IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INNOVATION AND APPLICATIONÃ, OF INTELLIGENT PROCESSING OF DATA, INFORMATION AND KNOWLEDGE AS RESOURCES IN EDGE COMPUTING

Received May 14, 2019, accepted June 13, 2019, date of publication June 17, 2019, date of current version July 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923459

Group-Wise Itinerary Planning in Temporary Mobile Social Network

YUYU YIN[1,2] JING XIA[1,2], YU LI[1,2], YUESHEN XU[3,4], WENJIAN XU[5], AND LIFENG YU[6]

[1]School of Computer, Hangzhou Dianzi University, Hangzhou 310018, China
[2]Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou 310018, China
[3]School of Computer Science and Technology, Xidian University, Xi'an, 710126, China
[4]Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, China
[5]Alibaba Co., Ltd., Hangzhou 311121, China
[6]Hithink RoyalFlush Information Network Co., Ltd., Hangzhou 310023, China

Corresponding author: Yu Li (liyucomp@hdu.edu.cn)

This paper was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1400601, in part by the National Natural Science Foundation of China under Grant 61802098, in part by the Natural Science Foundation of Zhejiang Province under Grant LY16F020017, in part by the Shaanxi Province under Grant 2018JQ6050, and in part by the Fundamental Research Funds for Central Universities under Grant JBX171007.

ABSTRACT Temporary mobile social networks has been used at hotels, concerts, theme parks, and sports arenas, where people form a mobile social group for a short time with a common interest or activity. People confined to such specific places or activities are allowed to join the temporary mobile social networks using their main social network accounts (e.g., Foursquare, Facebook). Users registered for the same business/research conference may have common connections and thus may be willing to travel together in the conference city. Traveling with temporal friends can improve the mobile users' experiences as well as help them save money. Currently, renting cars to travel around becomes very general, and one car usually can contain at least four guests. Therefore, traveling with temporal friends can help those guests save their travel cost, such as renting cost and oil cost. To this end, in this paper, we propose a group-wise itinerary planning framework to improve the mobile users' experiences. The experiment results over real data sets illustrate the effectiveness of our proposed framework.

INDEX TERMS Travel planning, temporal mobile social network, mobile computing.

I. INTRODUCTION

The concept of temporary mobile social networks has been used at hotels, concerts, theme parks, and sports arenas, where people form a mobile social group for a short time with a common interest or activity. People confined to such specific places or activities are allowed to join the temporary mobile social networks using their main social network accounts (e.g., Foursquare, Facebook). Then they could text the entire group, share pictures or locations, and set up subgroups, etc. For example, LobbyFriend is a hotel social network designed to connect a current guest to other fellow guests and the hotel staff, as well as to a live feed of what is happening in or around the hotel. When a guest checks out of the hotel, all her interactions in the temporary mobile social network are erased.

A key functionality in temporary mobile social networks is traveling. For instance, in hotel social networks, guests may come for the same business/research conference, that is, those guests may have common connections and thus are willing to travel together in the conference city. Traveling with temporal friends may improve the feeling of those guests as well as help them save money. Currently, renting cars to travel around becomes very general, and one car usually can contain at least 4 guests. Therefore, traveling with temporal friends can help those guests save their travel cost, such as renting cost and oil cost.

However, simply and indiscriminately organize guests into temporal mobile social groups may risk the users' experiences. The main reason is that, even though those guests gather together for the same conferences/meetings/concert, their available times and interests may vary a lot. For instance, some are not willing to wake up until 10:00 am, while some may enjoy the sunshine at 8:00. Some guests are interested in museums while some like delicious foods. It is not

The associate editor coordinating the review of this manuscript and approving it for publication was Honghao Gao.

TABLE 1. Differences between prior work and our system.

Paper	User's deadline	User's preference	POI's category	POI's popular	POI's crowdedness	Group recommendation	Group size limit
[1]	✓	✓	×	×	×	×	×
[2]	×	✓	✓	×	×	✓	×
[3]	✓	✓	✓	✓	×	✓	×
[4]	✓	✓	✓	✓	✓	×	×
[5]	✓	✓	✓	✓	✓	×	×
[6]	✓	✓	✓	✓	✓	×	×
ours	✓	✓	✓	✓	✓	✓	✓

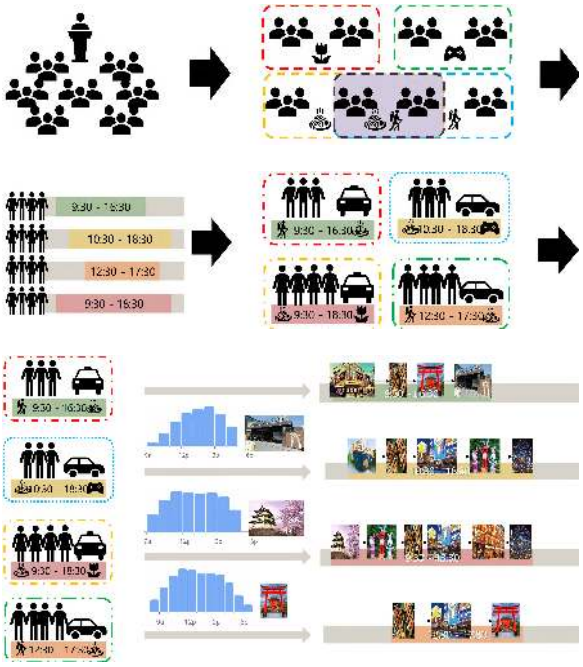

FIGURE 1. Example of the framework.

appropriate to organize users with very different active times into one group, since this assignment may lead a waste of time. Also, putting guests with very different interest preferences into the same group is not a good choice, as their requirements cannot be satisfied along a single itinerary.

To this end, we propose a framework designed for temporary mobile social networks to suggest group-wise itinerary plans. In this framework, when a guest registers at a conference/meeting, the hotel will collect his available time periods for traveling and interest preferences. As illustrated in Figure 1, we first put the relevant guests into a group and the group sizes are limited by car capacity (e.g. 4). Then, we recommend the itinerary for each group such that guests' traveling experiences can be improved and their time limits can be satisfied. For the relevance requirement in guest grouping steps, we combine available-time and interest-preference factors to measure the relevance of registered guests. Specifically, we generate the mobile social-temporal groups through considering guests' interest preferences and available times step by step.

There exist some researches studying similar problems as this paper does. To compare with them, we summarizes the differences between our paper and existing frameworks in Table 1. In this paper, to improve the quality of suggested itineraries, we consider the deadline and preference of each user, the popularity and crowdedness (e.g. waiting time) of each point of interest (POI). For each generated group, we limit its size according to general capacity of cars, that is 4. As far as we know, no paper has studied the same problem as we do. The contributions of our work can be summarized as follows:

- We propose the group-wise itinerary planning framework, which can be used to improve the traveling experiments of users in a mobile social-temporal network.
- We design relevance measure functions to groups users in terms of their interest preferences and available times.
- We discuss different the itinerary recommendation methods for group-wise users.
- We evaluate our proposed framework through extensive experiments based on a real social network data set.

The rest of this paper is organized as follows. We first review the preliminaries of this paper and present our proposed framework in Section II. Then, we train the city model in Section III and allocate car groups in Section IV. In Section V, we describe the itinerary recommendation methods. Section VI analyzes experimental results. And related literatures are discussed in Section VII. Finally, Section VIII concludes this paper.

II. PRELIMINARIES AND PROBLEM STATEMENT

In this section, we first illustrate the preliminaries definitions used in this paper and then formally describe the framework proposed in this paper.

$U = \{u_1, \ldots, u_n\}$ be the set of users. $V = \{v_1, \ldots, v_l\}$ be the set of POIs. Given that $C = \{c_1, \ldots, c_o\}$ denotes the set of all POI categories, each POI v belongs to a category $c \in C$.

A. POI

Given a user u who has visited k POIs, we define his/her travel history as an ordered sequence, $Sq_u = ((v_1, t^a_{v_1}, t^d_{v_1}), \ldots, (v_l, t^a_{v_l}, t^d_{v_l}))$, with each triplet $(v_x, t^a_{v_x}, t^d_{v_x})$ comprising the visited POI v_x, and the arrival time $t^a_{v_x}$, and departure time $t^d_{v_x}$. Thus,the visit duration at POI v_x can be determined by the difference between $t^a_{v_x}$ and $t^d_{v_x}$.

B. AVERAGE POI VISIT DURATION

Given a set of travel histories for all users U, we determine the average visit duration for a POI v as follows:

$$Dur(v) = \frac{1}{n} \sum_{u \in U_{v_x}} \sum_{v_x \in Sq_u} (t^d_{v_x} - t^a_{v_x}) \delta(v_x = v), \quad \forall v \in V \quad (1)$$

C. POI POPULARITY

The popularity of a POI v is defined as: the number of visits $Visit(v)$ at POI v in the travel history of all users, normalize the value to [0, 1]. We use photo number as a approximation for visit number.

$$Pop(v) = \frac{Visit(v)}{Max(Visit(v_i))} \quad \forall v_i \in V \quad (2)$$

For example, if there are 3 POIs in total, and $Visit(1) = 10$, $Visit(2) = 50$, $Visit(3) = 20$, then $Pop(1) = 0.2$, $Pop(1) = 1$, $Pop(3) = 0.4$.

D. POI CROWDEDNESS

For POI v, we define the crowdedness $Crd(v, t)$ at time t using the number of users visited POI v in time t divided by the max visit user number during history period of time T_p. For example, visit user number at a POI is $\langle 100, 200, 300 \rangle$ at:9:00,10:00 and 11:00. The crowdedness during these three hours is $\langle 0.3, 0.6, 1.0 \rangle$.

The traffic crowdedness are widely provided by several map services. Obtaining accurate estimates of POI crowdedness in future can be modeled as a time series regression and forecasting problem which is an important problem but is beyond the scope of this paper, when such estimation method is available it can be incorporated as it is in our framework. Such as ARMA [7] model predict the number of people per hour on each POI using the history data. In our implementation we use the average number of people per hour on each POI in the history data as the predict and we calculate the POI's visit number by counting its photo number as a approximation:

$$Crd(v, t) = \frac{Visit(v, t)}{Max(Visit(v, t_p))} \quad \forall t_p \in T_p \quad (3)$$

E. USER INTEREST PREFERENCE

Given that $Visit_u(v)$ as the total number of visits by user u at POI v during the history, $Visit_u(c)$ as the total number of visits by user u at all the POI belongs to category c, we use the set of photos taken by user u as a approximation and we define the interest preference of user u in category c as:

$$Int_u(c) = \frac{Visit_u(c)}{Max(Visit_u(c_i))} \quad \forall c_i \in C \quad (4)$$

where $Visit_u(c) = \sum_{v \in c} Visit_u(v)$. In short, the interest preference of a user u in POI category c is based on the number of photos of POI that belongs to category c, and normalize the value to [0, 1]. The intuition behind this definition is that a user is more likely to take more photos at a POI category that interests him/her, and less photos otherwise.

$$\vec{I_u} = \langle Int_u(c_i), \ldots, Int_u(c_o) \rangle, \quad \forall \{c_1, \ldots, c_o\} \in C \quad (5)$$

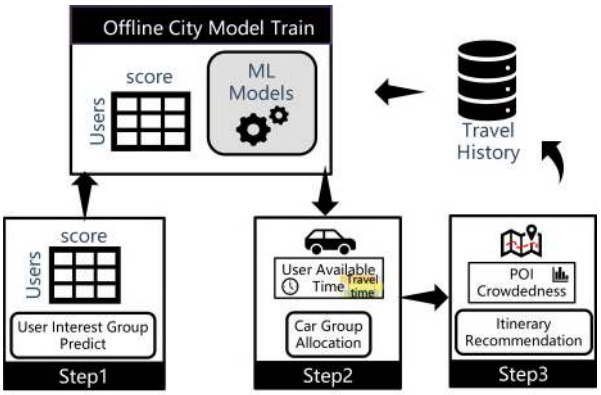

FIGURE 2. Framework.

F. ITINERARY PLANNING FRAMEWORK

As shown in Fig. 2, our proposed group-wise itinerary planning framework first group users into level-1 groups according to their interest preferences. After that, users' available times are taken into consideration to subdivide these level-1 groups into smaller groups, and each group's size is limited to the car capacity (e.g., 4). With generated groups, route recommender will suggest the itinerary for each group to satisfy group members' preferences as well as their time constraints.

III. STEP 1 OFFLINE CITY MODEL TRAIN

Because the categories are fairly stable in a city and there are not so many different kinds of interest preference among users according to history travel itinerary. In order to improve users' experiences by provide relevant service and improve the system efficiency in following steps, we train the offline city model and find the principal interest patterns in the city which guides us what kind of interest similarity threshold(CT) we should set in Step 2.

Given that the n users are belong to k interest patterns, let $P = \{p_1, \ldots, p_k\}$ be the set of interest preference patterns in the city. p_k is a city interest preference pattern, which is a clustering center and a user cluster $pt_k = \{u_1, \ldots, u_q\}$ represented by p_k denote the q users in the kth interest pattern have similar interest preference.

The k centers are:

$$p_k = \frac{1}{|pt_k|} \sum_{u \in pt_k} I_u, \quad \forall pt \in P \tag{6}$$

In this paper, we use Gaussian Mixture Model(GMM) to cluster users and analyze travel pattern. Formed by using a suficient number of Gaussians, and by adjusting their means and covariances as well as the coeficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy [8]. Effective clustering maximizes intra-cluster similarities and minimizes inter-cluster similarities [9]. The interest preference similarity between two users are calculated by cosine similarity $Cos(u_i, u_j) = \frac{\vec{I_{u_i}} \cdot \vec{I_{u_j}}}{\|\vec{I_{u_i}}\| \cdot \|\vec{I_{u_j}}\|}$, which tells us how similar two users are in terms of their interest preference.

The average cosine similarity of all pair-wise combinations of users in a cluster pt is defined as intra-cluster similarity $Intra(pt_k)$:

$$\frac{1}{(|pt_k| \cdot (|pt_k| - 1))} \sum_{u_i \in pt_k} \sum_{u_j \in pt_k, u_j \neq u_i} Cos(u_i, u_j) \tag{7}$$

The similarity between patterns is defined as inter-pattern similarity, for efficiency, we use the similarity between two cluster center rather than the similarity of all pair-wise combinations of two patterns' members respectively. The mean value of a Gaussian component is regarded as a clustering center.

Pattern inter similarity is defined as:

$$Inter(p_i, p_j) = \frac{\vec{p_i} \cdot \vec{p_j}}{\|\vec{p_i}\| \cdot \|\vec{p_j}\|}, \quad \forall p_i,\ p_j \in P \tag{8}$$

Formally, our goal is to find a best set of k patterns P that:

$$Max \frac{\frac{1}{k} \sum_1^k Intra(pt_k)}{\frac{1}{(k \cdot (k-1))} \sum_{p_i \in P} \sum_{p_j \in P, p_j \neq p_i} Inter(p_i, p_j)} \tag{9}$$

Suppose we find the best k patterns, and we part current users into k level-1 groups with distinguishing interest preference. In this way, users in the same level-1 group can be allocated by their available time in parallel which will reduce the server-side response time.

IV. STEP 2 CAR GROUP ALLOCATION

We aim to meet users' interests and time schedule while saving cost by grouping similar users according to the overlap of their available time, then allocate them into some small car groups and recommend them to travel together, even if they need to change their travel schedule slightly. In this section, we first introduce the notations used in group allocation, then we describe the methods used to allocation users in this paper, and finally we illustrate the evaluation metrics to measure the group allocation performance.

User label: From Step 1, we get the travel pattern model with k centers. We get users' query as $u = (I, t, v_s)$. We assign users into k level-1 groups by comparing $u.I$ with each k pattern centers then marked them with label $u.l$.

User's query: Users with travel pattern label, each user is defined as $u = \{I, l, t, v_s\}$, where $u.I$ is user's interest preference. $u.t = [t_s, t_e]$ is the user's start time and end time. $u.v_s$ is the user's start point.

Time Similarity: We calculate time period similarity between users u_i and u_j by jaccard similarity also known as intersection over union.

$$TS(u_i, u_j) = \frac{u_i.t \cap u_j.t}{u_i.t \cup u_j.t} \tag{10}$$

User Similarity: Then we calculate similarity between u_i, u_j by

$$Sim(u_i, u_j) = \mu TS(u_i, u_j) + (1 - \mu)\, Cos(u_i, u_j) \tag{11}$$

Cos is the interest preference similarity between two users.

Thresholds: We set threshold DT for maximum start/end time difference between two users, TT for the lowest time similarity and CT for the lowest interest cosine similarity of users in a car group. We use the minimum $Intra(pt_k)$ among all clusters in Step 1 as CT.

Car Capacity: CA The maximum user number of a group.

Occupancy Rate: The lowest car occupancy rate OR which means at least $OR \times CA$ users in a group, otherwise it's no need to group them up because everyone spends too much to rent a car.

Car Group $\mathcal{G} = \{g_1, \ldots, g_m\}$ is the set of car groups and each $|g|$ is smaller than CA and larger than $OR \times CA$. Each g is defined by (I, t_s, t_e, v_s), where $g.I$ is group's interest preference.

$$\vec{I_g} = \left\langle Int_g(c_1), \ldots, Int_g(c_o) \right\rangle, \quad \forall \{c_1, \ldots, c_o\} \in C \tag{12}$$

$g.t_s$ is the group's start time and $g.t_e$ is group's end time. $g.v_s$ is the group's start point. In this problem, we assume all users are start at the same query point.

Group's start time and end time:

$$g.t_s = \max_{(u \in g)} u.t_s \tag{13}$$

$$g.t_e = \min_{(u \in g)} u.t_e \tag{14}$$

Group's Interest Preference: One main challenge in recommending and planning itineraries for a group is the diverse interest preferences among group users. To address these diverse interest preferences, we construct a collective group interest preference. We define the interest preference of group g in category c as:

$$Int_g(c) = \omega \cdot rel(u, c) + (1 - \omega) \cdot (a_{max} - dis(u, c)), \quad \forall u \in g \tag{15}$$

where: a_{max} is $MaxInt_{u_i}(c_j)$ among all users in the group, the group relevance for category c is defined as $rel(u, c) = \frac{1}{|g|} \sum_{u \in g} Int_u(c)$, which captures the users' interest relevance of the category c_j by using the average preference score among group members. The group disagreement for category c is defined as $dis(u, c) = \frac{1}{(|g|(|g|-1))} \sum_{u_i \in g} \sum_{u_j \in g, u_j \neq u_i} |Int_{u_i}(c) - Int_{u_j}(c)|$.

Algorithm 1 Car Group Allocation

Input: a level-1 group g, CA, OR, DT, TT, CT
Output: $\mathcal{G} = \{g_1, g_2, \cdots, g_m\}$
1: $waitQueue \leftarrow \emptyset, \mathcal{G} \leftarrow \emptyset, singlePerson \leftarrow \emptyset$
2: $used \leftarrow \emptyset$
3: **while** $g \setminus used \neq \emptyset$ **do**
4: $u' \leftarrow$ first $u \in g \setminus used$,
5: $candi \leftarrow$ empty $MaxHeap(CA-1)$
6: **for** $u'' \in g \setminus \{used \cup u'\}$ **do**
7: **if** $SG(u', u'')$ **then**
8: $candi \leftarrow candi \cup u''$
9: **end if**
10: **end for**
11: **if** $OR \times CA - 1 \leq |candi| \leq CA - 1$ **then**
12: $g' \leftarrow u' \cup candi, \mathcal{G} \leftarrow \mathcal{G} \cup g'$
13: $used \leftarrow used \cup u' \cup candi$
14: **else**
15: $used \leftarrow used \cup u', waitQueue \leftarrow u'$
16: **end if**
17: **end while**
18: $g \leftarrow waitQueue$ **go to** line 2
19: **if** $u \in waitQueue \notin Groups$ **then**
20: $singlePerson \leftarrow singlePerson \cup u$
21: **end if**
22: return $\mathcal{G} \cup singlePerson$

Problem Define: Given (1) car capacity: CA, (2) occupancy rate OR, (3) users' queries U. Our goal is to group more people travel together and each group size is bounded by CA and OR, in the meantime, maximum total average cosine similarity and time similarity of users in each group including users who should travel alone.

A. METHOD

Since the optimal solution needs to check all possible combinations with vary high complexity, we design a greedy algorithm to allocation each level-1 groups into small car groups as described in Algorithm 1. We use a *waitQueue* to wait for more queries. Finally, when a group g created, their query would send to itinerary and they would get their recommended travel schedule, as in Setp 3. If the user is not satisfied with the result, he/she can quit the group and adjust his/her time schedule or interest preference as needed, then we can redo our car group allocation for the group members. The key idea is that, we use the following rules and use a fixed size heap to store $[OR \times CA - 1, CA - 1]$ most similar users for each user.

Rule 1: If u and u' both sent a query, the start time of u is earlier than u', we service u first. We organize user's query sorted by start time. If users start times are same, we want to keep them sorted by end time.

Lemma: Suppose there are $u_1.t_s = 9:00, u_2.t_s = 9:10, u_3.t_s = 9:20, u_4.t_s = 9:30, u_5.t_s = 9:40$, u_1 is the earliest user according to start time among users. We find u_1's similar users starting from 9:00, then we find u_2's similar users starting from 9:10 except u_1 whose start time earlier than himself. If u_2 can group with u_1, when we find u_1's similar user previously, we would group u_1 with u_2. Then when we consider u_2, u_1 and u_2 already in a group, so it's no need to consider u_2 with u_1.

TABLE 2. Users' similarity.

	u_1	u_2	u_3	u_4	u_5
u_1		0.8	0.8	0.7	0.7
u_2			0.8	0.7	0.7
u_3				0.8	0.7
u_4					0.7
u_5					

Rule 2: If u and u' both sent queries, we use function $SG(u, u')$ to determine whether u and u' should be allocated in a same group. $SG(u, u')$ returns *true* if u and u' satisfies DT, TT and CT, we call u is *feasible* for u' and u' is *feasible* for u.

Furthermore, although the number of similar users is not enough(lower than $OR \times CA$) to become a group, the results of itinerary recommended for them are similar. Thus, there is no need for itinerary recommended to perform for each individual traveler. Instead of trying to derive the *exact* travel schedule which may incur large computation overhead, we send their queries as a batch to drive only one itinerary for them.

Example:

Suppose $\{u_1, u_2, u_3, u_4, u_5\}$ are tagged with the same level-1 group label by model in Step 1 as $g.l_1$ and Table 2 shows their similarity. First, we mark all users not used in g. u_1, u_2, u_3, u_4, u_5 are sorted by their start and end time. Suppose $CA = 2$, $OR = 1$, a car can carry only 2 people. We choose u_1 who has the earliest start time as u' in line 4. Then we maintain a maximum heap with size of $CA - 1$(except u') which store the users most similar to u_1. After we traverse all users *fesiable* for u_1 We find u_1 and u_2 are most suitable to travel together, so we group them and mark them as used in line 13. Next step, u_3 is the earliest and not used u', then we group $\{u_3, u_4\}$ together. Finally, u_5 needs to travel alone.

We we set $CA = 3$, $OR = 1$, $\{u_1, u_2, u_3\}$ will in a group. Although u_4, u_5 satisfies $SG(u_4, u_5)$, it's more economical for them to travel alone rather than to share a car together. However, u_4, u_5 have similar category and time schedule, we will invoke our itinerary recommend service(Step 3) only once and return them with the same itinerary recommendation which avoids unnecessary computations and offers correctness at the same time.

B. GROUP EVALUATION

We use average time utilization, car utilization, and interest similarity of all groups to evaluation car group allocation.

1) TIME UTILIZATION

The average time utilization of all users in a group g:

$$Tu(g) = \frac{1}{|g|} \sum_{u \in g} \frac{g.t_e - g.t_s}{u.t_e - u.t_s} \tag{16}$$

The average time utilization of all groups $\mathcal{G}$:

$$Tu(\mathcal{G}) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \frac{1}{|g|} \sum_{u \in g} \frac{g.t_e - g.t_s}{u.t_e - u.t_s} \tag{17}$$

2) CAR UTILIZATION

The average car utilization of all users in a group g:

$$Cu(g) = \begin{cases} \frac{|g|}{CA}, & \text{if } \frac{|g|}{CA} >= OR \\ 0, & otherwise \end{cases} \tag{18}$$

The car utilization of all groups $\mathcal{G}$:

$$Cu(\mathcal{G}) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} Cu(g) \tag{19}$$

3) USER INTEREST SIMILARITY

A group average cosine similarity is the average interest cosine similarity of all pair-wise combinations of users in group g:

$$Cos(g) = \frac{1}{|g|} \sum_{u_i \in g} \sum_{u_j \in g, u_j \neq u_i} \frac{\vec{I_{u_i}} \cdot \vec{I_{u_j}}}{\|\vec{I_{u_i}}\| \cdot \|\vec{I_{u_j}}\|} \tag{20}$$

The average cosine similarity of all groups $\mathcal{G}$:

$$Cos(\mathcal{G}) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \frac{1}{|g|} \sum_{u_i \in g} \sum_{u_j \in g, u_j \neq u_i} \frac{\vec{I_{u_i}} \cdot \vec{I_{u_j}}}{\|\vec{I_{u_i}}\| \cdot \|\vec{I_{u_j}}\|} \tag{21}$$

V. STEP 3 ITINERARY RECOMMENDATION WITH POI CROWDEDNESS

After group allocation, an optimal itinerary will be recommended for each group. In this section, we first describe the requirements for an itinerary, after that, three methods are proposed to recommend itineraries.

For each group g with category preference $g.I$, start time $g.t_s$, end time $g.t_e$ and start point $g.v_s$, we recommend a itinerary $R_g = \{v_1, v_2, \ldots, v_n\}$ such that the following object function is maximized:

$$Score(R_g) = \sum_{v \in R_g} pr_g(v) \tag{22}$$

where the profit $pr_g(v, t)$ means that group g visit POI v at time t, and v belongs to category c which is defined as:

$$pr_g(v, t) = \frac{(\rho \cdot Pop(v) + (1 - \rho) \cdot I_g(c))^{\gamma}}{Crd(v, t)} \tag{23}$$

And the itinerary constraints are as following:

Rule 1: Each itinerary can visit same POI only once which avoids blindly searching back and forth among POIs or rounding in cycle which is only time consuming with no profit collected.

Rule 2: Group time budget. The itinerary total time cost $Cost(R)$ associated with a total stay time on each POI and a total transit time

$$Cost(R) = Dist(v_s, v_1) + Dist(v_n, v_s) + \sum_{v_i \in R} (Dur(v_i) + Dist(v_i, v_{i+1})) \tag{24}$$

should not exceed $g.t_e - g.t_s$. We use $Sat(v_i, v_j, t)$ returns *true* if leaving from v_i at time t to visit v_j has enough time to reach the destination v_s within the budget time.

A. METHOD 1 BACKTRACK WITH PRUNING

Due to the Irregular profit changing with time, a brute-force approach to solving itinerary recommendation is to do an exhaustive search: we enumerate all candidate Itineraries from the source v_s at start time t_s. Instead of searching in a breadth-first manner which imposes a bottleneck on the memory requirement, we search itineraries in a depth-first manner, so that only the current branch at any level is searched at any time.

Suppose that v_i is the last POI of current partial itinerary, which satisfies the group time budget. We need to deeper search for its connected v_j and extend current partial itinerary by adding v_j after visiting v_i, by the way, add v_j into *visited*. Finally, if there is no POI could add to current partial itinerary after v_j, due to the constraints, we will remove v_j from *visited*, remove v_j from current partial itinerary and backtrack to its pre-visited POI v_i. Then we extend the partial itinerary again without *visited* POI like v_j. During the search process we update the best itinerary with the highest profit score as our final itinerary recommended to the group.

Algorithm 2 Backtrack With Pruning

Input: POI map, group query g, profit function pr_g
Output: optimal itinerary R

```
R ← ∅, r' ← ⟨v_s → ∅ → v_s⟩, visited ← ∅
for v_j ∈ V \ visited do                          // Rule 1
    if Sat(v_i, v_j, t) then                      // Rule 2
        r' ← r' ∪ v_j;
        visited ← visited ∪ v_j;
        if Score(r') > Score(R) then
            R ← r';
        end if
        backtrack(r', visited);
        r' ← r' \ v_j;
        visited ← visited \ v_j;
    end if
end for
return R
```

However, the backtrack search is computationally prohibitive. The time complexity is $O(|V|^{\lfloor \frac{\Delta}{Dist_{min}+Dur_{min}} \rfloor})$, where Δ is a specified time budget, the route length is at most $\lfloor \frac{\Delta}{Dist_{min}+Dur_{min}} \rfloor$, where $Dist_{min}$ is the smallest distance

between two POIs and Dur_{min} is the smallest duration time of all the POIs.

Example: There are POI 1, 2, 3 in the map and a group start at v_s. We use R to store the best itinerary. Initially, current itinerary r' is $\langle v_s \to \emptyset \to v_s \rangle$ and POI 1, 2, 3 is not visited. We add POI to r' until all POI are visited or time runs out, so we get $r' = \langle v_s \to 1 \to 2 \to 3 \to v_s \rangle$. Every time we add one POI to extend our current itinerary, we compare it with R and replace R with r' if current $Socre(r')$ is higher than $Score(R)$. Then we remove 3 from r' and backtrack to $\langle v_s \to 1 \to 2 \to v_s \rangle$ level, but 3 are tried and no other POI could try, so we remove 2 from r' further and backtrack to $\langle v_s \to 1 \to v_s \rangle$ level. Then 2 is tried in this level so we extend r' to $\langle v_s \to 1 \to 3 \to 2 \to v_s \rangle$. Finally, we tried all possible POI's permutation within time budget and recoded the best one as our result.

B. METHOD 2 GREEDY

To avoid the expensive Backtrack with Pruning search, the greedy algorithm iteratively selects next POI v_j added to current partial itinerary until no other POIs can be added to the itinerary. We choose the next POI whose strategy function $f(v_j) = \frac{pr(v_j,t)}{Dist(v_i,v_j)+Dur(v_j)+Dist(v_j,v_s)}$ is maximal and no constraints are violated, where v_i is the previously visited POI, t is the arrive timeof POI v_j.

Algorithm 3 Greedy

Input: POI map, group query g, strategy function $f(v)$
Output: optimal itinerary R

```
R ← ∅, r' ← ⟨v_s → ∅ → v_s⟩, visited ← ∅
while next ≠ ∅ do
    maxP ← 0, next ← ∅
    for v_j ∈ V \ visited do                    // Rule 1
        if Sat(v_i, v_j, t) and f(v_j) > maxP then    // Rule 2
            maxP ← f(v_j), next ← v_j;
        end if
    end for
    visited ← visited ∪ v_j, R ← R ∪ next
end while
return R
```

Example: There are POI 1, 2, 3 in the map and a group start at s. POI 1 has the highest profit from v_s, so the partial itinerary is $\langle v_s \to 1 \to v_s \rangle$ and we choose next highest profit POI from POI 1 until running out of time budget.

C. METHOD 3 BREADTH-FIRST-SEARCH WITH GREEDY PRUNING

To avoid the expensive backtrack with pruning search, we present a approximation algorithm BFS-GP which sacrifice optimality for efficiency. We use greedy pruning in breadth first search to reduce the cost of enumerating all possible itineraries.

The basic idea is to consider extending each generated partial itinerary II. Each partial itinerary II is labeled by $II = (A, T, S, E, r, er, v_i)$, where A is the set of POIs already visited, T is the departure time from v_i, S is the overall profit collected(i.e., $Score(r)$) from current partial itinerary, E is a *Expect* score which is the lower bound maximum total profit collected by the fully extended itinerary which is estimated by greedy strategy, r is the current partial itinerary $\langle v_s \to \cdots \to v_i \rangle$(without the sub-itinerary $v_i \to v_s$), er is the full itinerary within time budget corresponding to E and v_i is the ending POI. Initially, there is only one feasible partial itinerary $II_0 = (\emptyset, g.t_s, 0, E_0, v_s, er_0, v_s)$, representing the empty itinerary $s \to s$.

Expect Score: Given a partial itinerary $r' = \langle v_s, \cdots, v_i \rangle$, its expect score E is defined as fllows:

$$Expect(r') = Score(r') + extra(r') \tag{25}$$

For efficiency reasons, we need to find a approximation for $extra(r')$ as big as possible without underestimating. We use greedy strategy complete the partial itinerary from v_i by keeping selecting the next best v_j such that its $\frac{pr(v_j,II'.T)}{Dist(v_i,v_j)+Dur(v_j)+Dist(v_j,v_s)}$ is maximal and no constraints are violated until no other POIs are can be added to the itinerary.

At the k th iteration($k > 0$), the BFS-GP algorithm extends each II of size $k-1$ to a new partial itinerary of size k by add a new POI. Specifically, a partial itinerary $II = (A, T, S, E, r, er, v_i)$ is extended into a new partial itinerary II' associated with POI $j \notin II.A$ according to the follow rules:

$$\begin{cases} II'.A = II.A \cup \{v_j\} \\ II'.T = II.T + Dist(v_i, v_j) + Dur(v_j) \\ II'.S = II.S + pr(v_j, II.T + Dist(v_i, v_j)) \\ II'.r = II.r \to v_j \\ II'.E, II'.er = Expect(II'.r) \end{cases} \tag{26}$$

A new partial itinerary II' continues to extend if $Sat(v_i, v_j, II.T)$ returns *true* and $II'.E > II.E$. Intuitively, this means that the current partial itinerary r of the II can be extended to v_j and then finished at the destination v_s within the time budget. Besides, add v_j to current partial itinerary will not decrease the expect total profit E as otherwise we would rather follow the greedy strategy.

We organize those partial itineraries by a queue Q. In each step, we select one partial itinerary II_i from the queue. Then extend it to generate more candidate partial itinerary by adding a new POI at the end of the itinerary in $II_i.r$. Those new extended itineraries whose time cost is smaller than the time budget and *Expect* score is higher than the *Expect* score before extended are enqueued. In this way, the more promising itineraries are kept and at the same time guarantee efficient computing. We terminate the algorithm when Q is empty.

Example: Consider the example graph in Fig. 3. There are POI 1, 2, 3 and they are belong to category $c1, c2, c3$ respectively. A group's query is $(g.I, 9:00, 18:00, v_s)$, where the group is start from v_s at 9:00 am and $g.I$ is $\langle c1 = 2, c2 = 1, c3 = 3 \rangle$. And Fig. 3.e shows the crowdedness of POI 1, 2, 3 in a day from 9:00 am to 18:00 pm,

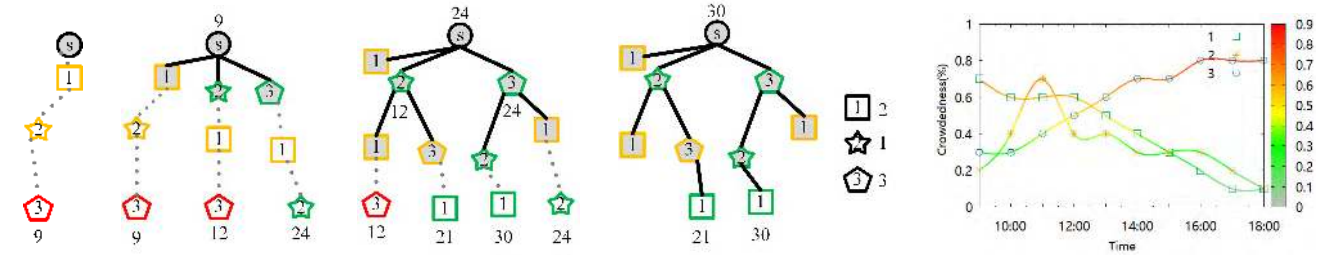

FIGURE 3. Steps of BFS-GP.

Algorithm 4 BFS-GP

Input: POI map G, group profit function pr, group query g
Output: Itinerary, R

1: Queue Q, $R \leftarrow \emptyset$
2: $E_0, er_0 \leftarrow greedy(s, g.t_s)$,
3: $II_0 \leftarrow (\emptyset, g.t_s, 0, E_0, s, er_0, s)$
4: $Q.enqueue(II_0)$
5: **while** Q is not empty **do**
6: $II \leftarrow Q.dequeue()$;
7: **for** $v_j \in V \setminus II.A$ **do**
8: **if** $Sat(v_i, v_j, II.T)$ **then**
9: $II'.r \leftarrow II.r \rightarrow v_j$
10: $II'.T \leftarrow II.T + Dist(v_i, v_j) + Dur(v_j)$
11: $II'.E, partR'.er \leftarrow Expect(II'.r)$
12: $II' \leftarrow (II.A \cup \{v_j\}, II'.T, II.S + pr(v_j, II.T + Dist(v_i, v_j)), II'.E, II'.r, II.er, v_j)$;
13: **if** $II'.E > Score(R)$ **then**
14: $R \leftarrow II'.er$;
15: **else if** $II'.E > II_i.E$ **then**
16: $Q.enqueue(II')$
17: **end if**
18: **end if**
19: **end for**
20: **end while**
21: **return** $R \rightarrow v_s$

for simplicity we assume there are 3 crowdedness types: red:1, yellow:0.5, green:0.2. Suppose 1 is the nearest POI from v_s and 3 is the farthest. For each POI i, the touring time $Dur(i)$ is set to 30 minutes. We compute $E_0 = 9$ and $er_0 = \langle v_s \rightarrow 1 \rightarrow 2 \rightarrow 3 \rangle$ using greedy strategy in (25). The initial II_0 is $(\emptyset, 9 : 00, 0, 9, \langle v_s \rangle, \langle v_s \rightarrow 1 \rightarrow 2 \rightarrow 3 \rangle, v_s)$ and enqueue it to Q then dequeue it as current partial itinerary II in line 6. First, we add 1 to II and get $II' = (\{1\}, 11 : 30, 4, 9, \langle v_s \rightarrow 1 \rangle, \langle v_s \rightarrow 1 \rightarrow 2 \rightarrow 3 \rangle, 1)$, $II'.E = 9$ is not more than $II.E$, so we will not enqueue it and prune it. Then, we add 2 and get $II' = (\{2\}, 12 : 00, 5, 12, \langle v_s \rightarrow 2 \rangle, \langle v_s \rightarrow 2 \rightarrow 1 \rightarrow 3 \rangle, 2)$ which is better than II, so we enqueue it and wait for extending it at next time. Adding 3 is also promising and should be enqueued. Every time we generate a new partial itinerary II', we compare it with our recorded best itinerary R and replace R with a better one, if current complete itinerary $II'.er$ has a better score $II'.E$ than R. Now the best result is 24 and the best itinerary R is $\langle v_s \rightarrow 3 \rightarrow 1 \rightarrow 2 \rangle$. In the next level as Fig. 3(c), we dequeue $II' = (\{2\}, 12 : 00, 5, 12, \langle v_s \rightarrow 2 \rangle, \langle v_s \rightarrow 2 \rightarrow 1 \rightarrow 3 \rangle, 2)$ from Q and we extend it with 1 and 2 and get the new partial itinerary $\langle v_s \rightarrow 2 \rightarrow 3 \rangle$ with expect score $II'.E = 21$ are enqueued, by the way, we update our best result R. And expend $II.r = \langle v_s \rightarrow 3 \rangle$ using POI 2 and 1, and $\langle v_s \rightarrow 3 \rightarrow 2 \rangle$ has expect score 30. Now current best itinerary R is $\langle v_s \rightarrow 3 \rightarrow 2 \rightarrow 1 \rangle$ with score as 30. Keeping dequeue partial itineraries from Q and extends them until Q is empty. Finally, we get our best itinerary in R which is $\langle v_s \rightarrow 3 \rightarrow 2 \rightarrow 1 \rangle$ with score 30.

However, due to the greedy expect score, BFS-GP loses the optimality in some cases. For example, suppose the partial itinerary $r = \langle v_s \rightarrow 1 \rightarrow 3 \rightarrow 2 \rangle$, $Score(r)$ is 30. But when we calculate the *Expect* score of partial itinerary $r' = \langle v_s \rightarrow 1 \rangle$, because POI 2 is nearer than 3 to 1, we extend r' with $\langle r' \rightarrow 2 \rightarrow 3 \rangle$ and we compute the *Expect* score is 9 which is not larger than the *Expect* score we calculate from partial itinerary v_s. While the route $\langle v_s \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow v_s \rangle$ could be the optimal solution, but it cannot be generate because $r' = \langle v_s \rightarrow 1 \rangle$ was pruned.

VI. EXPERIMENTAL STUDY

A. EXPERIMENTAL SETTINGS

We evaluate the algorithms above on datasets extracted from Flickr photos [10] in four cities, namely, Edinburgh, Melbourne, Osaka and Toronto, with statistics shown in Table 3. All datasets are provided by Lim *et al.* [3], [4], [11].

Our algorithms are implemented in Java and Python sklearn package. All the experiments were run on a PC with 3.6 GHz Intel i7 Quad-Core and 16 GB of RAM.

B. EFFECT OF STEP 1.OFFLINE CITY MODEL TRAIN

Based on the entire set of users' photos as a good approximation for real-life tourist visits for each city, we first construct the interest preference vector I for all users $u \in U$, as stated in (5). We perform offline model in Step 1, Table 4. shows the result of Step 1 for each dataset. The best k represents the goal in (9) is maximum when we use GMM model find the

TABLE 3. A summary of the datasets.

City	No. of POIs	No. of Categories	No. of Photos	No. of Users	#POI visits	#Travel Sequences
Toronto	30	6	157,505	1,395	39,419	6,057
Edinburgh	29	6	82,060	1,454	33,944	5,028
Osaka	29	4	392,420	450	7,747	1,115
Melbourne	242	16	94,142	911	17,087	3975

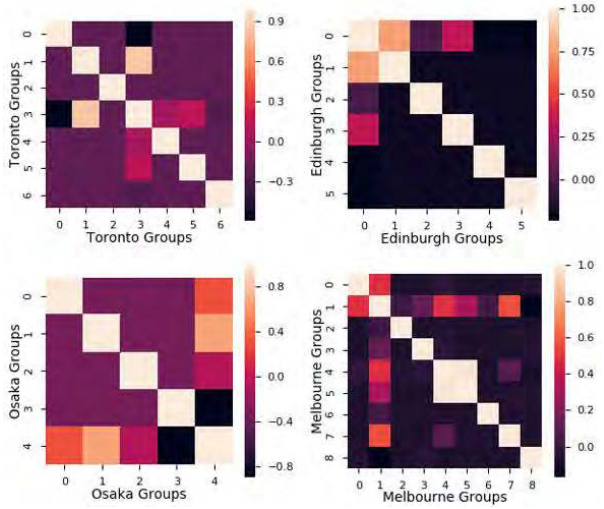

FIGURE 4. The Inter similarity of k Groups in Setp 1.

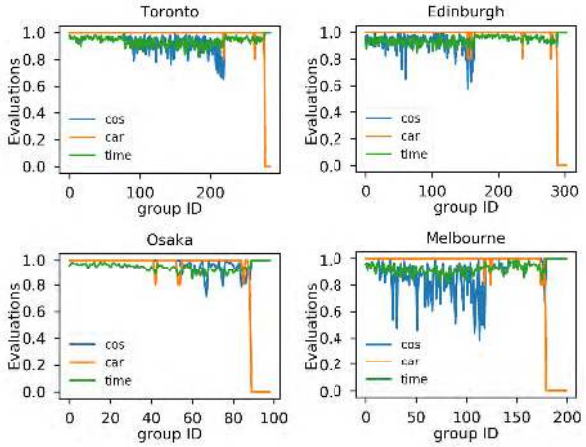

FIGURE 5. Evalution results of all groups in Setp 2.

TABLE 4. Evaluation of step 1.

	Toronto	Edinburgh	Osaka	Melbourne
best k	7	6	5	9
$Intra(P)$	0.9172	0.9179	0.9109	0.9173
$Inter(P)$	0.1101	0.1357	0.1809	0.1225
CT	0.4203	0.5077	0.5547	0.3704

best clusters partition by history data. The $Inter(P)$ line shows the average inter-cluster similarity of k patterns' pair-wise combinations. If some clusters' inter cosine similarity score $Inter(p_i, p_j)$ is high, those clusters will be merged and redundant. The intra-cluster cosine similarity $Intra(pt_k)$ shows how similar users in those k clusters respectively are in therms of their interest which also can help us to determine what cosine threshold CT we should set in Step 2. In this experimental we set CT using the lowest cosine similarity among k clusters by GMM model. As shown in the table, the average intra-cluster similarity $Intra(P)$ of all clusters are very high, suggesting good cluster compactness.

The heat map in Fig. 4 shows the correlation coefficients between each pattern, the deeper the color the more different between two patterns and the light color means the two patterns should be merged into one cluster. As shown in Fig. 4, the gaussian mixture model (GMM) in Step 1 achieves great intra-cluster similarity, clear distinguish patterns and less redundancy on all datasets.

In this way, we get most k representative travel pattern in a city and in the rest of the experiments, users are classified to distinguish level-1 groups with label $u.l$.

C. EFFECT OF STEP 2.CAR GROUP ALLOCATION

Due to users' check-in times in datasets are too scattered and their travel time during a day are usually too short which is not suitable for our problem. All users' start time $u.t_s$ and end time $u.t_e$ were generated randomly from 9:00 to 11:00 and 16:00 to 18:00, so the time budget for groups could range from 5 hours to 9 hours. Car Capacity Ca is set to 5 users, and occupancy rate OR is set to 0.8, and the longest time difference threshold DT is set to 60 minutes, the lowest time similarity TT is set to 0.75, mu is set to 0.5, ω is set to 0.5. From previous step, part user into k distinct groups and set the lowest $Intra(pt_k)$ among all k patterns as cosine threshold CT, we compare each user with those k patterns and label them by $u.l$ and part them into k distinct level-1 groups.

Fig. 5 shows the evaluation results stated in Section IV-B of each small car groups in each dataset after Step 2.

Table 5. shows the average evaluation results on each dataset. Although our Car Group Allocation algorithm is not a optimal solution, as show in Table 5., the average time utilization $Tu(\mathcal{G})$ is over 90% on each datasets, which means users travel in a same car or travel alone have more than 90% time overlap, which means for every users, the budget time wait for someone else are at most 60 minutes within DT as we have set. And the average cosine similarity $Cos(\mathcal{G})$ is much higher than and the lowest cosine CT we expect in Setp 1.

TABLE 5. Average evalution results of Step 2.

	Toronto	Edinburgh	Osaka	Melbourne
$Tu(\mathcal{G})$	0.9372	0.9439	0.9436	0.9270
$Cu(\mathcal{G})$	0.9699	0.9505	0.8889	0.89
$Cos(\mathcal{G})$	0.9534	0.9590	0.9727	0.8997
single person	8/1395	14/1454	10/450	21/911
runtime(s)	9.6000	12.4001	1.0024	4.8957

$Cu(\mathcal{G})$ and single person shows our method groups most of people. And the runtime is a single-threaded execution time which is also acceptable.

After we use the equationPs in Section IV to organize all users' queries by groups' query set $\mathcal{G}$, and query their recommended itineraries in Step 3.

D. EFFECT OF STEP 3.ITINERARY RECOMMEND

In this section, we evaluate the itinerary search methods in Section V.

Based on Definition (1), we use the time taken of a user's first and last photo at a POI to determine his/her visit duration and determine the average visit duration of every POI in four cities respectively. For some cold POIs whose duration time can not be computed, we use the average of all POIs' duration time in the city to represent. The crowdedness of each POI is calculated by (3). And we assume the start point(hotel s) is the most popular POI in a city. The distance $Dist(v_i.v_j)$ between two POIs were estimated using their Euclidean distance. If the distance is smaller than 1 km, we travel on foot and set a walking speed of 4 km/hour, which is from the literature [11], otherwise, we travel by car, the time cost estimated using Google Maps.[1]

Fig. 6 (left column) presents the score of the itinerary recommended by backtrack with pruning, greedy and BFS-GP method. The first three use y-axis being the ratio of profit score of itineraries found by BFS-GP and greedy to optimal profit score found by backtrack with pruning over all groups and x-axis being the time budget(hour) b of each group. Note that the optimal backtrack method can't perform on Melbourne dataset due to the large POI numbers, so we don't have a optimal result to evaluate the BFS-GP and greedy. For the first three datasets, on the average BFS-GP performs 98% of optimal solution. Greedy performs about 10% worse than optimal Backtrack, as its search strategy is rather simple and about 8% worse than BFS-GP.

In rare cases, BFS-GP contains too many possible partial itineraries, so we can use a Priority Queue with fixed capacity $|Q|$ replace the Queue in BFS-GP algorithm line 1 which adjusts the size of queue by removing the IIs with the lowest scores and keep the top $|Q|$ most promising IIs. The queue size should be set according to the real-word application. In this experiment, Q size is set to 100. The last one in Fig. 6 (left column) is the ratio(y-axis) of the score of itineraries found by BFS-GP using limited size priority queue to the score of itineraries found by BFS-GP vs each groups' time budget(x-axis);

[1]https://www.google.com/maps

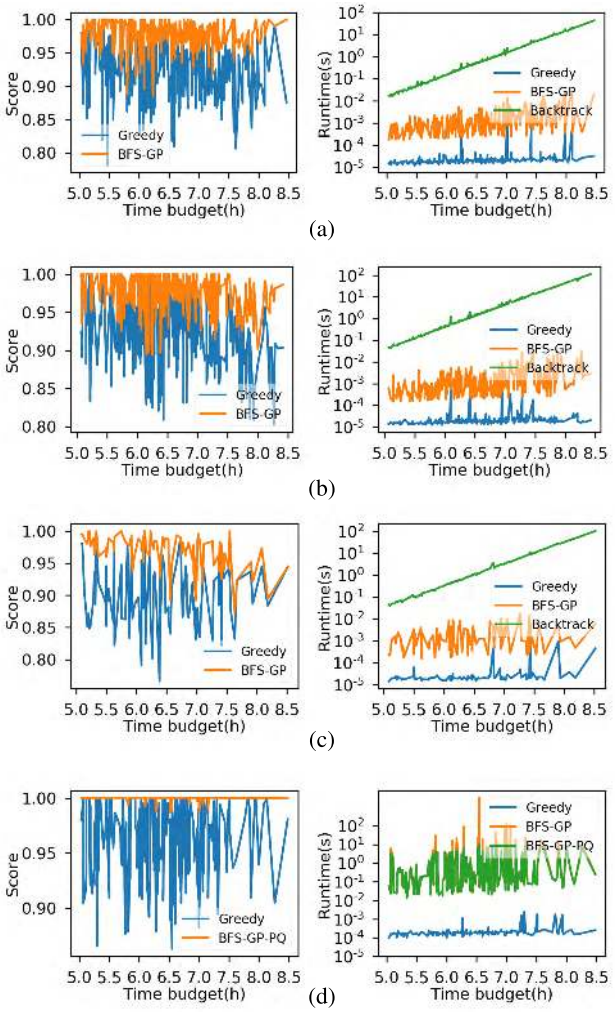

FIGURE 6. Effect of step 3: (left) (a)(b)(c) are the ratio(y-axis) of the score of itineraries found by BFS-GP to the score of itineraries found by Backtrack with pruning vs each groups' time budget(x-axis), and (d) is the ratio(y-axis) of the score of itineraries found by BFS-GP using limited size priority queue to the score of itineraries found by BFS-GP vs each groups' time budget(x-axis); (right) the runtime(y-axis) vs each groups' time budget(x-axis).

Fig. 6(right column) presents the runtime of those three algorithms for per group, with y-axis being the runtime(seconds) and x-axis being the time budget of a trip b(hours). Greedy have a fast and stable runtime because Greedy only maintain one route, but this feature also overlooks other possible combinations of POIs, thus hardly finding optimal solutions. When the feasible POI number is larger than 30, backtrack with pruning algorithm runs over 5 minutes. But Backtrack is still the best way to find the optimal itinerary when the travel city has a small number of POI or to find a best way inside a big POI. In Fig. 6 (d), BFS-GP using priority queue optimize the efficiency of BFS-GP at some time and find the result same as BFS-GP at the most of time.

In summary, Backtrack with pruning finds the optimal solution; BFS-GP is a very good trade-off for efficiency at a slightly lower itinerary profit score than the optimal solution.

VII. RELATED WORK

A. GROUP-WISE ITINERARY PLANNING

There exist some researches studying the group-wise itinerary planning problem [1]–[6]. However, none of them consider the constraints as complete as this paper does. In this paper, to improve the quality of suggested itineraries, we consider the deadline and preference of each user, the popularity and crowdedness (e.g. waiting time) of each point of interest (POI). Moreover, for each generated group, we limit its size according to general capacity of cars, that is 4.

B. ROUTE RECOMMENDATION

The route recommendation problem is related to the orienteering problem [12] that is a variant of the selective traveling salesman problem [13], where (i) not all requests need to be completed, and (ii) the cost is the sum of the total travel time and the penalty of rejected requests. The orienteering problem is well studied, but only several works [14]–[16] consider the Orienteering Problems with each request having a Time Window (OPTW). The route recommendation service [17], [18] is also provided in other forms including trip planning [19], optimal sequenced routing [20] and path searching [21]. They require finding the shortest route that passes through specific types of points-of-interests.

C. SPATIAL QUERIES

Spatial queries (e.g., range queries [22], [23], nearest neighbor queries [23], [24] and top-k spatial keyword queries [25], [26]) has been extensively studied in recent literatures, most of which employ the technique of safe region to reduce the client-server communication cost significantly. It is guaranteed that the query result remains unchanged as long as the user is within the safe region. A new result and an updated safe region are requested from the server if the user leaves the safe region.

D. MOBILE INTERNET OF THINGS

References [27] and [28] designed protocols to solve congestion and save energy in mobile IoT. Reference [29] use cache policy to reduce latency and computation on edge service. Reference [30] proposes a verification platform to compute the service reliability and cost in the design phase. Reference [31] studies the approach to build service compositions in sharing community. Reference [32] proposes method of performing a service reconfiguration. And [33] provides method to guarantee data consistency of the dynamic replacement for service Process.

VIII. CONCLUSION

In this paper, we propose a group-wise itinerary planning framework to improve the user experiences in temporal mobile social network. In the studied problem, users registered for the same business/research conference are treated to have common connections and thus may be willing to travel together in the conference city. Our method generates temporal mobile groups of guests according to their interest preferences and available times. For each group, an itinerary is recommended to achieve maximum user satisfactory. Experimental results depict the effectiveness of our methods.

REFERENCES

[1] C. Zhang, H. Liang, K. Wang, and J. Sun, "Personalized trip recommendation with POI availability and uncertain traveling time," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Melbourne, VIC, Australia, Oct. 2015, pp. 911–920.
[2] L. Fan, L. Bonomi, C. Shahabi, and L. Xiong, "Optimal group route query: Finding itinerary for group of users in spatial databases," *GeoInformatica*, vol. 22, no. 4, pp. 845–867, 2018.
[3] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Towards next generation touring: Personalized group tours," in *Proc. 26th Int. Conf. Automated Planning Scheduling (ICAPS)*, London, U.K., Jun. 2016, pp. 412–420.
[4] X. Wang, C. Leckie, J. Chan, K. H. Lim, and T. Vaithianathan, "Improving personalized trip recommendation by avoiding crowds," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Indianapolis, IN, USA, Oct. 2016, pp. 25–34.
[5] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu, "TripPlanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1259–1273, Jun. 2015.
[6] K. H. Lim, J. Chan, S. Karunasekera, and C. Leckie, "Personalized itinerary recommendation with queuing time awareness," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Tokyo, Japan, Aug. 2017, pp. 325–334.
[7] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.
[8] C. M. Bishop, "Pattern recognition and machine learning," *J. Electron. Imag.*, vol. 16, no. 4, pp. 430–455, 2007.
[9] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: An overview from a database perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 6, pp. 866–883, Dec. 1996.
[10] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "YFCC100M: The new data in multimedia research," *CoRR*, vol. 1, no. 8, Mar. 2015.
[11] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized tour recommendation based on user interests and points of interest visit durations," in *Proc. 24th Int. Joint Conf. Artif. Intell. (IJCAI)*, Buenos Aires, Argentina, Jul. 2015, pp. 1778–1784.
[12] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *Eur. J. Oper. Res.*, vol. 209, no. 1, pp. 1–10, Feb. 2011.
[13] H. A. Eiselt, M. Gendreau, and G. Laporte, "Location of facilities on a network subject to a single-edge failure," *Networks*, vol. 22, no. 3, pp. 231–246, May 1992.
[14] G. N. Frederickson and B. Wittman, "Approximation algorithms for the traveling repairman and speeding deliveryman problems," *Algorithmica*, vol. 62, nos. 3–4, pp. 1198–1221, 2012.
[15] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, "Approximation algorithms for deadline-TSP and vehicle routing with time-windows," in *Proc. Symp. Theory Comput.*, 2004, pp. 166–174.
[16] G. Righini and M. Salani, "Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming," *Comput. Oper. Res.*, vol. 36, no. 4, pp. 1191–1203, Apr. 2009.
[17] M. U. S. Khan, O. Khalid, Y. Huang, R. Ranjan, F. Zhang, J. Cao, B. Veeravalli, S. U. Khan, K. Li, and A. Y. Zomaya, "MacroServ: A route recommendation service for large-scale evacuations," *IEEE Trans. Services Comput.*, vol. 10, no. 4, pp. 589–602, Jul./Aug. 2017.
[18] Y. Li, M. L. Yiu, and W. Xu, "Oriented online route recommendation for spatial crowdsourcing task workers," in *Proc. SSTD*, 2015, pp. 137–156.
[19] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in *Proc. SSTD*, 2005, pp. 273–290.

[20] M. Sharifzadeh, M. R. Kolahdouzan, and C. Shahabi, "The optimal sequenced route query," *VLDB J.*, vol. 17, no. 4, pp. 765–787, 2008.
[21] W. Li, J. Guan, X. Lian, S. Zhou, and J. Cao, "Probabilistic time-constrained paths search over uncertain road networks," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 399–414, Mar./Apr. 2018.
[22] M. A. Cheema, L. Brankovic, X. Lin, W. Zhang, and W. Wang, "Multi-guarded safe zone: An effective technique to monitor moving circular range queries," in *Proc. IEEE ICDE*, Mar. 2010, pp. 189–200.
[23] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Location-based spatial queries," in *Proc. ACM SIGMOD*, 2003, pp. 443–454.
[24] S. Nutanong, R. Zhang, E. Tanin, and L. Kulik, "The V*-diagram: A query-dependent approach to moving KNN queries," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 1095–1106, 2008.
[25] D. Wu, M. L. Yiu, and C. S. Jensen, "Moving spatial keyword queries: Formulation, methods, and analysis," *ACM Trans. Database Syst.*, vol. 38, no. 1, pp. 7:1–7:47, Apr. 2013.
[26] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial Web objects," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 337–348, Aug. 2009.
[27] X. Zeng, G. Xu, X. Zheng, Y. Xiang, and W. Zhou, "E-AUA: An efficient anonymous user authentication protocol for mobile IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1506–1519, Apr. 2018.
[28] G. Q. Xu, Y. Zhang, A. K. Sangaiah, X. H. Li, A. Castiglione, and X. Zheng, "CSP-E2: An abuse-free contract signing protocol with low-storage TTP for energy-efficient electronic transaction ecosystems," *Inf. Sci.*, vol. 476, pp. 505–515, Feb. 2019.
[29] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.
[30] H. Gao, H. Miao, L. Liu, J. Kai, and K. Zhao, "Automated quantitative verification for service-based system design: A visualization transform tool perspective," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 10, pp. 1369–1397, 2018.
[31] S. Deng, L. Huang, J. Taheri, J. Yin, M. Zhou, and A. Y. Zomaya, "Mobility-aware service composition in mobile communities," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 555–568, Mar. 2017.
[32] H. Gao, W. Huang, X. Yang, Y. Duan, and Y. Yin, "Toward service selection for workflow reconfiguration: An interface-based computing solution," *Future Gener. Comput. Syst.*, vol. 87, pp. 298–311, Oct. 2018.
[33] H. Gao, Y. Duan, H. Miao, and Y. Yin, "An approach to data consistency checking for the dynamic replacement of service process," *IEEE Access*, vol. 5, pp. 11700–11711, 2017.