Pers Ubiquit Comput
DOI 10.1007/s00779-011-0419-8

ORIGINAL ARTICLE

Social itinerary recommendation from user-generated digital trails

Hyoseok Yoon · Yu Zheng · Xing Xie · Woontack Woo

Received: 15 November 2010 / Accepted: 20 April 2011

Abstract Planning travel to unfamiliar regions is a difficult task for novice travelers. The burden can be eased if the resident of the area offers to help. In this paper, we propose a social itinerary recommendation by learning from multiple user-generated digital trails, such as GPS trajectories of residents and travel experts. In order to recommend satisfying itinerary to users, we present an itinerary model in terms of attributes extracted from user-generated GPS trajectories. On top of this itinerary model, we present a social itinerary recommendation framework to find and rank itinerary candidates. We evaluated the efficiency of our recommendation method against baseline algorithms with a large set of user-generated GPS trajectories collected from Beijing, China. First, systematically generated user queries are used to compare the recommendation performance in the algorithmic level. Second, a user study involving current residents of Beijing is conducted to compare user perception and satisfaction on the recommended itinerary. Third, we compare mobile-only approach with Mobile+Cloud architecture for practical mobile recommender deployment. Lastly, we discuss personalization and adaptation factors in social itinerary recommendation throughout the paper.

H. Yoon · W. Woo (✉)
Gwangju Institute of Science and Technology,
Gwangju 500-712, South Korea
e-mail: wwoo@gist.ac.kr

H. Yoon
e-mail: hyoon@gist.ac.kr

Y. Zheng · X. Xie
Microsoft Research Asia, Beijing 100190, China
e-mail: yuzheng@microsoft.com

X. Xie
e-mail: xingx@microsoft.com

1 Introduction

Despite the increased number of travelers, international and inexperienced travelers still face many difficulties in planning their trip. A dilemma travelers face is related to the efficient use of the limited time. Since there are many possible locations to visit, travelers want to maximize the travel experience, i.e., visit many interesting locations without wandering around. In this regard, many recommendation techniques are researched and being developed in the tourism industry [5, 8, 17, 23].

Especially, inexperienced travelers can take a social approach to ask people who know about the area to be explored. Travelers can ask travel experts who have already traveled through the area or local residents for recommendation. The advantage is that the recommendation is up-to-date with accurate and timely information. However, the quality of recommendation varies depending on different people, and it takes time for users to digest and put collected information together for use.

In our approach, we want to enhance itinerary recommendation by incorporating knowledge of socially relevant experts such as travel experts and active residents of the region. To extract meaningful knowledge, we data mine user-generated digital trails, such as GPS trajectories for finding interest locations, inter-related locations in sequence, and time to stay and travel. Such data-mined knowledge enables many interesting application scenarios. This work is an extension to [25], and we made further contributions in personalization factors of social itinerary recommendation

Published online: 16 June 2011

and adaptation to practical mobile development and deployment.

Consider a researcher is attending a conference in Beijing, China. At the end of the conference, he has 8 h to spend before catching up his flight. Being a newcomer to this area, he uses Social Itinerary Recommendation Service. He starts from his current location, which is automatically recognized with his GPS-enabled phone. He marks the Beijing Capital International Airport in the map as the destination, inputs 8 h for travel duration, and sends the query. Then, he receives an itinerary recommendation visualized on the map that shows interesting locations to visit, how long to stay in each location, and estimation of traveling time. With these information at hand, he gets a good picture of where he will go and able to manage his time in advance.

As explained in this application scenario, we can recommend new travelers an itinerary that makes the efficient use of the given duration by considering multiple users' accumulated travel routes and experiences. If user-generated GPS trajectories from travel experts and active residents in the region are accumulated as good examples and processed properly, we can extract many features to produce collective social knowledge and aid new users in building an efficient travel itinerary. Figure 1 illustrates the concept of our work.

Our contribution in this paper is as follows.

1. **Social attributes**. We data mine and extract social attributes from multiple user-generated GPS trajectories to incorporate social collective knowledge which is used in the recommendation framework.
2. **Itinerary modeling**. We model and define what a good itinerary is in terms of attributes detected in the digital trails and present a quality evaluation approach of an itinerary.
3. **Recommendation framework**. We present a social itinerary recommendation framework consisted of offline data mining and online recommendation with a simplified user query.
4. **Evaluation**. We evaluate our method using a large GPS dataset collected from 125 users and compare against baseline methods both in simulation level and through user study. Also, the performance of Mobile+Cloud architecture is measured for practical mobile application adaptation and deployment.

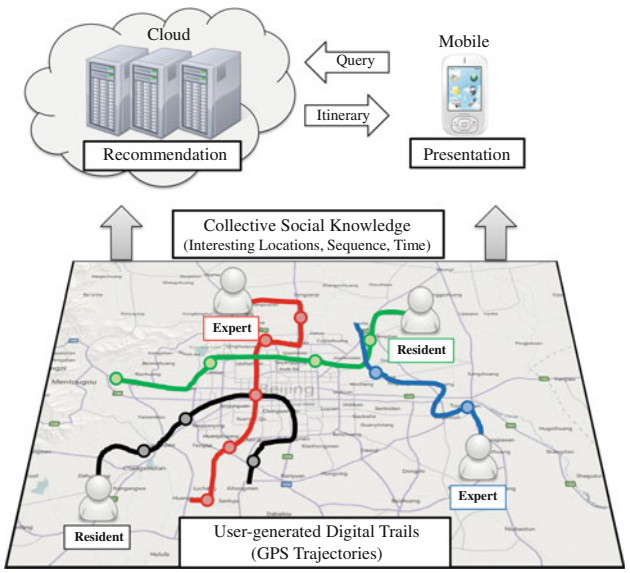


Fig. 1 Social itinerary recommendation service

The rest of the paper is organized as follows. Section 2 reviews related works on itinerary recommendation and GPS data mining. Section 3 describes the proposed itinerary modeling. Section 4 presents detail description of social itinerary recommendation processes. In Sect. 5, we present experiment results and provide discussions followed by conclusion in Sect. 6.

2 Related work

2.1 Itinerary recommendation

In itinerary recommendation, there are two branches of related work. One branch deals with high user intervention for an interactive recommendation system, and the other branch aims for less user intervention for an automated recommendation system.

Dunstall et al. [5] presented an automated but more of an interactive travel itinerary planning system where a user defines which places to visit and avoid. Similarly, Ardissono et al. [1] developed an interactive system where a user specifies general constraints such as time and attraction items to be included in the itinerary. Kim et al. [12] also presented an interactive system that starts by a user selecting the first location to get recommendation on similar types of places. The advantage of such interactive recommendation systems is that more the user knows about the traveling area, and the more accurate and detailed itinerary is prepared by the user. However, this assumption is not practical for novice travelers without any prior knowledge.

In more automated approach, Huang and Bian [11] build a travel recommendation system based on heterogeneous online travel information such as tourism ontology and estimated travel preferences by the Bayesian network. Kumar et al. [14] presented GIS-based Advanced Traveler Information System (ATIS) for Hyderabad City in India that includes a site-tour module based on the shortest distance calculation. Chodhury et al. [4] used Flickr photo stream as digital trails where location and time information

are extracted from individual photo streams and turned into a Place of Interest (POI) to generate itineraries.

Compared to the related works, our approach is more of an automated approach, which requires a simplified query composed of two points and duration to recommend an automatically generated itinerary. Unlike other approaches, we also focus on the realistic and social knowledge preparation required for itinerary recommendation through the use of user-generated GPS trajectories.

2.2 GPS data mining applications

GPS data have been used to link geographical location with time stamp and people involved. Through data mining or post-processing multiple GPS data, many applications extract meaningful information for various uses. Simply, GPS trajectory can be analyzed to find patterns within [7, 16] and predict the repeating pattern [13]. Through post-processing, the raw GPS can be turned into a more usable form, such as routable road map [3]. Regarding travel and tourism applications, GPS data have been used to find locations of interest [2], integrated into a mobile tour guide [19, 20, 22], and combined with other resources such as geo-tagged photographs [21]. In GeoLife [29, 32, 36], many GPS-related applications and scenarios are introduced. Zheng and Xie used GPS traces for travel recommendations [37] to recommend both generalized and personalized interesting locations [33]. GPS data are also used to classify different categories of transportation modes, such as driving, walking, bus, and bike [30, 31, 35]. Different approaches use location history to recommend geographic locations such as shops or restaurants by mining correlated locations [34] and also reflect user similarity considering the travel sequence and hierarchical structure of geographical spaces [15]. The user similarity is also used in [38] as a user-centric collaborative filtering model for recommending friends and locations. Furthermore, Zheng et al. [28] recommended similar users and recommended activity-related locations or location-related activities with user-generated comments [27].

Our work in this paper extends location-level recommendation to an itinerary-level recommendation and proposes an efficient itinerary recommendation algorithm considering a multiple number of social attributes. We also evaluate and validate our method with a large set of real user-generated GPS trajectories to confirm the efficiency found in algorithmic level to the real use cases and adapt to mobile settings.

3 Proposed itinerary model

An itinerary outlines which locations to visit and leave when and in what order. Moreover, it shows an estimated traveling time from one location to a subsequent location. Therefore, well-constructed itineraries give users a good indication of what to expect next and where they stand in their trip.

When building an itinerary, the duration is the most important constraint. The goal of providing a usable itinerary is to provide a sequence of visiting locations accurately with traveling time and staying time under the given duration. It is easy to make an itinerary that maximizes the number of visiting locations, yet the task becomes difficult when time constraint is introduced. On the other hand, time constraint is applied as a stopping condition for simplifying recommendation complexity. Additionally, we consider the following four factors collectively to determine quality of an itinerary.

1. **Elapsed time ratio (ETR)**: Simply, available time should be used fully. If the total time needed for an itinerary is much shorter than available time, then the remaining time is wasted unless used for a part of travel. ETR measures the overall use of the available time. Generally, this factor is also related to the number of locations visited, since visiting more places requires more time. It is unlikely that an itinerary with shorter duration yields more locations than an itinerary with longer duration. Therefore, we aim to maximize the elapsed time in an itinerary as close as the maximum duration specified in a query.
2. **Stay time ratio (STR)**: We also consider how the available time is spent. We model travel itinerary in a way that travelers spend more time on the locations compared to the traveling time. An itinerary with more staying time is considered to be a better choice. STR depicts the balance between visiting time on-site and transferring time. Higher STR means that a user is spending more time on visiting actual places than spending time on transferring between locations. This is also a desirable and typical factor that we assume to be true. For example, given 10-h duration, we treat an itinerary with 8 h of stay time and 2 h of traveling time better than another itinerary with 2 h of stay time and 8 h of traveling time.
3. **Interest density ratio (IDR)**: In an itinerary, it is important what types of locations are included. General assumption is that new visitors like to visit as many highly interesting locations as possible, i.e., popular locations and locations with cultural importance. If an itinerary is composed of many locations of high interests (greater interest density), then it is considered to be a better itinerary than another itinerary with locations of less interest. IDR shows the overall degree of popularity for the included locations. For the simplicity of illustration, if ETR and

STR are the same for two different itinerary, then higher IDR is preferred, since the only difference is that the visited locations differ in popularity or interest level.

4. **Classical travel sequence ratio (CTSR)**: In our itinerary model, we incorporate social aspects as well. We value travel sequences frequently used by people more important than other random sequences. An itinerary that contains classical travel sequence of previous users is better, more realistic and practical as well. Since we socially recommend itineraries based on previous users' experiences, we generate itineraries that revisit good patterns of previous users. For example, if two itineraries have similar ETR, STR, and IDR, what we consider further is how practical each itinerary is. If one itinerary revisits and includes patterns found from other users, namely classical travel sequence or visiting pattern between locations, we choose this itinerary over another itinerary that does not have this pattern.

We use the first three characteristics to find potential quality trips that surpass some thresholds shown as a cube in Fig. 2. In theory, the best itinerary would have values equal to 1 in all three dimensions, which is depicted as a black dot in Fig. 2. The trips falling into the cube are considered a high-quality itinerary since it performs well in all three factors. The selected trips in the cube are re-ranked according to classical travel sequence to differentiate candidates further.

For a generic recommender, all four factors are considered equal by assigning the same weight value, because we have not found any evidence or support of the dominating attribute yet. However, there is different personal preference on these four factors, so as a personalization factor, we allow the weight to be changed if a user's preference is known or the user chooses to modify it in a personalized recommender. The assignment of different weight for these four attributes is empirically decided or varied with applications by design choice.

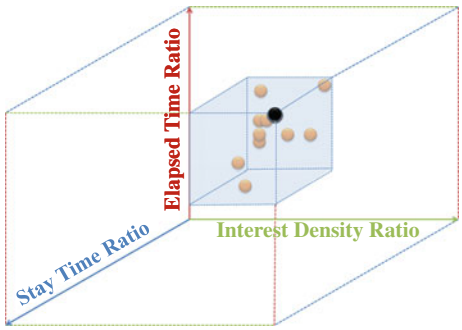


Fig. 2 Trip candidates for a good itinerary

4 Social itinerary recommendation

4.1 Architecture

For the itinerary recommendation, we configure our architecture into offline tasks for processing time-consuming and static information and online tasks for processing variable user queries as depicted in Fig. 3.

In offline processing, the user-generated GPS trajectories are analyzed to build a *Location-Interest Graph* (G) with location and interest information, this is quite time-consuming process that needs to be done initially. Then, G is to be built again only after a significant amount of user-generated GPS trajectories are updated. In online processing, we use the G built in offline to recommend an itinerary based on a user-specified query.

Our recommendation method is consisted of the following six modular tasks. First two operations are carried out in offline, and the latter four operations are performed online. Here, we briefly describe each task, and details are presented in following sections.

1. Stay points extraction and clustering (4.1.1, 4.2.1)
2. Location interest and sequence mining (4.1.2, 4.2.3)
3. Query verification (4.1.3)
4. Trip candidate selection (4.1.4)
5. Trip candidate ranking (4.1.5)
6. Re-ranking by travel sequence (4.1.6)

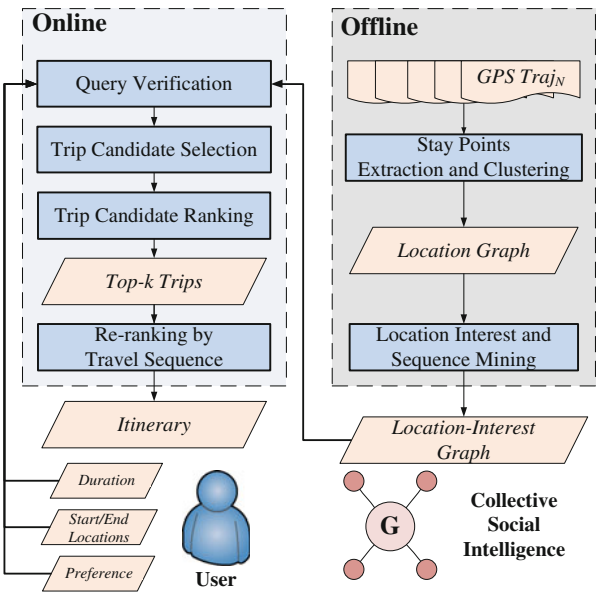


Fig. 3 Architecture of social itinerary recommender

4.1.1 Stay points extraction and clustering

From multiple user-generated GPS trajectories, we extract stay points using certain distance and time thresholds. Then, these stay points are clustered into locations which become candidate locations to be included in an itinerary, and connected locations are checked to form an edge. This operation ensures that only locations with significant activity (many people visiting and staying at certain location) and relevance (connected locations) are chosen. For each location cluster of stay points, we calculate arrival time, leaving time, and typical stay time which are used to estimate the duration of an itinerary. We find the median for staying time by subtracting arrival time from leaving time of each location to represent how long a typical visitor spends in that location.

4.1.2 Location interest and sequence mining

Locations can be characterized by its popularity which we call location interest, and some locations are typically visited in a certain sequence, i.e., one location after another. These characteristics are inferred in this operation, and details are presented in [33]. These inferred information provides check points and measurements toward how realistic and practical recommended itineraries are. After calculating these properties for all locations, we build a G. G is composed of locations as vertexes and traveling time between two connected locations as edges, also each location is assigned an interest and classical sequence value.

4.1.3 Query verification

When a user sends a query to receive an itinerary recommendation, we first check and verify the query. For extreme cases, a user might have queried with such a short duration that it is impossible to even go straight from the start location to the end location. These impractical queries can be filtered out by checking the distance between start and end location with respect to the duration. Also, start and end points may not fall into one of locations in G. In this case, we adjust the user query by finding the nearest location and updating the query.

4.1.4 Trip candidate selection

Using G, traveling time between locations and a user query are used to generate and select n candidate trips. As a personalization factor for a revisiting user, G can be further refined by excluding locations of previous visit to generate itinerary composed of new locations. Then, the only constraint we check for this step is that the generated trips adhere to the given duration constraint and to start and end as specified in the user query. Among these n trips, some trips are not worth considering, which are eliminated in subsequent steps.

4.1.5 Trip candidate ranking

From the generated and selected n trips, we rank trips according to elapsed time ratio, stay time ratio, and interest density ratio. We consider trips with higher ratio of elapsed time a better itinerary, prefer trips with higher stay time ratio meaning that the user stays longer at locations rather than traveling, and look for trips with higher interest density for visiting many locations of greater interest. We rank each trip according to the Euclidean distance of trip in three dimensions of elapsed time ratio, stay time ratio, and interest density ratio. The *top-k* trips ranked out of n candidates by the Euclidean distance are selected as further itinerary candidates.

4.1.6 Re-ranking by travel sequence

Among *top-k* candidates, we score and rank each trip again considering classical travel sequence. This strengthens the resulting itinerary to be practical and realistic to revisit many previous users' sequence of choice.

4.2 Offline data mining

In this section, we describe offline itinerary recommendation processes. We describe how G is generated and describe the involved data mining procedures. From multiple users' GPS trajectories, we detect stay points (Definition 3) and cluster them into locations (Definition 5). Further, location interest is calculated (Definition 7), and classical travel sequence is mined by considering hub scores, authority scores, and probability of taking this specific sequence (Definition 8). Details of mining interesting locations and classical travel sequences are presented in [33]. With this information, we build G offline (Definition 9). The definitions of terminologies are adopted from [25].

4.2.1 Stay point detection

Definition 1 *Trajectory*. A user's trajectory *Traj* is a sequence of time-stamped points, $Traj = \langle p_1, p_2, \ldots, p_k \rangle$. Each point is represented by $p_i = (lat_i, lng_i, t_i), (i = 1, 2, \ldots, k); t_i$ is a time stamp, $\forall 1 \leq i < k, t_i < t_{i+1}$ and (lat_i, lng_i) are GPS coordinates of points.

Definition 2 *Distance and Interval*. Dist(p_i, p_j) is the geospatial distance between two points p_i and p_j, and the time interval between two points is denoted $\text{Int}(p_i, p_j) = |t_i - t_j|$.

Definition 3 *Stay Point*. A stay point s is a geographical region where a user stayed over a time threshold T_r within a distance threshold of D_r. In a user's trajectory, s is characterized by a set of consecutive points $P = \langle p_m, p_{m+1}, \ldots, p_n \rangle$, where $\forall m < i \leq n$, Dist$(p_m, p_i) \leq D_r$, Dist$(p_m, p_{n+1}) > D_r$ and Int$(p_m, p_n) \geq T_r$. Then, $s = (slat, slng, at, lt, st)$ where

$$slat = \frac{\sum_{i=m}^{n} lat_i}{|P|}, \quad slng = \frac{\sum_{i=m}^{n} lng_i}{|P|} \tag{1}$$

respectively stands for the average *lat* and *lng* coordinates of the collection P; $at = t_m$ is the user's arriving time on s, and $lt = t_n$ represents the user's leaving time.

4.2.2 Location clustering

Definition 4 *Location History*. An individual's location history h is represented as a sequence of stay points they visited with corresponding arrival: *at*, leaving times: *lt* and time interval from s_i to $s_j : \Delta t_{i,j} = at_j - lt_i$ where $\forall 1 < i < j \leq n$

$$h = \langle s_1 \overset{\Delta t_{1,2}}{\rightarrow} s_2 \overset{\Delta t_{2,3}}{\rightarrow} s_3, \ldots, s_{n-1} \overset{\Delta t_{n-1,n}}{\longrightarrow} s_n \rangle \tag{2}$$

We put together the stay points detected from all users' trajectories into a dataset $\boldsymbol{S}$ and employ a clustering algorithm to partition this dataset into some clusters. Thus, the similar stay points from various users will be assigned into the same cluster.

Definition 5 *Locations*. $L = \{l_1, l_2, \ldots, l_n\}$ is a collection of locations. Between any two locations, there is no overlapping Stay points ($s \in \boldsymbol{S}$) detected from multiple users' trajectories: $i \neq j$, $l_i \cap l_j = \varnothing$.

After the clustering operation, we can substitute a stay point in a user's location history with the cluster ID the stay point pertains to. Supposing $s_1 \in l_i, s_2 \in l_j, s_3 \in l_k, s_{n-1} \in l_l, s_n \in l_m$, (2) can be replaced with

$$h = \langle l_i \overset{\Delta t_{i,j}}{\rightarrow} l_j \overset{\Delta t_{j,k}}{\rightarrow} l_k, \ldots, l_l \overset{\Delta t_{l,m}}{\longrightarrow} l_m \rangle \tag{3}$$

Thus, different users' location histories become comparable and can be integrated to recommend a single location.

Definition 6 *Typical Stay Time and Time Interval*. For each location $l_i \in L$ with m stay points that pertain to this location, typical stay time of location (l_i), st_i is defined as median of stay time $(st_k = lt_k - at_k)$ of stay point (s_k) where $\forall s_k \in \boldsymbol{l_i}, \forall 1 \leq k \leq m$.

$$st_i = \text{Median}(st_k) \tag{4}$$

For n location histories $(h^1, \ldots, h^n)$ with a sequence $l_i \overset{\Delta t_{i,j}}{\rightarrow} l_j$ where $l_i, l_j \in L$ and $l_i \neq l_j$, typical time interval $\Delta T_{i,j}$ from l_i to l_j is defined as in (5), and all typical time intervals are put into a dataset ΔT where $\forall 1 \leq k \leq n$.

$$\Delta T_{i,j} = \text{Median}(h^k_{\Delta t_{i,j}}) \tag{5}$$

4.2.3 Location interest

Definition 7 *Location Interest*. We utilize HITS (hypertext-induced topic search) idea that a good hub points to many good authorities, and a good authority is pointed to by many good hubs to represent location interest. In HITS-based inference model, a hub is a user who has accessed many places, and an authority is a location which has been visited by many users [33]. I_j represents location interest at l_j which has a mutual reinforcement relationship with user travel experience. Figure 4 depicts this relationship using HITS-based inference model.

For example, a user with greater travel expertise would visit many interesting locations, and interesting locations are visited by many users with much travel experiences. More specifically, a user's travel experience can be represented by the sum of the interests of the locations they accessed; in turn, the interest of a location can be calculated by integrating the experiences of the users visiting it [33]. The mutual relationship of location interest I_j and travel experience e_i is represented as (6) and (7). An item r_{ij} stands for the times that user u_i has stayed in location l_j.

$$I_j = \sum_{u_i \in U} r_{ji} \times e_i \tag{6}$$

$$e_i = \sum_{l_j \in L} r_{ij} \times I_j \tag{7}$$

4.2.4 Classical travel sequence

Definition 8 *Classical Travel Sequences*. The classical travel sequence integrates three aspects, the authority score of going in and out and the hub scores, to score realistic and practical travel sequences [33].

Figure 5 demonstrates the calculation of the classical score for a 2-length sequence $l_1 \rightarrow l_3$. The connected edges

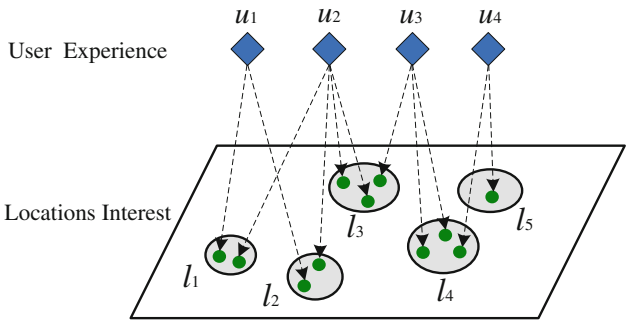


Fig. 4 User experience and location interest relationship

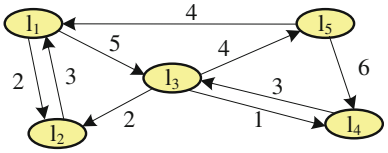


Fig. 5 Demonstration of classical travel sequence

represent people's transition sequence, and the values on the edges show the times users have taken the sequence. Equation (8) shows the calculation based on the following parts. 1) The authority score of location l_1 (a_{l1}) weighted by the probability of people moving out from this sequence ($\text{Out}_{l1,l3}$). In this demonstration, $\text{Out}_{l1,l3} = 5/7$. 2) The authority score of location l_3 (a_{l3}) weighted by the probability of people's moving in by this sequence ($\text{In}_{l1,l3}$). 3) The hub scores h_b of the users ($U_{l1,l3}$) who have taken this sequence. Classical travel sequence scores are stored in a k-by-k adjacent matrix M between locations.

$$c_{l_1,l_3} = \sum_{u_k \in U_{l_1,l_2}} (a_{l_1} \times \text{Out}_{l_1,l_2} + a_{l_3} \times \text{In}_{l_1,l_3} + {h_b}^k) \tag{8}$$

4.2.5 Location-Interest Graph

Definition 9 *Location-Interest Graph* (G). Formally, a G is a graph $G = (V, E)$. Vertex set V is locations (Definition 5) $L, V = L = \{l_1, l_2, \ldots, l_k\}$. Edge set E is replaced by ΔT where $\Delta T_{i,j}$ stands for a travel sequence from l_i to l_j where $1 \leq i < j \leq k$ with typical time interval as its value. So if there exists an edge between l_i and l_j, then there is a non-zero travel time in corresponding $\Delta T_{i,j}$.

In summary, G contains information on (1) Location itself (interest, typical staying time) and (2) relationship between locations (typical traveling time, classical travel sequence).

4.3 Online recommendation

In this section, we describe online itinerary recommendation processes. We describe how G is utilized and describe the involved recommendation procedures. For a user-supplied query, trip candidates are first selected considering the user query constraints. Then, trip candidates are ranked according to three attributes defined in our itinerary modeling and further re-ranked with classical travel sequence.

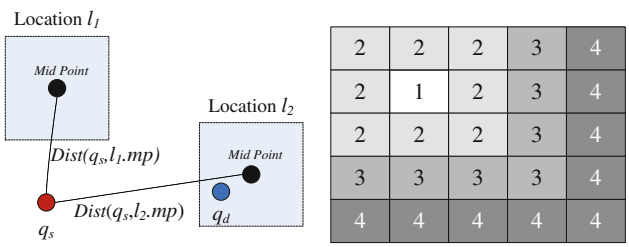
4.3.1 Query verification

Definition 10 *User Query*. A user-specified input with three tuple attributes (start point, end point, and duration) is defined as a user query, $Q = \{q_s, q_d, q_t\}$.

We first verify user query Q in the online process by calculating the distance between the start point and end point. Note that the query uses points as unit for specifying start and end point which is different from unit of location which is a cluster of stay points. There are two approaches we can estimate the distance, $\text{Dist}(q_s, q_d)$. First, we can use the Haversine formula [9] or the spherical law of cosines [18] with the raw GPS coordinates of start point and end point. Once we have an estimated distance, we estimate the traveling time by dividing the estimated distance by an average traveling speed of car in the region, i.e., 30 km/h.

Alternatively, we can use Web service such as Microsoft Bing Map to find traveling distance between two specified locations and traveling time. Since we only have traveling time between the two locations, we multiply the estimated time by a factor of w, which is empirically determined for different applications.

After confirming the duration, we attempt to locate the start point and the end point in G. If we can successfully locate the specified point q_d as in Fig. 6a, then the point is adjusted to l_2. However, if the specified point in the query does not belong to any locations in G as in q_s in Fig. 6a, then we find the nearest location l_1 among others by checking the distance to the location's midpoint. To speed up the process of finding nearest location, we employ a grid-based indexing and searching. For grid-based indexing, we put all locations according to its latitude and longitude ranges into n grids, in Fig. 6b, $n = 25$. Then, we find the grid cell that contains the specified point. We start from that grid cell, and if the grid cell does not contain any location, then we increase one level to increase the searching window. The Fig. 6b depicts searching windows for different levels when the specified point belongs to the 7th grid. To support the round-trip cases where the start point and end point are the same, we find nearest locations that do not overlap between the start and end points. Then, we connect the start point to the start location and connect the end point to the end location. As the final step, we find the traveling time from the specified location to the nearest location found using Bing Map and subtract the traveling time to update *Dur*. The original query $Q = \{q_s, q_d, q_t\}$ becomes $Q' = l_s, l_d, q'_t$, then the query is sent.



(a) Finding nearest location **(b)** Grid index for search

Fig. 6 Adjusting locations in user query

4.3.2 Trip candidate selection

With the verified user query, we select trip candidates from the starting location l_s to the end location l_d.

Definition 11 *Trip*. A trip Trip is a sequence of locations with corresponding typical time intervals,

$$\text{Trip} = \langle l_1 \overset{\Delta T_{1,2}}{\rightarrow} l_2 \overset{\Delta T_{2,3}}{\rightarrow} l_3, \ldots, \overset{\Delta T_{k-1,k}}{\longrightarrow} l_k \rangle \tag{9}$$

where $\forall 1 \leq i < j \leq k, \Delta T_{i,j} \in \Delta T$ and $l_i, l_j \in L$ are locations. Trip has four attributes: (1) the total staying time for visiting locations t_{stay}, (2) the total traveling time t_{trav}, (3) the duration of the trip t_{dur}, and (4) the interest density of the trip i_{den} defined by the total sum of interest of locations divided by the number of locations.

$$t_{\text{stay}} = \sum_{i=1}^{k} st_i \tag{10}$$

$$t_{\text{trav}} = \sum_{i=1}^{k-1} \Delta T_{i,i+1} \tag{11}$$

$$t_{\text{dur}} = t_{stay} + t_{trav} \tag{12}$$

$$i_{\text{den}} = (\sum_{i=1}^{k} I_i)/k \tag{13}$$

The only restriction we impose in this stage is time constraint so that the candidate trips do not exceed the given duration q_t. The candidate selection process is shown in Algorithm 1.

We first start from a path that includes the start location l_s as the sole location. Then, we check other locations not in this path but are feasible to visit with the remaining duration recursively. The constraint of duration and visited location information are used as heuristics to select the next location (refer to Lines 2–3). As we add a new location for the path, we also add them to L_v, so that this location is not checked in the next recursive call (refer to Line 5). For a personalization factor for revisiting users, L_v can be updated with previous visiting history to exclude previously visited locations, so it is not included again in the generated itinerary. For each location added to the path, we subtract the stay time of the location and traveling time to the location to yield a new remaining time (refer to Line 6). Once the path reaches at the end location, we add the generated path as a candidate trip (refer to Line 10). When all the candidates are added, it returns an array of n trip candidates Tr as results.

4.3.3 Trip candidate ranking

After selecting n trip candidates from previous step, we rank each trip with factors from Section 3 The factors used to rank each trip $tr_i \in Tr, 1 \leq i \leq k$ are as follows:

Algorithm 1 CandidateSelection(G, l_s, l_d, q_t, L_v)

Input: A Location-Interest Graph G, a start location l_s, a destination
l_d, duration q_t, and a visited location set L_v
Output: A set of candidate trips Tr

```
1: for all i such that 1 ≤ i ≤ k do
2:   if (!L_v.Contains(l_i)) then
3:     if (q_t ≥ ΔT_{s,i} > 0) then
4:       L_n.Add(L_v)
5:       L_n.Add(l_i)
6:       Dur_n ⇐ q_t − st_i − ΔT_{s,i}
7:       if L_n[0] = = l_s then
8:         Dur_n ⇐ Dur_n − st_s
9:       if l_i = = end and Dur_n ≥ 0 then
10:        Tr.Add(L_n)
11:      else if Dur_n > 0 then
12:        CandidateSelection(G, l_i, l_d, Dur_n, L_n)
13: return Tr
```

1. Elapsed time ratio (ETR) $= t_{\text{dur}} / q_t$
2. Stay time ratio (STR) $= t_{\text{stay}} / q_t$
3. Interest density ratio (IDR) $= i_{\text{den}} / i_{\max}$

Here, we can use some threshold values to quickly reject undesirable candidates, i.e., reject candidates with elapsed time ratio less than 0.5. Then, we find the Euclidean distance of each trip using these 3 dimensions as in (14). Here, $i_{\max}$ refers to a maximum interest density value in all of candidate trips which we use for normalization. We can assign different weight values for the factors by setting α_1, α_2, and α_3. For our system, we treat three factors equally important by setting $\alpha_1 = \alpha_2 = \alpha_3 = 1$ for a generic recommender, but with the user preference value, this setting can be changed for personalization.

$$\text{ED} = \sqrt{\alpha_1(\text{ETR})^2 + \alpha_2(\text{STR})^2 + \alpha_3(\text{IDR})^2} \tag{14}$$

As described in Algorithm 2, for each trip, three factors are calculated to yield the Euclidean distance value. The algorithm returns an array of *top-k* trips in decreasing order of the Euclidean distance value.

4.3.4 Re-ranking by travel sequence

We have cut down the number of candidate trips from n to k. These k trips will likely have similar Euclidean distance values. So how can we differentiate between candidates and recommend one over another? Our solution is to examine each trip's travel sequence and score them for any classical travel sequences (Definition 8). When two trips have similar values in Euclidean distance after the first ranking, however they will have different classical travel

Algorithm 2 CandidateRanking(G, Tr, q_t)

Input: A Location-Interest Graph G, a set of trips Tr, and the duration q_t

Output: A set of top-k trips Tr', sorted by Euclidean distance

1: **for all** Trip $tr \in Tr$ **do**
2: **for all** Location $loc \in tr$ **do**
3: $t_{trav} \Leftarrow t_{trav} + \Delta T_{prevLoc,loc}$
4: $t_{stay} \Leftarrow t_{stay} + st_{loc}$
5: $i_{den} \Leftarrow i_{den} + I_{loc}$
6: $prevLoc \Leftarrow loc$
7: $t_{dur} \Leftarrow t_{trav} + t_{stay}$
8: **if** $i_{den} > i_{max}$ **then**
9: $i_{max} \Leftarrow i_{den}$
10: $tr.SetEucDist(t_{dur}/q_t, t_{stay}/q_t, i_{den}/i_{max})$
11: $Tr' \Leftarrow SortByED(Tr)$
12: **return:** Tr'

sequence score. We give preference toward trips with higher classical travel sequence score, which means that we recommend trips to revisit previous visitors' practical travel sequences. Using the classical travel sequence accumulation using classical travel sequence matrix M, we can score any travel sequence,

$$c(l_1 \rightarrow l_2 \rightarrow l_3) = c_{1,2} + c_{2,3} \tag{15}$$

Once we have classical travel sequence score of tr_i by calculating $c(tr_i)$, we normalize it by the maximum classical travel sequence score $MaxC$ found of all candidates.

Classical travel score ratio (CTSR) $= c(tr_i)/MaxC$. Then, we once again use the Euclidean distance, this time including classical travel sequence score to re-rank k candidates. We use equal weights for all four factors as shown in (16), but with the user preference value or user interaction, this settings can be changed for personalization. The first itinerary with the highest Euclidean distance value is recommended to user, and the user can view alternative itineraries in the order of the Euclidean distance.

$$\mathrm{ED}' = \sqrt{\alpha_1(\mathrm{ETR})^2 + \alpha_2(\mathrm{STR})^2 + \alpha_3(\mathrm{IDR})^2 + \alpha_4(\mathrm{CTSR})^2} \tag{16}$$

Definition 12 *Itinerary*. An itinerary It is a trip recommendation based on user's start point q_s and destination q_d constrained by trip duration threshold q_t in a query.

$$It = \langle q_s \in l_s \overset{\Delta T_{s,1}}{\rightarrow} l_1 \overset{\Delta T_{1,2}}{\rightarrow} l_2, \ldots, l_{k-1} \overset{\Delta T_{k-1,k}}{\rightarrow} l_k \overset{\Delta T_{k,d}}{\longrightarrow} q_d \in l_d \rangle \tag{17}$$

This means that the resulting itinerary starts from q_s and end in q_d where the duration of trip t_{dur} does not exceed available q_t, $t_{\mathrm{dur}} \leq q_t$ while maximizing the Euclidean distance of four attributes.

Fig. 7 User interface of itinerary recommender

4.4 User interface

Figure 7 shows the user interface of social itinerary recommendation system. Our user interface has three components. The upper-left input panel is for specifying a start, an end location and duration for querying. User can mark locations by clicking on the map or by searching with keywords. User can also set the start and end location same for the round-trip. The lower-left is an output panel for showing step-by-step instructions in text, and on the right, an itinerary is visualized on the map.

5 Experiments

In this section, we explain the experiment settings and evaluation approaches and present the experiment results.

5.1 Settings

To collect user-generated GPS trajectories, we have used stand-alone GPS receivers as well as GPS phones. Using these devices, 125 users recorded 17,745 GPS trajectories in Beijing from May 2007 to Aug 2009. One hundred and twenty-five volunteers are recruited from Microsoft employees, employees of other companies, government staff, and college students. These volunteers are motivated by payment-based incentives to log their outdoor movements as much as possible since more GPS trajectory collected by them would yield more money. In this experiment, time threshold T_r and distance threshold D_r are set to 20 min and 200 m, respectively. With these parameters, we detected 35,319 stay points from the dataset and excluded work/home spots. For clustering these stay points into unique locations, we used a density-based clustering algorithm OPTICS (Ordering Points To Identify the Clustering Structure) which resulted in 119 locations as depicted in Fig. 8a. For the grid-based indexing and nearest location search, we divided Beijing area into 25 grid cells

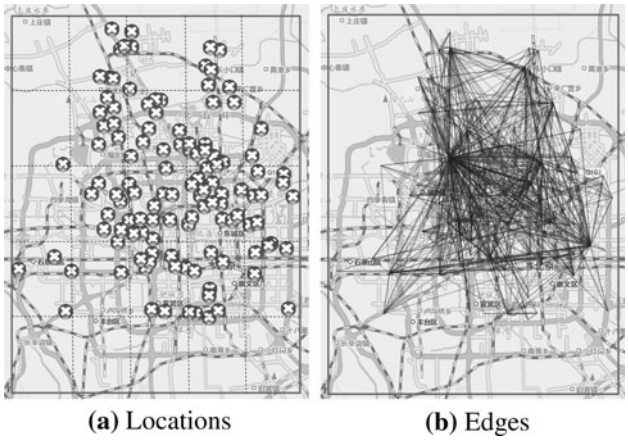


Fig. 8 Locations and edges of G_r

as shown in Fig. 8a. Note that there are some grids with no locations or very few locations. Among these 119 locations, typical traveling time is assigned for the connected locations, which serves as an edge set for G. Figure 8b shows the visualization of edges, representing travel connections in our dataset. The dataset has been made public for research use [6, 30, 33].

5.2 Evaluation approaches

In the experiment, we use three evaluation approaches to evaluate our itinerary recommendation methods. First approach is based on a large amount of simulated user queries for the algorithmic level comparison. Using this synthetic dataset, we evaluate the quality of the generated itineraries quantitatively compared to other baseline methods. Second approach is based on a user study where real users evaluate the generated itineraries by our method and baseline methods. In second approach, we observe how user's perceived quality of itineraries compares by different methods. Lastly, we evaluate our recommendation methods on Mobile+Cloud architecture for practical mobile adaptation and application deployment.

5.2.1 Simulation

We simulated a large quantity of user queries to evaluate the effectiveness of our method. For our simulation to cover most general cases of user input, we used four different levels for duration, 5, 10, 15, and 20 h. Duration longer than 20 h is not simulated, since it is unlikely to travel for that long duration continuously. Nonetheless, user can break down a longer travel to a number of shorter trips of manageable length. Also, the duration length seems reasonable for Beijing, China, where all the GPS trajectories are exclusively collected, since it covers an area of about 16,000 km^2. For each duration level, we generated 1,000 queries. Since user query $Q = \{q_s, q_d, q_t\}$ is composed of two points, we generate two sets of GPS coordinates randomly. Here, we put some constraints so that the generated queries follow normal distribution in terms of the distance between the start and end points. Figure 9 shows that the 1,000 queries generated for each level follows a normal distribution.

The maximum distance between two points is set to $2 \times q_t$ to increase chances for some round-trip like itineraries (shorter distance between) and to guarantee enough locations are added for comparison. For instance, we limit the distance between start and end points for duration 20 h to 40 km, so that the query may return results by providing enough time for traveling and staying.

5.2.2 User study

In user study, we recruited 10 participants who are currently active residents and have lived in Beijing for preferably at least 3 years (average of 3.8 years), since our GPS logs are exclusively collected from the past three years. We asked each participant to use our system to generate itineraries by selecting a start location, an end location, and duration of their choice. The recruited participants knew the Beijing area well and generated queries in their choice of locations which they were familiar with. Each user submitted 3 queries and gave ratings to 3 itineraries generated by our method and two other baseline methods. They carefully reviewed locations and

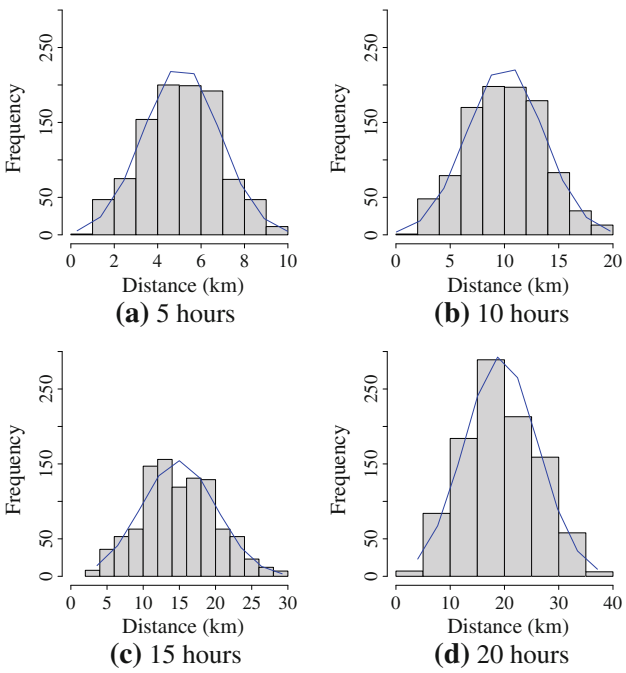


Fig. 9 Distribution of distance in simulated queries

Table 1 Questions for itinerary evaluation

Criteria	Question
Elapsed time	How efficient is the itinerary in terms of the duration? (1–5)
Stay and travel time	How reasonable/appropriate are staying time and traveling time? (1–5)
Interest	How interesting/popular and representative are the included locations? (1–5)

Table 2 User's rating on the overall itinerary

Ratings	Explanation
2	This itinerary is realistic and I like most of directional instructions
1	I would take this itinerary with minor changes
0	I would have taken different directions, but don't oppose the given itinerary
−1	This itinerary is unrealistic and poorly constructed

sequences in the itinerary without knowing about the methods that produced results. Participants took about 30 min to completely review 3 sets of 3 itineraries where they were allowed to browse through 3 different itineraries for the query to give relative ratings after comparison. We asked participants following questions to give scores for each generated itinerary in different aspects (score of 1 being the lowest and 5 represents the highest score for better performance) as shown in Table 1. Also, users rated itineraries according to relevance score presented in Table 2.

5.2.3 Baselines

We compared the result of our recommendation with two other baseline methods, Rank-by-Time (RbT) and Rank-by-Interest (RbI). RbT recommends an itinerary with the highest elapsed time usage. Ideally, it recommends an itinerary with the elapsed time equal to the duration in the query, if there is such candidate exists. Similarly, RbI ranks the candidates in the order of total interest of locations included in the itinerary. So the candidate with the highest interest density ratio is recommended.

5.2.4 Mobile+Cloud configuration

For practical mobile application deployment, we adopted Mobile+Cloud architecture. In the mobile part, we keep the number of tasks to minimal and include processes that are light and essential that cannot be processed elsewhere. In the cloud, it takes care of social itinerary recommendation and data mining. Figure 10 shows the Mobile+Cloud architecture. We show the performance advantage in Mobile+Cloud architecture since the recommendation process is time-consuming for mobile device. For implementation, we used a commercial mobile phone, Samsung SCH-M490 running Windows Mobile 6.1 at CPU clock of 806 MHz. For implementing the cloud side, we used a

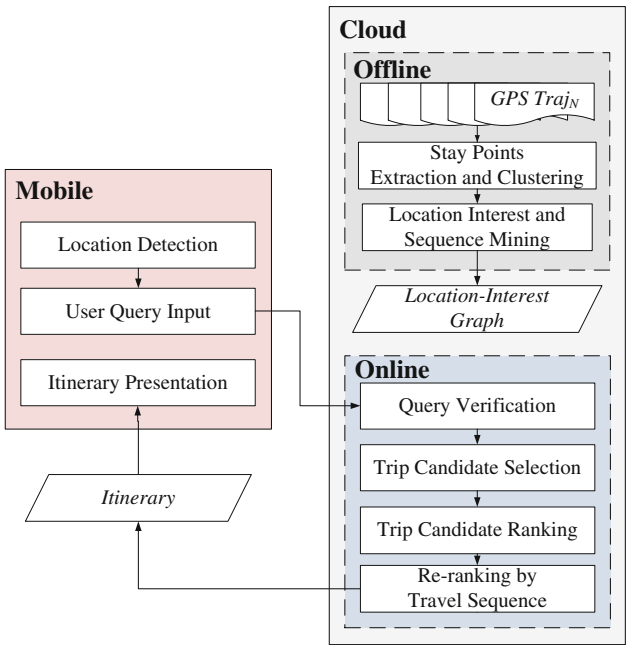


Fig. 10 Mobile+Cloud architecture

server PC running Intel Xeon CPU clock of 2.40 GHz with 2 processors, main memory of 4.00 GB, and Windows Vista Enterprise Service Pack 2. For the communication between the mobile device and the cloud service, we used Microsoft Windows Communication Foundation (WCF) for mobile as a client and the cloud service as a WCF service.

5.3 Results

5.3.1 Simulation

We generated 1,000 queries for each time level (5, 10, 15, and 20) and ran through 3 algorithms. For the duration of 5 h, only 452 itinerary results were retrieved. For the duration of 10, 15, and 20 h, 935, 961, and 973 itinerary results are acquired, respectively. There are three reasons

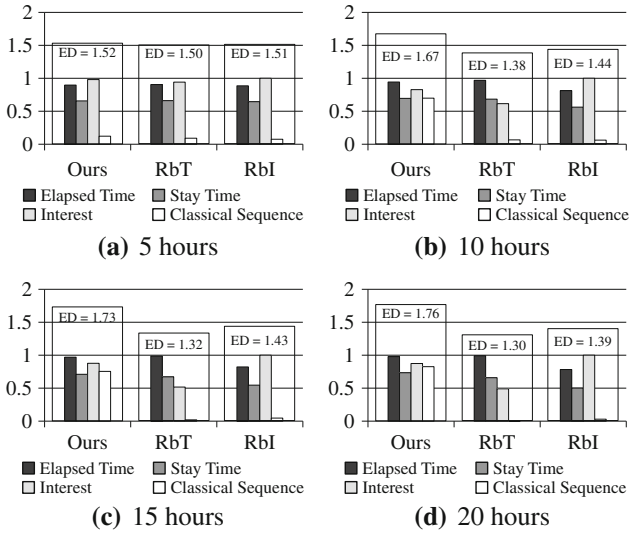


Fig. 11 Simulation results showing the average quality of itinerary generated by different methods

that not all queries returned results. First reason is that simply there was not enough time to go from a start location to an end location. Even though the queries would pass initial query verification, there may be very few or no shorter directions to the end location while consuming the specified duration. Second reason is that as shown in Fig. 8a, there are areas with very few or no locations at all. So when the given time is short and the user starts from one of these empty areas, the most of time is used up to go to a nearby location, yielding no results. Third reason is that user starts at a location that has very few outgoing edges, in that case, user might end up in dead end early even though there are plenty of remaining time. For the recommended itineraries, we looked closely at the average of elapsed time, stay time, interest, classical sequence, and Euclidean distance. Figure 11 shows the result for four different time levels. As expected, the baseline algorithmsRbTand RbI yield best results in the aspect of elapsed time and interest, respectively. However, the difference is minimal in the 5-h level. All three algorithms produced similar quality results. If the duration is very short, then there are not many candidates to consider and then many of them would overlap anyway. This explains almost identical graphs in Fig. 11a. The difference gets larger and noticeable as the duration gets longer. Still baseline algorithms successfully recommend itineraries that perform well in only one aspect. RbT has lower average of interest score compared to RbI and our algorithm. Also, RbI has lower average of elapsed time compared to RbT and ours. Furthermore, both baseline algorithms produce itineraries that are poor in classical sequence aspect as defined in Definition 8. The result shows that both RbT and RbI fail to produce itineraries revisiting classical sequence, which means that locations visited in sequence are not practical nor realistic. So we can observe on the average RbT and RbI will produce a biased or skewed itinerary focusing on only one attribute in a long term. On the other hand, our algorithm produces well-balanced itineraries in all four aspects. The Euclidean distance value gives a good indication that our algorithm produces balanced itinerary overall and even the recommended itineraries are comparable in other factors that are specialized by baseline algorithms. By looking at the Euclidean distance value, we observe that the performance of our algorithm increases with time whereas two baseline algorithms suffer from performance degradation.

5.3.2 *User study*

5.3.2.1 Itinerary at equilibrium For 10 participants' 30 queries over Beijing area, we observed the equilibrium of different itinerary attributes in our algorithm compared to the baseline algorithms. As we observed from the simulation, our algorithm produces an itinerary that is well balanced in the four attributes. So in this user study, we show that our algorithm produces results that are nearly equal to the baselines that specialize in a certain single attribute. For instance, we check how our result compares with RbT produced itinerary in terms of elapsed time, stay time, and travel time. Since RbT produces results that maximize the time use, we wanted to check whether the difference user perceives is significant compare to our result that produces well-balanced and nearly close result. Table 3 shows the comparison between our algorithm and RbT in terms of time use, as the T-test reveals that there is no significant advantage in perceived elapsed time, stay time, and travel time from using RbT over ours. Actually, in 30 queries in our user study, itineraries recommended by our algorithm received higher scores compared to that of RbT. Similarly, we compared our result in terms of locations interest included in the itinerary as shown in Table 3. Here, again our results scored higher, and the T-test reveals that there is no significant advantage in perceived interest from using RbI over ours.

5.3.2.2 Ranking ability Table 4 shows the ranking ability of different methods measured by MAP (mean average precision). MAP for a set of queries is the mean of the average precision scores for each query.

Table 3 Comparison of temporal and location interest attributes

Attributes	Ours	Rank-by-Time	*T* test
Elapsed time	3.97	3.67	$p \not\leq 0.01$
Stay and travel time	3.60	3.27	$p \not\leq 0.01$
Interest	3.27	2.92	$p \not\leq 0.01$

Table 4 Ranking ability of different methods

Measurement	Ours	Rank-by-Time	Rank-by-Interest
MAP	0.684	0.622	0.645

Table 5 Recommendation processing time (s) comparison for 15 h, (L) indicates the longest/largest and (S) indicates the shortest/smallest

	Mobile_Only	Mobile+Cloud	Candidates
Q1	367.658	15.847	6,472
Q2	449.245	20.988	1,165
Q3	267.207	13.545	4,603
Q4	1,081.038 (L)	27.489 (L)	30,734 (L)
Q5	460.502	20.242	6,375
Q6	174.044	10.200	45
Q7	123.984	6.181	17,899
Q8	291.448	15.292	789
Q9	1.647 (S)	2.517 (S)	21 (S)
Q10	19.958	3.472	503

$$\text{MAP} = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{18}$$

where Q is the number of queries. We treat 30 recommended itineraries as a ranked retrieval run. In our experiment, MAP stands for the mean of the precision score after each relevant itinerary is retrieved, which is determined by users. We consider an itinerary relevant, if its relevance score is 1 or 2 as shown in Table 2. For 30 itineraries generated for three different methods, the result is shown in Table 5. Our method showed a better performance compared to other baselines.

5.3.3 Mobile+Cloud configuration

We compared the time it takes to recommend an itinerary in standard-alone mobile approach (Mobile_Only) and cloud approach (Mobile+Cloud). In Mobile_Only, recommendation is operated fully in the mobile phone. Mobile+Cloud has distributed tasks between the mobile and the cloud. All collected trajectories are data mined in offline in a separate server, and the resulting Location-Interest Graph (G) is used in both Mobile_Only and Mobile+Cloud modes. The data size of Location-Interest Graph (G) is small, which only contains information on 1) location itself (interest, typical staying time) and 2) relationship between locations (typical traveling time, classical travel sequence). We measured the processing time for three durations (5, 10, and 15 h). We did not test with 20 h, since it took unreasonable long time to measure on the mobile side. For each duration level, 10 unique test queries are used. Figure 12 shows the experiment results.

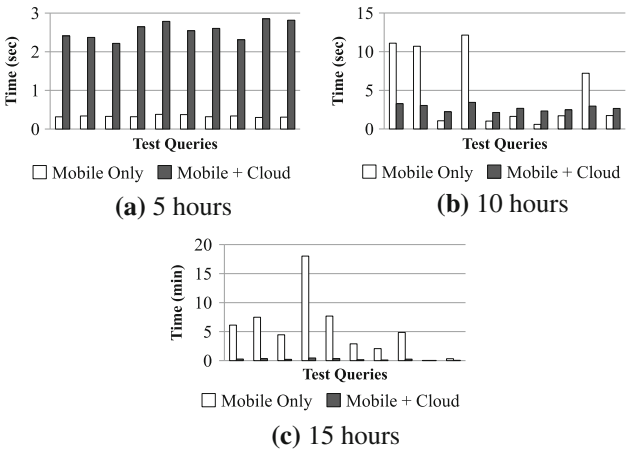


Fig. 12 Performance comparison of Mobile+Cloud architecture

In 5 h, Mobile_Only was faster than Mobile+Cloud. This can be explained by the fact that the finding trip candidate process in shorter duration does not take much time since we cannot add many locations nor travel further given shorter duration. In Mobile_Only, it took less than 1 second to find candidate trips and get the recommended itinerary for 5-h duration. However, in Mobile+Cloud it took couple of seconds. So even though the actual processing was done much faster in the cloud, the time needed for binding the mobile client with the WCF service took some time and it also took more time for communication. So in our observation, the cloud approach had at least couple of seconds for communication. The performance gap is noticeable for time duration of 10 and 15 h. The longer duration means that there are more number of candidate trips to find and rank. As shown in Fig. 12b, the processing time for Mobile_Only takes about three times longer than Mobile+Cloud. For some queries, Mobile_Only was faster, this is due to the fact that the result contains only a small number of candidates. In 15 h, the high computation burden on Mobile_Only is clear. As shown in Fig. 12c, Mobile_Only takes at least few minutes to recommend an itinerary whereas Mobile+Cloud can handle the request within a minute. In comparison, Mobile_Only failed to produce results in neither real-time nor interactive time. Some test queries lasted for over 15 minutes, which is unbearable in real use cases.

5.4 Discussions

5.4.1 Temporal aspects

The length of duration is an interesting attribute to look at. Many participants used duration between 6 and 12 h. It supports our initial assumption that people would not have such a long journey and keep them in a manageable size. For shorter duration, the measured quality of itineraries

was less for our algorithm based on Euclidean distance of attributes. Conversely, two baseline algorithms produced the best quality at the shorter duration and recommended less efficient itineraries with longer duration. In extreme cases though, it was possible for baseline algorithms such as RbI to recommend an itinerary that only contains a couple of interesting locations without spending all available time. However, since duration was used as a stopping condition in selecting candidates, most recommended itineraries spend good ratio of available time in simulation and in real user queries alike. Also, users were more judgmental of and interested in traveling time (on average, less than 1 h) than staying time at locations (on average, greater than 1 h). Therefore, it would be interesting research direction to give a good projection on traveling time including many options of transportation modes.

5.4.2 Location interest and classical sequence

As shown in Fig. 11, our algorithm produced a balanced itinerary with higher classical sequence scores. In algorithmic level, our algorithm showed a great performance advantage in terms of the four attributes including classical travel sequence. However, in real queries by users, it was difficult to measure location interest and classical travel sequences from the recommended itinerary. Even though an itinerary is composed of many locations and sequences, we only asked the participants to give ratings for the overall location interest and classical sequence. So they gave high score for classical travel sequences they could find and gave lower score for any abnormal sequences that sometimes balances each other out. So this is different from our simulation where each location interest and classical travel sequences were accumulated to give the overall score. In our current algorithm, we only consider increment of score for location interest and any classical travel sequences found, yet in the real situation, we might need to decrease score or give penalties for totally uninteresting locations and awkward sequences. This is another research direction that can help recommend a better itinerary by avoiding (possibly known) bad sequences in the first place.

5.4.3 Mobile deployment

Table 5 shows performance comparison results for 15-h duration. When we closely observe the query that took the longest time to process, we can notice that those queries resulted in a very large number of initial candidates. Also, the query with the shortest processing time deals with a very small number of initial candidates. So we have two heuristics that we can use to choose between Mobile_Only and Mobile+Cloud.

1. If the time duration is large ($q_t > 5$ h), we are better off to use Mobile+Cloud. Only use Mobile_Only for a very short duration, since Mobile+Cloud approach has a reasonably low lower-bound around 2 seconds to match the performance of mobile approach.
2. If we know that the query will generate many candidate trips, then we should use Mobile+Cloud. We can simply check this by counting the number of outgoing edges of start location and incoming edges of end location. If these numbers are large, then the number of candidate trips will be large also.

5.4.4 Alternative sources of digital trails

In our work, we used GPS trajectories as the primary means of digital trails generated by users. Taking this idea further, different combinations of digital trails can be incorporated for cross-checking and improving accuracy of the work proposed here. The notable and relevant research domain deals with many resources found on the web, especially with user comments [27], geo-tagged multimedia such as Flickr photo streams [4, 10] and social network services. For end-user side, [24] proposed mashup paradigm in ubiquitous computing environment through augmented reality where users can produce content and services attached to real-world objects. Recently, Zhang et al. proposed "social and community intelligence (SCI)" to learn from different patterns of individual, group, and societal behaviors detected pervasive sensing and context-aware computing environment [26].

5.4.5 Limitation

There are number of limitations in our approach. Since our method relies on user-generated GPS trajectories, it is important to collect a good dataset. Our data are collected by highly motivated and active volunteers as a trusted source, but in real case including different Web 2.0 check-in sites, many user-generated GPS trajectories need to be validated. In our current setting, we use stay point detection, stay point clustering, and location clustering to remove much noise, but stronger means of detecting abnormities in the source are required. One of our goal in itinerary recommendation was to minimize user intervention and automate the process with a simple query. However, to get a more personalized and accurate itinerary beyond itinerary recommendation for new travelers, we need to consider various contextual information, such as different transportation modes and ranges of itineraries. In our previous works [30, 31], transportation mode is considered, but these are not fully incorporated with itinerary recommendation. Also, our itinerary recommendation is

tested in city level, but scalability is another direction for future work. Lastly, semantic aspects need to be combined with locations, since some locations have opening and closing hours and may be affected by the contextual situations such as weather conditions, traffic jams, festivities, and crowded holiday periods. Our current itinerary recommendation framework can be improved by considering these issues.

6 Conclusion

In this paper, user-generated digital trails such as GPS trajectories are collected and data mined to extract collective social intelligence. Specifically, we used GPS trajectories from 125 users to build *Location-Interest Graph* that reflects travel history and experience of travel experts and active residents. By using *Location-Interest Graph* and the proposed itinerary model, itinerary is recommended according to a user-supplied query. To recommend an efficient itinerary, we collectively used four attributes mined from out dataset to generate balanced and practical itineraries. When compared to baselines RbT and RbI, our proposed method was competitive in both time use and interest level and outperformed baselines in classical travel sequence aspect in both simulation mode and through user study. Especially, the best performance of our method was observed in the longer duration. Also, we found that computation intensive task such as social itinerary recommendation can be distributed effectively in Mobile+Cloud architecture for practical mobile adaptation and application deployment.

For future work, we would like to give better indications in traveling time between locations by differentiating use of transportation modes. Also, hybrid itinerary recommendation based on other sources of digital trails and contextual information such as geo-tagged multimedia and check-ins are potential works.

Acknowledgments This research was supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA), under the Culture Technology(CT) Research & Development Program 2011 and Microsoft Research Asia (MSRA). We also like to acknowledge anonymous reviewers for providing invaluable comments and suggestions.

References

1. Ardissono L, Goy A, Petrone G, Segnan M (2005) A multi-agent infrastructure for developing personalized web-based systems. ACM Trans Internet Tech 5:47–69. doi:10.1145/1052934.1052936
2. Ashbrook D, Starner T (2003) Using GPS to learn significant locations and predict movement across multiple users. Pers Ubiquit Comput 7:275–286. doi:10.1007/s00779-003-0240-0
3. Cao L, Krumm J (2009) From GPS traces to a routable road map. In: Proceedings of GIS 2009, pp 3–12. doi:10.1145/1653771.1653776
4. Chodhury MD, Feldman M, Amer-Yahia S, Golbandi N, Lempel R, Yu C (2010) Automatic construction of travel itineraries using social breadcrumbs. In: Proceedings of HT 2010, pp 35–44. doi:10.1145/1810617.1810626
5. Dunstall S, Horn MET, Kilby P, Krishnamoorthy M, Owens B, Sier D, Thiebaux S (2003) An automated itinerary planning system for holiday travel. Inf Technol Tour 6:195–210
6. GeoLife GPS Trajectories (2010) http://bit.ly/bd78Rt, http://bit.ly/gY2JHq
7. Giannotti F, Nanni M, Pinelli F, Pedreschi D (2007) Trajectory pattern mining. In: Proceedings of KDD 2007, pp 330–339. doi:10.1145/1281192.1281230
8. Girardin F, Calabrese F, Flore F, Ratti C, Blat J (2008) Digital footprinting: uncovering tourists with user-generated content. IEEE Pervas Comput 7:36–43. doi:10.1109/MPRV.2008.71
9. Haversine Formula (2010) http://en.wikipedia.org/wiki/Haversine_formula. Accessed 24 Nov 2010
10. Hollenstein L, Purves R (2010) Exploring place through user-generated content: using Flickr to describe city cores. J Spat Inf Sci 1:21–48. doi:10.5311/JOSIS.2010.1.3
11. Huang Y, Bian L (2009) A Bayesian network and analytic hierarchy process based personalized recommendations for tourist attractions over the Internet. Expert Syst Appl 36:933–943. doi:10.1016/j.eswa.2007.10.019
12. Kim J, Kim H, Ryu JH (2009) TripTip: a trip planning service with tag-based recommendation. In: Proceedings of CHI EA, pp 3467–3472. doi:10.1145/1520340.1520504
13. Krumm J (2010) Where will they turn: predicting turn proportions at intersections. Pers Ubiquit Comput 14:591–599. doi:10.1007/s00779-009-0248-1
14. Kumar P, Singh V, Reddy D (2005) Advanced traveler information system for Hyderabad city. IEEE Trans Intell Transp 6:26–37. doi:10.1109/TITS.2004.838179
15. Li Q, Zheng Y, Xie X, Chen Y, Liu W, Ma WY (2008) Mining user similarity based on location history. In: Proceedings of GIS 2008, pp 1–10. doi:10.1145/1463434.1463477
16. Monreale A, Pinelli F, Trasarti R, Giannotti F (2009) WhereNext: a location predictor on trajectory pattern mining. In: Proceedings of KDD 2009, pp. 637–646. doi:10.1145/1557019.1557091
17. Stabb S, Werther H, Ricci F, Zipf A, Gretzel U, Fesenmaier D, Paris C, Knoblock C (2002) Intelligent systems for tourism. IEEE Intell Syst 17:53–66. doi:10.1109/MIS.2002.1134362
18. Spherical Law of Cosines (2010) http://en.wikipedia.org/wiki/Spherical_law_of_cosine. Accessed 24 Nov 2010
19. Suh Y, Shin C, Woo W (2009) A mobile phone guide: Spatial, personal, and social experience for cultural heritage. IEEE Trans Consum Electr 55:2356–2364. doi:10.1109/TCE.2009.5373810
20. Suh Y, Shin C, Dow S, MacIntyre B, Woo W (2010) Enhancing and evaluating users' social experience with a mobile phone guide applied to cultural heritage. Pers Ubiquit Comput 1–17 (online first). doi:10.1007/s00779-010-0344-2
21. Tai CH, Yang DN, Lin LT, Chen MS (2008) Recommending personalized scenic itinerary with geo-tagged photos. In: Proceedings of ICME 2008, pp 1209–1212. doi:10.1109/ICME.2008.4607658
22. Takeuchi Y, Sugimoto M (2009) A user-adaptive city guide system with an unobtrusive navigation interface. Pers Ubiquit Comput 13:119–132. doi:10.1007/s00779-007-0192-x
23. Werthner H (2003) Intelligent systems in travel and tourism. In: Proceedings of IJCAI 2003, pp 1620–1628
24. Yoon H, Woo W (2009) CAMAR mashup: empowering end-user participation in U-VR environment. In: Proceedings of ISUVR 2009, pp 33–36. doi:10.1109/ISUVR.2009.22

25. Yoon H, Zheng Y, Xie X, Woo W (2010) Smart itinerary recommendation based on user-generated GPS trajectories. In: Proceedings of UIC 2010, pp 19–34. doi:10.1007/978-3-642-16355-5_5
26. Zhang D, Guo B, Li B, Yu Z (2010) Extracting social and community intelligence from digital footprints: an emerging research area. In: Proceedings of UIC 2010, pp 4–18. doi:10.1007/978-3-642-16355-5_4
27. Zheng VW, Zheng Y, Xie X, Yang Q (2010) Collaborative location and activity recommendations with GPS history data. In: Proceedings of WWW 2010, pp 1029–1038. doi:10.1145/1772690.1772795
28. Zheng VW, Cao B, Zheng Y, Xie X, Yang Q (2010) Collaborative filtering meets mobile recommendation. In: Proceedings of AAAI 2010, pp 238–241
29. Zheng Y, Wang L, Zhang R, Xie X, Ma WY (2008) GeoLife: managing and understanding your past life over maps. In: Proceedings of MDM 2008, pp 211–212. doi:10.1109/MDM.2008.20
30. Zheng Y, Li Q, Chen Y, Xie X, Ma WY (2008) Understanding mobility based on GPS data. In: Proceedings of Ubicomp 2008, pp 312–321. doi:10.1145/1409635.1409677
31. Zheng Y, Liu L, Wang L, Xie X (2008) Learning transportation mode from raw GPS data for geographic applications on the web. In: Proceedings of WWW 2008, pp 247–256. doi:10.1145/1367497.1367532
32. Zheng Y, Chen Y, Xie X, Ma WY (2009) GeoLife2.0: a location-based social networking service. In: Proceedings of MDM 2009, pp 357–358. doi:10.1109/MDM.2009.50
33. Zheng Y, Zhang L, Xie X, Ma WY (2009) Mining interesting locations and travel sequences from GPS trajectories. In: Proceedings of WWW 2009, pp 791–800. doi:10.1145/1526709.1526816
34. Zheng Y, Zhang L, Xie X, Ma WY (2009) Mining correlation between locations using human location history. In: Proceedings of GIS 2009, pp 472–475. doi:10.1145/1653771.1653847
35. Zheng Y, Chen Y, Li Q, Xie X, Ma WY (2010) Understanding transportation modes based on GPS data for web applications. ACM Trans Web 4:1–36. doi:10.1145/1658373.1658374
36. Zheng Y, Xie X, Ma WY (2010) GeoLife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull 33:32–39
37. Zheng Y, Xie X (2011) Learning travel recommendations from user-generated GPS traces. ACM Trans Intell Syst Technol 2:1–29. doi:10.1145/1889681.1889683
38. Zheng Y, Zhang L, Ma Z, Xie X, Ma WY (2011) Recommending friends and locations based on individual location history. ACM Trans Web 5:1–44. doi:10.1145/1921591.1921596