, \mathcal{E}^{(LU)}, \mathcal{E}^{(SS)}, \mathcal{E}^{(UU)}, \mathcal{E}^{(LL)})$, where $\mathcal{S}, \mathcal{U}$ and $\mathcal{L}$ are the set of Sessions, Users and Locations, respectively.
Weight the edges between the nodes of $\mathcal{G}$ by using Equations 1 - 7
Construct the transition probability matrix **P** and normalize it using Equations 9 - 10
$difference$=1
t=1
repeat
  Compute $S^{(UU)}_{Friend-Sessions-Checkins}(t+1)$ based on Equation 17
  t=t+1
  $difference = S^{(UU)}_{Friend-Sessions-Checkins}(t+1) - S^{(UU)}_{Friend-Sessions-Checkins}(t)$
until *difference* < 0.00001
Find the top-N similar users to u and put them in F.
foreach *location* l *in* $ultime$ *so that* $r_{u,t,l}$ *is NULL* **do**
  compute $\hat{r}_{u,t,l}$ using Equation 18
  add $(l, \hat{r}_{u,t,l})$ to E
Sort E on $\hat{r}_{u,t,l}$ and retain the top-N locations for u.
return F, E

The creation of artificial session nodes and the Heterogeneous Spatio-Temporal (HST) graph are described in lines 1-2. Line 3 presents the weighting procedure of the edges of the HST graph. Line 4 describes the construction of the transition probability matrix P and its normalization. Lines 7-11 compute the user-user similarity matrix S^{UU} based on (17). Then, we provide friend recommendation in line 12. Lines 13-15 show how the location recommendations are computed. In particular, the predicted $\hat{r}_{u,t,l}$ (line 14) is computed using (18). The locations with the estimated values are accommodated in a list (line 15), and the top-N predicted location values are returned for recommendation (lines 15-17). In

other words, the locations and their estimated values are added to set E (line 15), which is then sorted on predicted value, retaining only the top-N positions (line 16). The locations at these positions are recommended to the user u (line 17).

4.11 Complexity analysis

Time complexity of our RWR-HST approach may be carried out by using the auxiliary information. Thus, if only the user-user network is considered in our HST graph, then its complexity is $O(t \cdot |\mathcal{E}^{(UU)}| \cdot |U|)$, where t is the number of iterations, $|\mathcal{E}^{(UU)}|$ is the number of user-user edges and $|\mathcal{U}|$ is the number of users. Then, if we add the user-location edges to the graph, its complexity is $O(t \cdot |\mathcal{E}^{(UU)}| + |\mathcal{E}^{(UL)}| \cdot |U|)$, where $|\mathcal{E}^{(UL)}|$ is the number of user-location edges. By including all 7 networks, the final time complexity is $O(t \cdot (|\mathcal{E}^{(UU)}| + |\mathcal{E}^{(UL)}| + |\mathcal{E}^{(US)}| + |\mathcal{E}^{(SU)}||\mathcal{E}^{(LU)}| + |\mathcal{E}^{(SS)}| + |\mathcal{E}^{(LL)}|) \cdot |U|)$.

5 Experimental evaluation

Here, we experimentally compare our RWR-HST approach against with three opponent methods: (i) a fast version of the classic Katz algorithm, denoted as Fast-Katz [4], (ii) the Random Walk with Restart algorithm, denoted as RWR [12, 14], and (iii) the state-of-the-art algorithm, denoted as GTAG-BPP [19]. The parameters used to evaluate the performance of the above algorithms are identical to those reported in the original papers. However, for datasets that were not used in these papers, we tuned the parameters so as to get the best results for all methods.

5.1 Data sets

We performed our experiments using two real-world big datasets, i.e., Foursquare[1] and Gowalla[2]. Foursquare [6] dataset contains 18,107 users 2,073,740 check-ins, 847,081 locations and 231,148 social ties among users. This dataset is collected between March 2010 and January 2011. Notice that we did not use the dataset of our main competitor [19] because it does not incorporate the friendship network. Gowalla [1] dataset concerns 196,591 users who have 950,327 social ties among them (i.e. friendship network). Also, they have performed 6,442,890 check-ins to 1,280,969 locations. This dataset is collected between February 2009 and October 2010.

Detailed information about both networks is illustrated in Table 2. In particular, information about friendship networks can be seen in Table 2a, where we present the type of each network (i.e. directed or undirected), the number of users, the number of links among users, the nodes' Average Degree (ADG) and the Local Clustering Coefficient (LLC). As expected, the sparsity of the user-user matrix is very big, i.e., 99.92% and 99.99% for the Foursquare and for the Gowalla datasets, respectively.

Furthermore, Table 2b contains information about the bipartite user-location network. In this table, we present the number of users, the number of locations, and the number of check-ins. Moreover, parameter AVG_u denotes the average number of check-ins per user, whereas parameter AVG_l denotes the average number of check-ins per location.

[1] http://www.public.asu.edu/hgao16/dataset.html

[2] http://snap.stanford.edu/data/loc-gowalla.html

Table 2 Datasets specifications

(a) Friendship Network

Dataset	Type	Users	Edges	ADG	LCC
Foursquare	undirected	18107	231148	10.5800	0.1841
Gowalla	undirected	196591	950327	4.8	0.2367

(b) User-Location Network

Dataset	User	Location	Check-ins	AVG_u	AVG_l
Foursquare	18107	847081	2073740	101	48.16
Gowalla	196591	1280969	6442890	37.18	3.11

(c) User-Location-Session Network

			Session nodes				
Dataset	Users	POIs	3 Hour	6 Hour	9 Hour	12 Hour	24 Hours
Foursquare	18107	847081	36606	78402	89079	90012	93204
Gowalla	196591	1280969	381697	645982	819441	1002941	1603128

In addition, as shown in Table 2c we have created artificial session nodes to study the effect of the length of a session slot. We have created session nodes based on 3 hours, 6 hours, 9 hours, 12 hours and 24 hours. The average session per user (for sessions of 3 hours) is 2.02 and 1.94 for the Foursquare and the Gowalla dataset, respectively. This means that it is easier to find co-locations among users in both cases since each user participates almost in two sessions (i.e. average co-location with at least two other users). Thus, this will affect positively the recommendation accuracy of all algorithms, as will be experimentally shown later.

In Figure 4 we show statistics on the Foursquare and the Gowalla datasets. Notice that both x-axis and y-axis are in the log scale. As shown, the datasets follow a power law distribution for both the number of users' check-ins (Figure 4a and c) and the number of visits to a particular location (Figure 4b and d). As shown in Figure 4a and c, there is a small number of users who have checked-in to many locations (short head) and many users that have only checked-in a small number of locations (long tail). Similarly, as shown in Figure 4b and d, few locations have many visits, whereas many locations have a small number of visits. Notice that, it is very difficult for all algorithms to recommend accurately locations, which have not been visited from many users (i.e., recommendation in the long tail of the distribution).

Finally, Figure 5 presents the location distribution for both datasets in terms of latitude and longitude on a world map. In particular, Figure 5a depicts the Foursquare dataset distribution and Figure 5b depicts the Gowalla dataset distribution. Please notice that both datasets have similar check-ins distribution.

5.2 Protocol

Here, we describe the experimental protocol followed for the friend and the location recommendation tasks. For the friend recommendation task, we consider the division of friends of each target user into two sets: (i) the training set $\mathcal{E}_u^{\mathcal{T}}$ is treated as known information, and (ii) the probe set $\mathcal{E}_u^{\mathcal{P}}$ is used for testing. It is obvious that, $\mathcal{E}_U = \mathcal{E}_u^{\mathcal{T}} \cup \mathcal{E}_u^{\mathcal{P}}$ and $\mathcal{E}_u^{\mathcal{T}} \cap \mathcal{E}_u^{\mathcal{P}} = \oslash$. Therefore, for a target user we generate the recommendations based only on the friends in $\mathcal{E}_u^{\mathcal{T}}$. For the location recommendation task, we have followed a similar

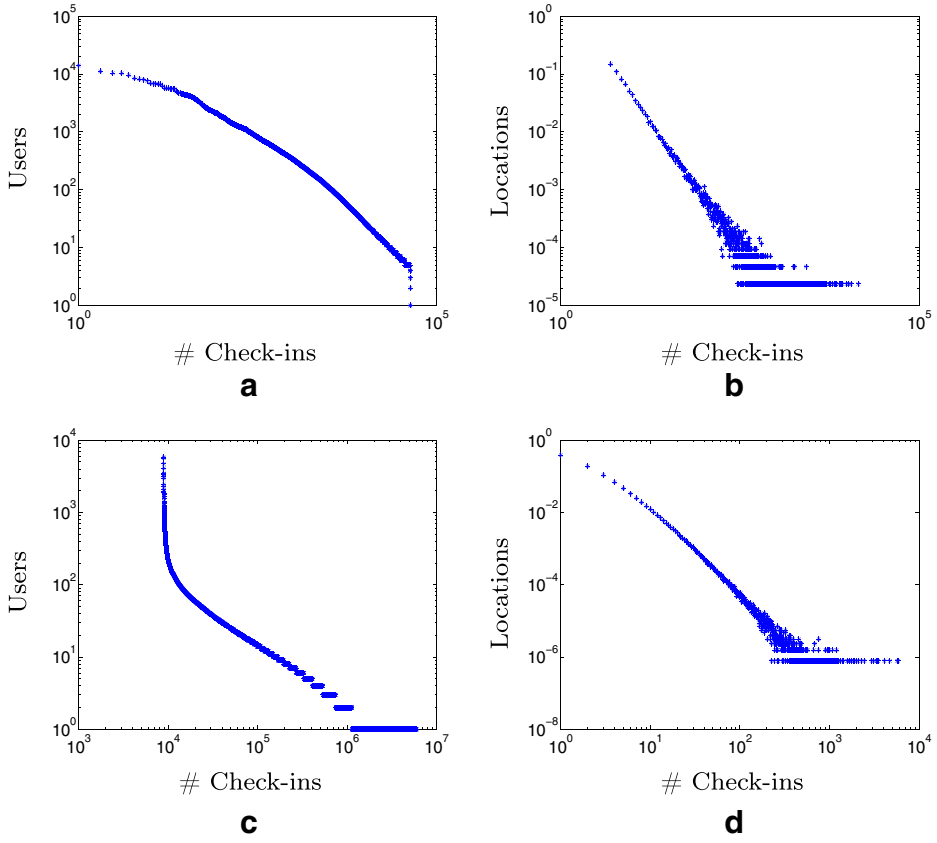

Figure 4 Power Law distribution diagrams for Foursquare [(**a**) and (**b**)] and Gowalla [(**c**) and (**d**)] datasets

procedure. That is, we have also divided the check-ins of each target user into two sets: (i) the training set $\mathcal{E}_C^{\mathcal{T}}$ is treated as known information, and, (ii) the probe set $\mathcal{E}_C^{\mathcal{P}}$ is used for testing.

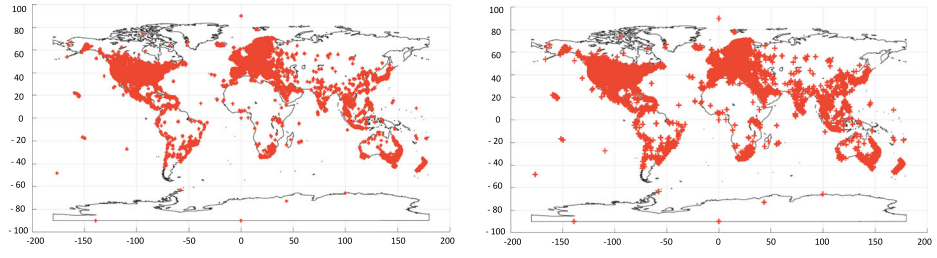

Figure 5 Distribution of locations on a world map for both datasets

Each experiment has been repeated 30 times (each time a different training set is selected at random) and the presented measurements, based on two-tailed t-test, are statistically significant at the 0.05 level. All algorithms have the task either to predict the friends or the locations visited of the target user in the probe sets. We use the classic precision/recall/$F1$ metrics as performance measures for friend and location recommendations. For the friend recommendation task, for a test user receiving a list of N recommended friends (top-N list), *precision* is the ratio of the number of relevant users in the top-N list (i.e. those in the top-N list that belong in the probe set $\mathcal{E}_{\mathcal{U}}^{\mathcal{P}}$ of friends of the target user) to N. On the other hand, *Recall* is the ratio of the number of relevant users in the top-N list to the total number of relevant users (all friends in the probe set $\mathcal{E}_{\mathcal{U}}^{\mathcal{P}}$ of the target user). For the location recommendation task, for a test user receiving a list of N recommended locations (top-N list), *Precision* is the ratio of the number of relevant locations in the top-N list (i.e., those in the top-N list that belong in the probe set $\mathcal{E}_{\mathcal{C}}^{\mathcal{P}}$ of locations of the target user) to N. Also, *Recall* is the ratio of the number of relevant locations in the top-N list to the total number of relevant (all locations in the probe set $\mathcal{E}_{\mathcal{C}}^{\mathcal{P}}$ of the target user), whereas *F1* is the normalized harmonic mean of precision and recall providing an overall picture of the algorithms' performance.

5.3 Sensitivity analysis

In this section, we perform sensitivity analysis of RWR-HST algorithm in terms of precision, when we vary: (i) the training set sizes, (ii) the length of the time window of a session, (iii) the combination of auxiliary sources used, and (iv) the top-N recommended users.

As far as the variation of the training set sizes is concerned, we aim to verify that more information results to better recommendations. For the variation of the length of the time-window of a session, we expect to verify that precision increases as we decrease the length of the time-window. That is, we expect two users to be more similar when they check-in to the same locations at closer time points. For the different combinations of auxiliary sources used, our main purpose is to find out what gain we get, when we add auxiliary information to the original friendship network. Finally, we expect that the precision decreases as we increase the number of recommended users.

Next, we present the sensitivity analysis for the tasks of friend and location recommendations.

5.3.1 Friend recommendation

For the Foursquare dataset, Figure 6a illustrates the precision and the recall vs. different sizes of the train set (20%, 40%, 60%, 80%) when we recommend a top-1 friend to the target users. As expected, as long as the size of the train set increases, precision and recall increase as well. The highest precision value is attained when the train set incorporates 80% of the information. Henceforth, we set the training set size to 80.

Figure 6b shows precision and recall vs. different lengths of the time-window of a session (i.e., 24-hours, 9-hours and 3-hours). As expected, as we decrease the length of time-window of a session, precision increases. The same applies for recall, related to a low false negative rate as the time-window decreases. Thus, the notion of similarity between two users is captured more effectively as we decrease the time-window of a session. That is, two users are more similar when they check-in to the same locations at closer time points.

Figure 6c present precision and recall vs. different combinations of auxiliary sources used (i.e., "F", "F-S", "F-C" and "F-C-S"). Notice that the networks are depicted with abbreviations (i.e. "F" is the Friendship network, "F-S" is Friendship with Session

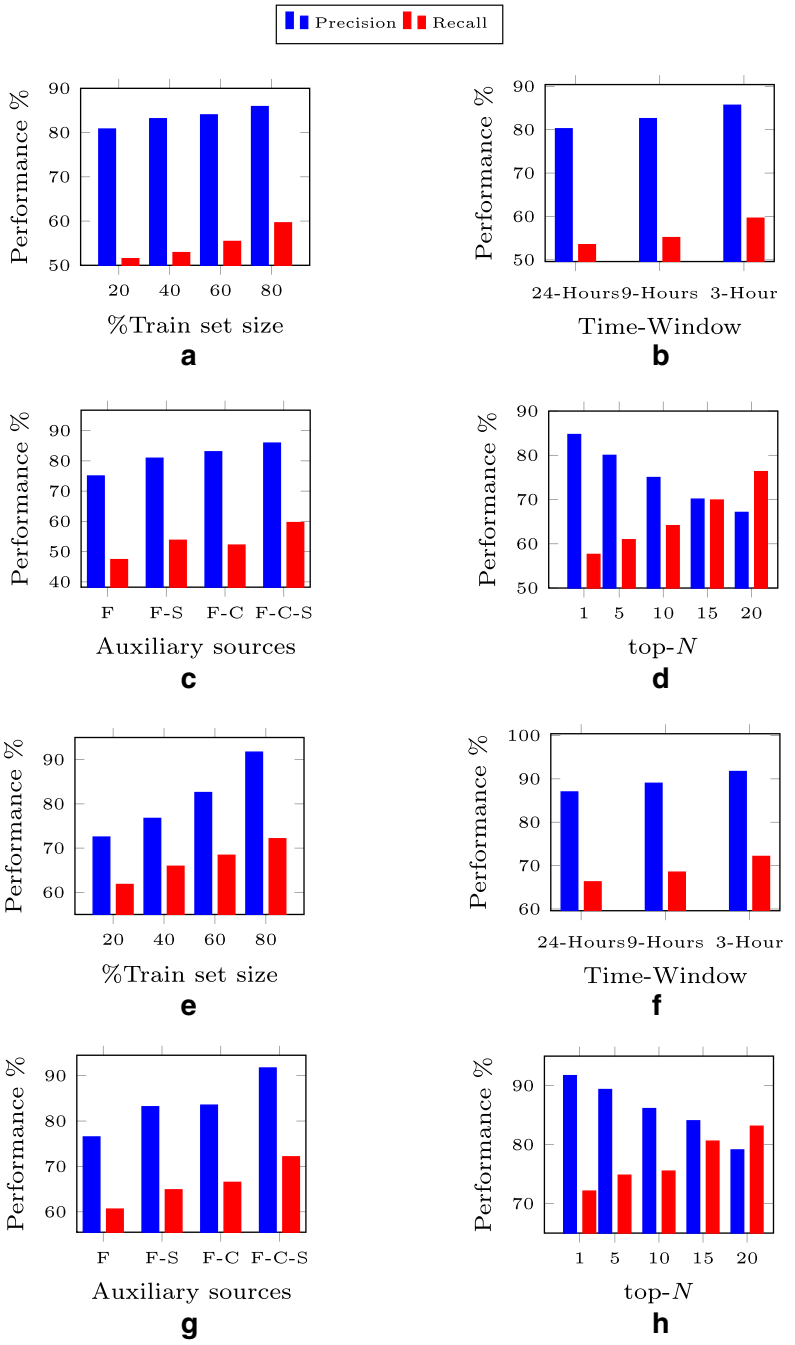

Figure 6 Sensitivity analysis of RWR-HST algorithm for friend recommendations. For the Foursquare dataset, precision and recall vs. (**a**) different training set sizes, (**b**) the length of the time-window, (**c**) auxiliary sources used, and (**d**) top-N recommended users. For the Gowalla dataset, precision and recall vs. (**e**) different training set sizes, (**f**) the length of the time-window, (**g**) auxiliary sources used, and (**h**) top-N recommended users

network, "F-C" is Friendship with Check-ins network and "F-C-S" is Friendship, Check-ins and Session networks, all three together). As shown, precision and recall increase as we add auxiliary sources into our model.

Figure 6d present precision and recall vs. top-N recommended users. As expected, precision performance of RWR-HST gradually decays when we ask for a higher number of recommended users. This is reasonable because precision drops as we increase the number of top-N recommendations, whereas at the same time recall increases.

For the Gowalla dataset, Figure 6e, f, g and h show almost similar results and trends to the ones that are explained previously for the Foursquare dataset (i.e., Figures 6a, b, c and d), respectively. Notice, also, that in both datasets RWR-HST algorithm achieves very high values in terms of precision. As already explained in Section 5.1, the reason is that the more the edges among the nodes exist, the higher the possibility of finding similar users with our target user is. Moreover, as shown in Table 2(c) there were found thousands of co-locations among users for both datasets, which eventually lead to the creation of equal number of artificial nodes.

5.3.2 Location recommendation

In this section, we present the sensitivity analysis for the task of location recommendations. As shown in Figure 7, we test the precision and the recall performance of RWR-HST as we vary: (i) the training set size, (ii) the length of session's time-window, (iii) the auxiliary sources used, and (iv) the top-N recommended users. The results are similar to the ones for the task of friend recommendation. That is, as we gradually vary parameters values, results show a scalar improvement of precision performance, which verifies our initial assumptions. That is, as we increase the size of training set, we enhance known information, which results to a better understanding of users' behavior. Secondly, we have verified that precision increases as we decrease the length of the session's time-window. Moreover, as we add auxiliary information to the original friendship network, we get better recommendations. That is, RWR-HST has more options to walk through the network structure using different paths and edge types. Finally, as we increase the number of top-N recommendations, precision drops for both datasets.

5.3.3 Impact of trade-off parameters

In this section, we tune the trade-off parameters (i.e., $\beta, \gamma, \delta, \epsilon$ and ζ) of RWR-HST and examine their impact. To find the optimal parameter values we tune them over a development set. This set consisted by the 30% of the training set of each dataset, respectively. Each time we tune one parameter, the rest ones remain fixed (i.e. equal to the division of the remaining percentage). We have chosen $F1$ metric to measure the accuracy of our framework because it considers both precision and recall while being computed, as shown in (19). Moreover, since it is assumed to be the weighted average of both mentioned metrics, the closer the values of $F1$ to 100% are, the higher the contribution of particular subnetwork is.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{19}$$

For the friend recommendation task, in Figure 8a and b, we visualize the $F1$ metric vs. different values (i.e., 0.1 to 0.8) of the trade-off parameters for the Foursquare and Gowalla

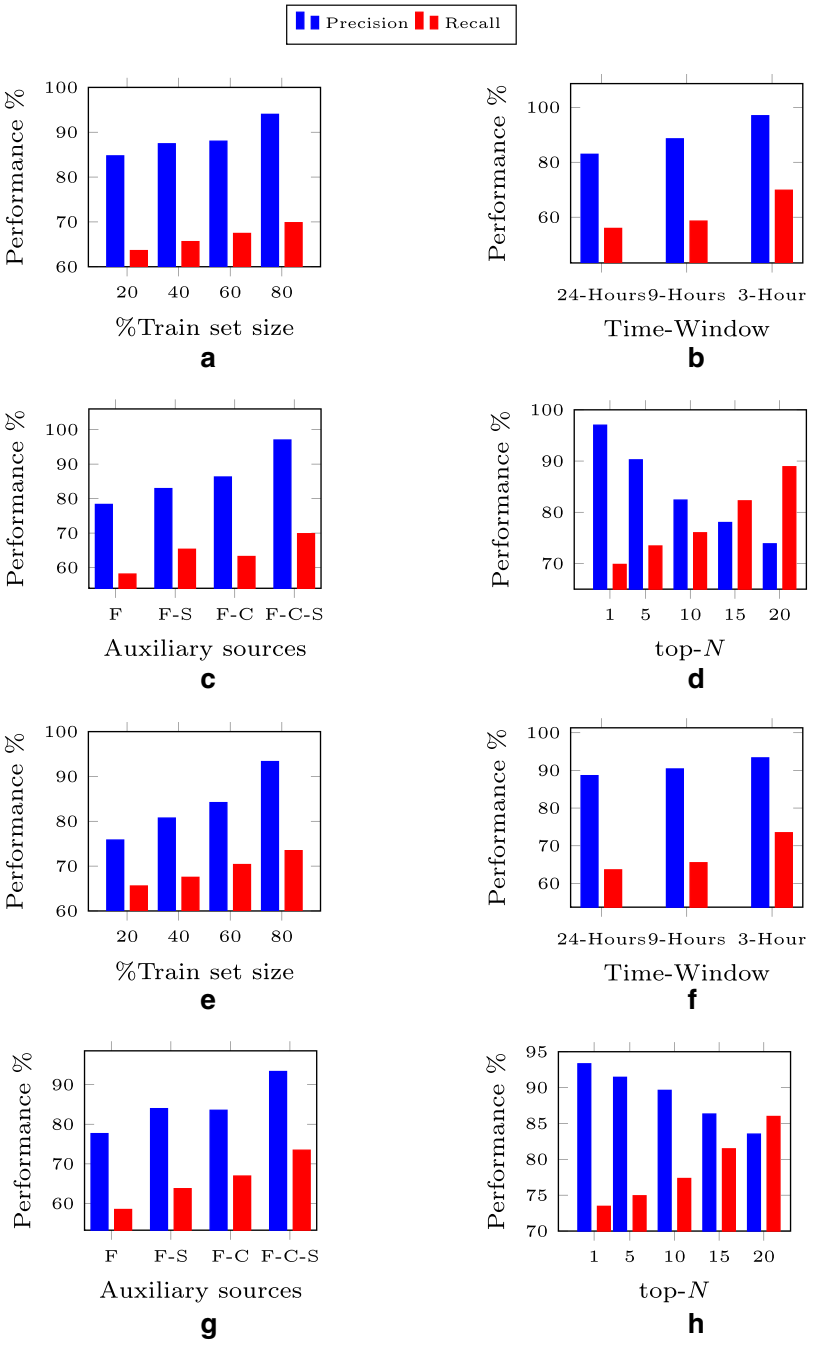

Figure 7 Sensitivity analysis of RWR-HST algorithm for location recommendations. For the Foursquare dataset, precision and recall vs. (**a**) different training set sizes, (**b**) the length of the time-window, **c** auxiliary sources used, and (**d**) top-N recommended users. For the Gowalla dataset, precision and recall vs. (**e**) different training set sizes, (**f**) the length of the time-window, (**g**) auxiliary sources used, and (**h**) top-N recommended users.

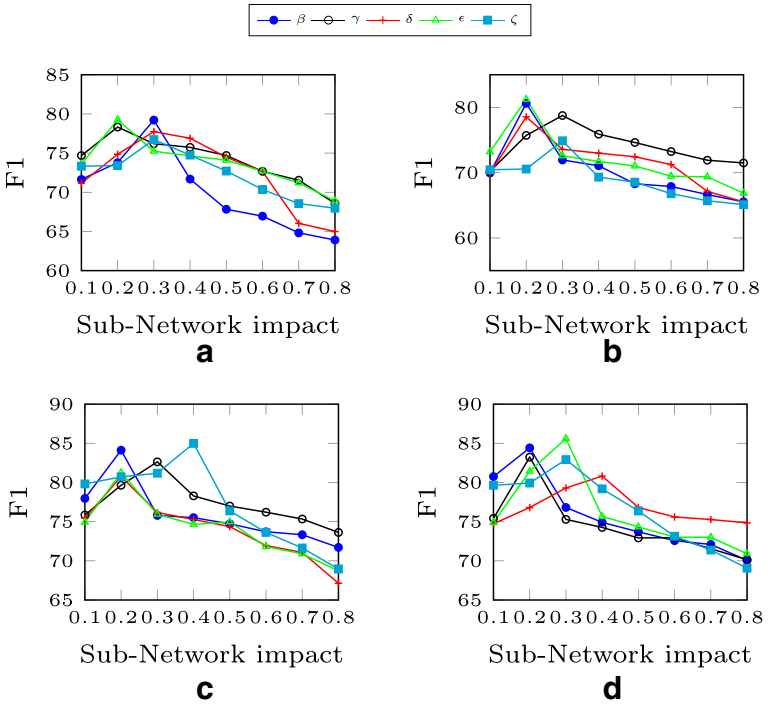

Figure 8 Impact of parameters $\beta, \gamma, \delta, \epsilon$ and ζ for RWR-HST to recommend: (**a**) friend on Foursquare dataset, (**b**) friend on Gowalla dataset, (**c**) location on Foursquare dataset, and (**d**) location on Gowalla dataset

datasets, respectively. Each trade-off parameter, as explained in Section 4.7 controls the contribution of each sub-network. As shown in Figure 8a, the blending of parameters outperforms the case, where we exploit information from only one sub-network. For example, when we exploit only information from the friendship network (by setting δ parameter equal to 0.8), we get $F1$ equal to 65. However, by blending information from all sub-networks, we get $F1$ almost 80. Notice that, the higher the values we achieve indicates the high contribution of the particular subnetwork. Thus, session-session (i.e. β parameter) and user-location (i.e. ϵ parameter) subnetworks can be considered as the more influential sub-networks for leveraging the accuracy for the friend recommendation task for both datasets.

For location recommendations, the results are shown in Figure 8c and d, for the Foursquare and Gowalla datasets, respectively. Again, the blending of information from different sub-networks outperforms the performance of each sub-network separately. While recommending locations using the Foursquare dataset, the session-session (i.e. β parameter) and the location-location (i.e. ζ parameter) sub-networks contribute the most in comparison to the other subnetworks. On the other hand, while recommending locations using the Gowalla dataset the session-session (i.e. β parameter) and user-location (i.e. ϵ parameter) subnetworks can be considered as the more influential sub-networks against the others. In general, information from the aforementioned sub-network can also re-confirmed by the results provided in Figure 6c and g for the task of friend recommendations and Figure 7c and g for the task of location recommendations. Finally, Figure 8 illustrates the final instance of

the tuned parameters, while experimenting with the training and the test set as described in Section 5.2.

5.4 Comparison with other methods

Here, we compare our approach with other three comparison partners i.e. RWR, Fast-Katz and GTAG-BPP, in terms of precision and recall. As the number N of the recommended users/locations varies starting from the top-1 to top-N, we examine the precision and recall scores. Achieving high recall scores while precision follows with the minimum decline indicates the robustness of the examined algorithm.

For the friend recommendation task, in Figure 9a and c, we visualize the precision versus recall curve for the Foursquare and Gowalla datasets, respectively. As N increases, precision falls, while recall increases as expected for all algorithms. RWR-HST demonstrates the best results achieving the highest precision, outperforming all other algorithms. Notice that in terms of precision, we get an average 11.2% and 12% improvement over GTAG-BPP, as shown in Figure 9a and c, respectively. The reason is that RWR-HST exploits effectively information from all sub-networks (i.e., friendship, user-location, etc.).

Moreover, the improvement of RWR-HST over GTAG-BPP for both datasets is due to the fact that our framework builds the graph in a different way, which eventually allow our algorithm to exploit the relations among the nodes more efficiently. In particular, since

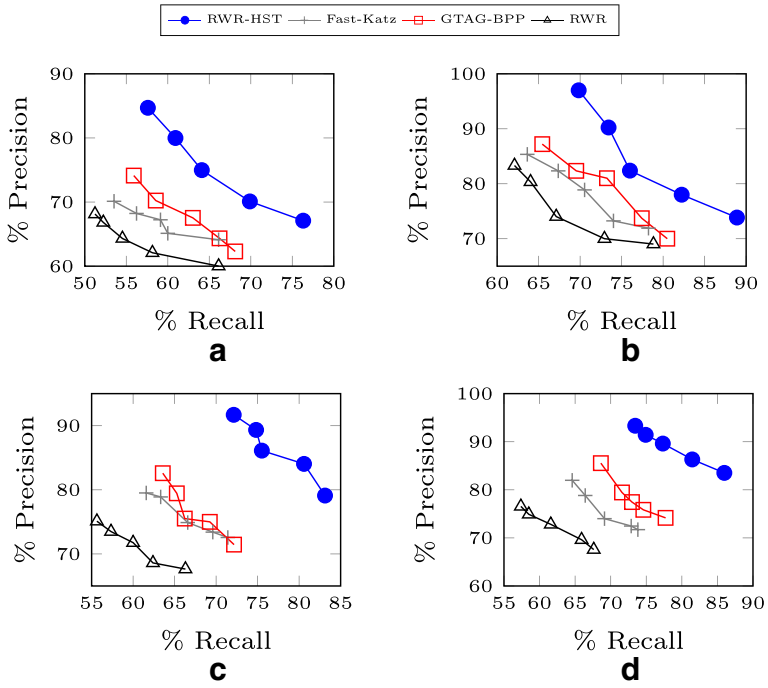

Figure 9 Comparing RWR-HST, Fast-Katz, GTAG-BPP and RWR performance in term of Precision and Recall at top-N recommended (**a**) users on Foursquare dataset, (**b**) locations on Foursquare dataset, (**c**) users on Gowalla dataset, and (**d**) locations on Gowalla dataset

GTAG-BPP uses the artificial session nodes as the only connection path between users and locations, it is very hard to provide accurate recommendations (especially in cases with few session nodes) in contrast to our heterogeneous spatio-temporal graph, which uses the user nodes as the connection path between the sessions and the locations as already explained in Section 4.2.

For location recommendations, we get similar results as shown in Figure 9b and d, for the Foursquare and Gowalla datasets, respectively. Notice, that RWR-HST outperforms again all other algorithms. Again in terms of precision, we get an average 6.8% and 13.3% improvement over GTAG-BPP, as shown in Figure 9a and c, respectively. The reason is that RWR-HST exploits information from more sub-networks than GTAG-BPP. Thus, RWR-HST has more options to walk through the network structure using different paths and edge types. Moreover, RWR-HST is more robust as we increase the number of top-N recommended locations because it achieves high recall scores, whereas precision score drops smoothly.

In summary, Figure 10 shows the average relative performance (in terms of $F1$) of RWR-HST versus GTAG-BPP, Fast Katz and RWR. Each red line inside a box plot represents the average value taken over 30 times of experimental execution. As shown, in all cases the comparison partners attain only a percentage (i.e., in the range 71-91%) of the performance

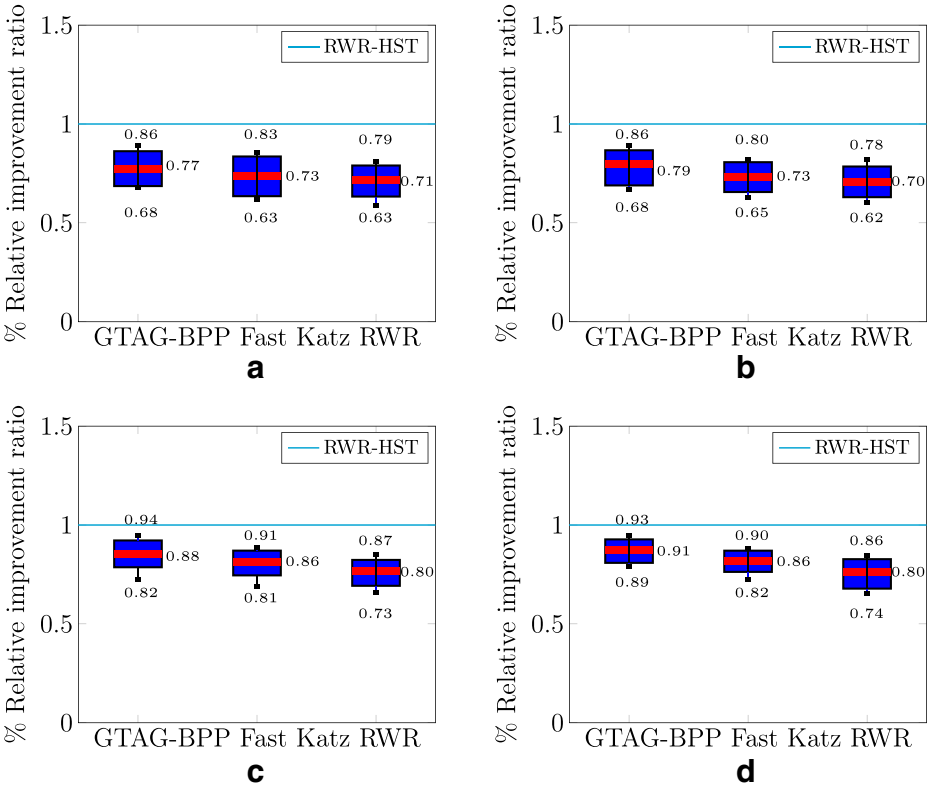

Figure 10 Relative average F1 improvement ratio of RWR-HST versus competitive algorithms recommending (**a**) users on Foursquare dataset, (**b**) location on Foursquare dataset, (**c**) users on Gowalla dataset, and (**d**) location on Gowalla dataset

of RWR-HST, which average relative performance is fixed to 1 (blue line). In particular, with respect to the friend recommendation task we attain an average improvement against GTAG-BPP algorithm between 12-23% for Fourquare and Gowalla datasets, respectively. Similarly, for location recommendation task we attain an average improvement between 9-21% against the same algorithm.

5.5 Comparative results in terms of efficiency

Here, we measure the execution time of our approach in comparison to the other three partners. The time needed for user and location recommendations vs. top-N for both datasets are shown in Figure 11. Regarding the user recommendation task, Figure 11a and c depict a increment of time spend with respect to the number of requested recommendations. For the location recommendation task, we get similar results as shown in Figure 11b and d.

Notice that, RWR-HST algorithm needs less time to provide friend recommendations compared to other methods. The reason is that, it walks only over the edges connecting the session and the location nodes with the target user node. Moreover, regarding the location recommendation task, RWR-HST uses the precomputed results of the friend recommendation and keep walking over the edges connected to it.

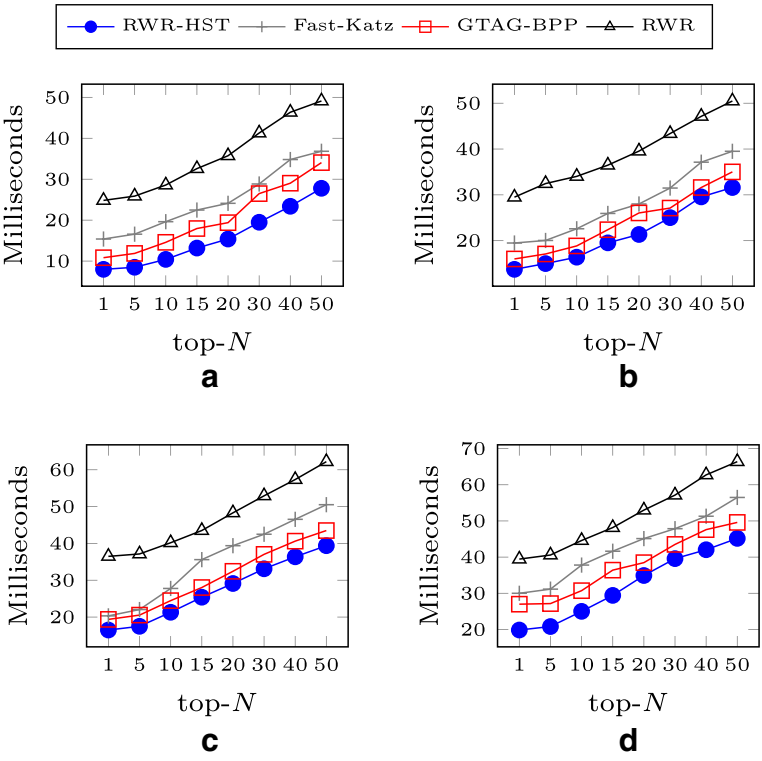

Figure 11 Comparison between algorithms in terms of execution time at top-N recommended (**a**) users on Foursquare dataset, (**b**) locations on Foursquare dataset, (**c**) users on Gowalla dataset, and (**d**) locations on Gowalla dataset

6 Conclusions

In this paper, we proposed a method for friend and location recommendations based on a Heterogeneous Spatio-Temporal graph (HST graph). HST graph incorporates spatial and temporal dimension into a single model. In particular, we have constructed a hybrid tripartite (i.e., user, location, session) graph, which incorporates 7 different unipartite and bipartite graphs. Then, we run on it an extended version of Random Walk with Restart (RWR) algorithm. We have experimented with two real world datasets (i.e., Foursquare and Gowalla) to test the accuracy of our recommendations. We have also compared our algorithm with three state-of-the-art algorithms (i.e., RWR, Fast-Katz, and GTAG-BPP). Results have shown that auxiliary information leverages the accuracy of the final recommendations since the time dimension plays a very important role. As future work, we intent to incorporate into our model information derived from session-location and location-session sub-networks. Another extension could be the adjustment of other algorithms (e.g., SimRank) so that they can benefit from our HST graph.

Acknowledgments This research has benefited from discussions in the working groups of ICT COST Action IC1406 on High-Performance Modeling and Simulation for Big Data Applications (cHiPSet).

References

1. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: User movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pp. 1082–1090 (2011)
2. Chu, M.T., Guo, Q.: A numerical method for the inverse stochastic spectrum problem. SIAM J. Matrix Anal. Appl. **19**(4), 1027–1039 (1998)
3. Ding, Y., Li, X.: Time weight collaborative filtering. In: Proceedings of the 14th ACM International Conference on Information & Knowledge Management (CIKM), pp. 485–492 (2005)
4. Foster, K.C., Muth, S.Q., Potterat, J.J., Rothenberg, R.B.: A faster katz status score algorithm. Computat. Math. Organ. Theory **7**(4), 275–285 (2011)
5. Gao, H., Tang, J., Hu, X., Liu, H.: Exploring temporal effects for location recommendation on location-based social networks. In: Proceedings of the 7th ACM Conference on Recommender Systems (RecSys), pp. 93–100 (2013)
6. Gao, H., Tang, J., Liu, H.: Exploring social-historical ties on location-based social networks. In: Proceedings of the 6th International AAAI Conference on Weblogs and Social Media (2012)
7. Ho, S.-S., Lieberman, M., Wang, P., Samet, H.: Mining future spatiotemporal events and their sentiment from online news articles for location-aware recommendation system. In: Proceedings of the 1st ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (MobiGIS), pp. 25–32 (2012)
8. Koren, Y.: Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM International Conference on Knowledge Discovery & Data Mining (KDD), pp. 447–456 (2009)
9. Lu, Z., Savas, B., Tang, W., Dhillon, I.S.: Supervised link prediction using multiple sources. In: Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), pp. 923–928 (2010)
10. Marinho, L.B., Nunes, I., Sandholm, T., Nóbrega, C., Araújo, J.a., Pires, C.E.S.: Improving location recommendations with temporal pattern extraction. In: Proceedings of the 18th Brazilian Symposium on Multimedia & the Web (WebMedia), pp. 293–296 (2012)
11. Noulas, A., Scellato, S., Lathia, N., Mascolo, C.: A random walk around the city New venue recommendation in location-based social networks. In: Proceedings of the International Conference on Privacy, Security, Risk & Trust (PASSAT), and International Conference on Social Computing (SocialCom), pp. 144–153 (2012)
12. Pan, J., Yang, H., Faloutsos, C., Duygulu, P.: Automatic multimedia cross-modal correlation discovery. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pp. 653–658 (2004)

13. Raymond, R., Sugiura, T., Tsubouchi, K.: Location recommendation based on location history and spatio-temporal correlations for an on-demand bus system. In: Proceedings of the 19th ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL), pp. 377–380 (2011)
14. Tong, H., Faloutsos, C., Pan, J.: Fast random walk with restart and its applications. In: Proceedings of the 6th International Conference on Data Mining (ICDM), pp. 613–622 (2006)
15. Vasuki, V., Natarajan, N., Lu, Z., Savas, B., Dhillon, I.: Scalable affiliation recommendation using auxiliary networks. ACM Trans. Intell. Syst. Technol. **3**(1), 3:1–3:20 (2011)
16. Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., Sun, J.: Temporal recommendation on graphs via long- and short-term preference fusion. In: Proceedings of the 16th ACM International Conference on Knowledge Discovery &Data Mining (KDD), pp. 723–732 (2010)
17. Yin, Z., Gupta, M., Weninger, T., Han, J.: A unified framework for link recommendation using random walks. In: Proceedings of the IEEE International Conference on Advances in Social Networks Analysis & Mining (ASONAM), pp. 152–159 (2010)
18. Yuan, Q., Cong, G., Ma, Z., Sun, A., Thalmann, N.M.: Time-aware point-of-interest recommendation, In: Proceedings of the 36th ACM International Conference on Research & Development in Information Retrieval (SIGIR), pp. 363–372 (2013)
19. Yuan, Q., Cong, G., Sun, A.: Graph-based point-of-interest recommendation with geographical and temporal influences. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM), pp. 659–668 (2014)