Personalized Trip Recommendation with POI Availability and Uncertain Traveling Time

Chenyi Zhang[#*], Hongwei Liang[*], Ke Wang[*], Jianling Sun[#]
[#] College of Computer Science, Zhejiang University
[*] School of Computing Science, Simon Fraser University
{chenyiz,hongweil,wangk}@sfu.ca, sunjl@zju.edu.cn

ABSTRACT

As location-based social network (LBSN) services become increasingly popular, trip recommendation that recommends a sequence of points of interest (POIs) to visit for a user emerges as one of many important applications of LBSNs. *Personalized* trip recommendation tailors to users' specific tastes by learning from past check-in behaviors of users and their peers. Finding the optimal trip that maximizes user's experiences for a given time budget constraint is an NP hard problem and previous solutions do not consider two practical and important constraints. One constraint is *POI availability* where a POI may be only available during a certain time window. Another constraint is *uncertain traveling time* where the traveling time between two POIs is uncertain. This work presents efficient solutions to personalized trip recommendation by incorporating these constraints to prune the search space. We evaluated the efficiency and effectiveness of our solutions on real life LBSN data sets.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; I.2.8 [**Problem Solving, Control Methods, and Search**]: Scheduling

Keywords

Trip plan; location-based social network; recommender systems

1. INTRODUCTION

With the emerging development of location-based social network (LBSN) services such as Yelp and Foursquare, users are able to "check in" at a certain point of interest (POI), such as restaurant/museum/park, via their mobile devices. A user may rate and make comments after visiting a POI and other users may consider those ratings and comments to select the POIs for their visits at a later time. The availability of such rating data and LBSN services open up an array of new research problems in both academia and industry, such as user behavior analysis, movement pattern study [5, 15], and various real-world applications [6, 32, 34]. Among them, POI recommendation and trip recommendation [14, 28] are hot topics and require a location sensitive solution. For example, recommending a highest rated Chinese restaurant in Beijing to a user who is currently visiting New York City will fail, even if the user loves Chinese food. Recommending a nearby Chinese restaurant with a reasonable rating score makes more sense in this case.

In this paper, we focus on the personalized trip recommendation problem. In this problem, a user travels to a new region (e.g., on a business trip to a new city) and wants to visit several POIs within a limited amount of time. The goal is to recommend a trip route visiting several POIs according to not only the temporal-spatial constraints (more details shortly), but also the user specific preferences on POIs.

1.1 Motivation

The trip recommendation is not trivial because of the following challenges:

• (*Personalization*) First, while a user has its own interests, explicitly soliciting this information does not work in large scale applications because the user often does not know what POIs are available and where they are. Modeling user preferences by learning from historical rating and check-in behaviors of users and their peers to predict the user's preferences on unvisited POIs would be a preferred solution.

• (*Sequence of POIs*) Second, the traditional POI recommendation recommends individual POIs with highest scores, such POIs may not form a feasible trip due to the spatial and time constraints.

• (*POI availability and uncertain traveling time*) Third, the traditional trip recommendation assumes that POIs are always available any time and the traveling time between two POIs are known in advance, but in practice, a POI may be available only at certain times (say, due to opening hours and closing hours) and traveling time is uncertain due to traffic conditions at the time of travel. As a result, whether a POI can be visited will depend on its available time and predictability of the time traveling to the POI. If the timeliness of finishing the trip is important to the user, a trip with a more predictable traveling time would be preferred. For example, the user may give up one more POI to visit in order to ensure a high probability of visiting another more preferred POI or arriving at the specified destination on time.

• (*Large search space*) Finally, the POI availability and uncertain traveling time imply each order of visiting a set of POIs may have a different consequence, thus, a brute-force search of all candidate trips is prohibitive. For example, with 150 POIs in total, the number of trips that consist of 5 POIs can reach billions (i.e., 150!). Most of these candidate trips do not follow the POI availability or match user's preferences, or cannot be finished within a given time limited. A strategy that prunes such infeasible and non-optimal trips based on user preferences, POI availability, traveling time uncertainty is essential for scaling a solution to large applications.

CIKM'15, October 19–23, 2015, Melbourne, VIC, Australia

ACM 978-1-4503-3794-6/15/10 ...$15.00
DOI: http://dx.doi.org/10.1145/2806416.2806558

Trip recommendation has been studied recently. [17] analyzed the characteristics of travel packages and proposed a graphical model to extract the topics conditioned on tourists, areas and travel seasons for personalized travel package recommendation. [3] developed a Bayesian learning model to extract travel paths from photos and conducted personalized travel recommendation according to user-specific profiles. All these works, however, adopt probabilistic models to generate a possible travel package or path but do not consider the objective function to maximize the user's happiness under the trip and other constraints.

The recent work [9] formulated the trip recommendation as a constrained objective function and presented a dynamic programming solution. Their assumption is that POIs can be grouped into several types or categories and the user knows the order of visiting POI types and likes to visit POIs of each type exactly once in a predetermined order. The restriction of visiting each type exactly once in a pre-determined order significantly reduces the search space. For example, for 150 POIs falling into 5 types equally, the original 150! possible routes are reduced to 30^5 if the order is fixed. In real world applications, however, the user may not provide this order either because she does not care about the order or because she is concerned that such a fixed order may restrict her options. In addition, their work does not consider the POI availability and the uncertainty of traveling time.

A detailed review of related work is presented in Section 8.

1.2 Contributions

In this paper, we address the trip recommendation by taking into account the following information and constraints: (1) the user's personalized preferences on POIs; (2) the user's time budget that constrains the total traveling and visiting time; (3) the time window for the POI availability; (4) the uncertainty of traveling time between POIs. We formulate the above requirements in our TripRec problem. The goal of the problem is to find an optimal trip that maximizes user happiness under the constraint that all the POIs in the trip can be visited and the trip can be completed within the user time budget with a probability not less than a user specified threshold. This problem is NP-hard as it is a special case of either the Knapsack problem [11] or the Orienteering problem [25].

We solve this problem by using the information and constraints in (1)-(4) to prune unpromising candidate trips. Our algorithm has an offline step and an online step. In the *offline step*, we apply collaborative filtering adopted to items with features to estimate user's preferences on unvisited POIs based on available check-in data. This step is performed only once as it applies to all users. In the *online step* where the user's time budget constraint and start/destination locations are provided, we search for the optimal trip route under the various constraints discussed above. We present two optimal solutions that guarantee to find the optimal trip if it exists. One is based on a state expansion approach and one is based on a prefix based depth-first search strategy. We also present two heuristic solutions that find "good trips" with a significantly better runtime than the optimal solutions. We evaluated all solutions on two real life LBSN data sets, Yelp and Foursquare, and demonstrated the superiority over previous trip recommendation algorithms.

1.3 The Road Map

The rest of the paper is organized as follows: Section 2 defines the problem. Section 3.1 presents the personalized rating estimation for POIs, which corresponds to the offline step, and Section 3.2-3.3 presents our modeling of POI availability and uncertain traveling time. These are the key factors that distinguish our modeling of trip recommendation from previous ones. Section 4 presents the optimal solution based on the state expansion approach, and the state relaxing strategy which scarifies optimality for efficiency. Section 5 introduces the second optimal solution based on the prefix based depth-first search strategy. We present an efficient heuristic solution in Section 6. Section 7 presents experimental evaluation of all solutions. Section 8 presents a review of related works. Finally, we conclude the paper with a discussion on extension to other scenarios of trip recommendation.

2. PRELIMINARY

This section describes our data model and the trip recommendation problem.

2.1 Data model

POI map: We assume that there are n POIs in a (virtual) complete directed graph $G = (V, E)$. $V = \{1, \cdots, n\}$ is a set of POIs. Each POI $i \in V$ is associated with the following information: a touring time m_i, indicating the typical or average staying time for users, and opening hours $[O_i, C_i]$, indicating that i opens at time O_i and closes at time C_i. Each edge $e_{ij} \in E$ represents the route from i to j, where $i, j \in V$, and associates with a traveling time t_{ij} following a distribution with probability density function $f_{ij}(\cdot)$. We assume that these functions $f_{ij}(\cdot)$ are given and that traveling times for different routes e_{ij} are independent.

Rating matrix: We consider a set of users where a user u may rate a POI i after visiting i. A rating matrix R contains all observed ratings r_{ui}. The rating matrix is usually extremely sparse with most entries undefined since a user may only rate a few POIs. Besides, a user u could leave comments on POI i when rating i, represented by a bag of words B_{ui} (If a user u does not rate a POI i, $B_{ui} = \emptyset$). The "content" of POI i is defined as $B_i = \bigcup_u B_{ui}$. Based on the matrix R and comments, we could estimate a user u's rating for an unvisited POI j, denoted as r^*_{uj}.

A trip route: For a specified source location x and a destination location y, and a departure time T_0, where x and y are not necessarily distinct, a trip route has the form $x \rightarrow \cdots i \cdots \rightarrow y$, that starts from x at the time T_0, visits each POI i listed in the route in order, and ends at y. We assume that the probability density functions $f_{xi}(\cdot)$ and $f_{iy}(\cdot)$ are known for any POI i, and we set $r^*_{ux} = r^*_{uy} = 0$, $m_x = m_y = 0$, $O_x = O_y = T_0$, $C_x = C_y = +\infty$. Such settings ensure that visiting x and y does not cost time because they serve only as the departure and destination locations for a trip. The score of a trip route $\mathcal{P}$ for a user u is defined by an additive function $F(\mathcal{P}, u) = \sum_{i \in \mathcal{P}} r^*_{ui}$. This function simply sums up the estimated ratings r^*_{ui} for all POIs in the route, which models the happiness of u with respect to the route $\mathcal{P}$.

Constraints on a trip route: *POI availability constraint:* a user is considered to visit a POI i only if the user spends m_i time at i during the opening hours $[O_i, C_i]$. Therefore, if a user arrives at i before O_i, she has to wait until the opening hour, and the user should arrive at i no later than $C_i - m_i$ to gain the happiness score. *Time budget constraint:* the whole trip is completed within a period of time b, including traveling time and touring time at POIs. *Completion probability constraint:* the probability that a trip finishes at the destination y by the time $T_0 + b$ is not less than a user specified threshold $\theta \in [0, 1)$.

2.2 Problem definition

PROBLEM 1 (TRIPREC). *Given a user u with the source x and the destination y, a departure time T_0, a time budget b, and a threshold $\theta \in [0, 1)$, we want to find an optimal trip route $\mathcal{P}$*

that maximizes user happiness $F(\mathcal{P}, u)$ under the following constraints: (1) The route starts at location x and ends at location y. (2) The route satisfies the POI availability constraint, the time budget constraint and the completion probability constraint.

In the rest of the paper, we first model the user's personalized preferences and the trip constraints; then, we present several approaches to search the optimal trip route according to the estimated preferences for TripRec.

3. MODELING PREFERENCES AND CONSTRAINTS

In this section, we discuss our modeling of user preference and trip constraints.

3.1 Estimating user preferences

Most existing POI recommendation methods either consider no content information of POIs or treat content information as side information (more discussion in Section 8). We believe that content information of POIs should play a more central role in user preference in that a user likes a POI because of certain features of the POI. To this end, we adopt the feature-centric collaborative filtering proposed in [33]. Unlike the traditional collaborative filtering on POIs, this approach performs collaborative filtering on the features of POIs and determines the rating on a POI using the predicted ratings on the features of the POI.

First, we transform the original user-POI rating matrix R into a user-feature matrix R', where each row represents a user u and each column represents a feature f in $\bigcup_i B_i$ for POIs i. An entry (u, f) in R' stores the aggregated rating on the feature f over the POIs i such that B_i contains f and i are rated by u:

$$g_{uf} = agg(\{r_{ui} | f \in B_{ui} \text{ and } r_{ui} \text{ is defined}\}) \quad (1)$$

In this work, $agg(X)$ returns the average of the values in X, but other aggregation operations are possible. $agg(X)$ is undefined if X is empty.

Then, we apply matrix factorization [12] to R' to extract the latent user vectors p_u for users u and the latent feature vectors q_f for features f. To predict the rating of user i on a POI i, we aggregate the predicted ratings $p_u^T q_f$ over all the features f in B_i:

$$r^*_{ui} = agg(\{p_u^T q_f | f \in B_i\}) \quad (2)$$

We will use r^*_{ui} as the estimated rating of a user u on a POI i. Thus, if the user is estimated to rate highly most features of a POI, the user is estimated to rate highly the POI. Note that the estimation of user ratings is performed offline and only once.

3.2 Testing time constraints

The basic idea of trip planning is to extend the route $\mathcal{P}$ gradually. Suppose that i is the last POI of $\mathcal{P}$, which satisfies the time budget and POI availability constraints, and π_i is the starting time of visiting i. We may extend the route by adding a new POI j after visiting i. We use the `Sat` function to test if the POI availability and the time budget constraints are satisfied after the extension. $\texttt{Sat}(i, j, \pi_i)$ returns *true* if $\pi_j + m_j \leq C_j$ and $\pi_j + m_j + t_{jy} \leq T_0 + b$, where $\pi_j = \max\{\pi_i + m_i + t_{ij}, O_j\}$ indicates the starting time of visiting j. This testing ensures that the user can get the full service at the POI j and still reach the destination y within the time budget.

3.3 Modeling uncertain traveling time

The above assumes that traveling time t_{ij} for a sub-route $i \to j$ is deterministic. However, even the traveling time can be estimated from historical data and external resources [20], the real traveling time remains uncertain due to many uncertain factors that could affect the traffic. To model this uncertainty, we shall treat the traveling time t_{ij} as a random variable following a certain distribution with probability density function $f_{ij}(\cdot)$. In this case, the best one can guarantee is that the probability that a trip $\mathcal{P}$ can be finished within a given time budget b is above some specified threshold θ. This probability, called *completion probability*, is denoted by $\psi(\mathcal{P})$ so that $\psi(\mathcal{P}) \geq \theta$.

We can modify the above constraint testing function `Sat` for uncertain traveling time as follows. Let $E[t_{ij}]$ denote the expectation of t_{ij}. $\texttt{Sat}(i, j, \pi_i)$ returns *true* if $\pi_j + m_j \leq C_j$, $\pi_j + m_j + E[t_{jy}] \leq T_0 + b$, and $\psi(\mathcal{P}) \geq \theta$, where $\pi_j = \max\{\pi_i + m_i + E[t_{ij}], O_j\}$ indicates the expected starting time of visiting j.

Let us derive $\psi(\mathcal{P})$ for a route $\mathcal{P}$. Consider a sub-route $i \to j$, we assume that the probability density function $f_{ij}(\cdot)$ is known. For simplicity, we also assume that t_{ij} are independent for different pairs (i, j). Let χ denote the traveling time of the sub-route. The probability of the traveling time less than t is given as follows:

$$P(\chi < t) = \int_0^t f_{ij}(\delta) d\delta \quad (3)$$

Suppose that we extend $i \to j$ by a POI k, the probability of traveling time of $i \to j \to k$ less than t is given by the multiple integral:

$$P(\chi < t) = \iint_D f_{ij}(\delta) f_{jk}(\gamma) d\delta d\gamma \quad (4)$$

where the domain $D = \{(\delta, \gamma) \in \mathbb{R}^2_{>0} \ : 0 < \delta + \gamma < t\}$ and $\mathbb{R}_{>0}$ means positive real number. In general, for any route $\mathcal{P} : i \to j \cdots j' \to k$ with c sub-routes, the probability of the total traveling time χ less than t is estimated by

$$P(\chi < t) = \iint \cdots \int_D f_{ij}(\delta_1) \cdots f_{j'k}(\delta_c) d\delta_1 d \cdots d\delta_c \quad (5)$$

where $D = \{(\delta_1, \cdots, \delta_c) \in \mathbb{R}^c_{>0} \ : 0 < \delta_1 + \cdots + \delta_c < t\}$. This probability can be computed given all the probability density functions $f_{ij} \cdots f_{j'k}$.

THEOREM 1 (**COMPLETION PROBABILITY BASED PRUNING**). *Let χ be the total traveling time of a route $\mathcal{P} : i \to \cdots \to j$ and let χ' be the total traveling time of another route $\mathcal{P}' : i \to \cdots j \to k$ obtained by adding a new POI k. Assume that both routes start at i at the same time. $P(\chi' < t) \leq P(\chi < t)$.*

PROOF. For simplicity, we consider a route $i \to j$ and the extension $i \to j \to k$, but the proof for the general case is similar. Due to the independence of traveling time at different sub-routes, $P(\chi' < t) = \iint_D f_{ij}(\delta) f_{jk}(\gamma) d\delta d\gamma$, so $P(\chi' < t) = \int_0^t f_{ij}(\delta) \int_0^{t-\delta} f_{jk}(\gamma) d\gamma d\delta \leq \int_0^t f_{ij}(\delta) \cdot 1 d\delta = P(\chi < t)$. □

In other words, the probability of finishing a route within the time budget is never increased by extending the route with one more POI at the end. This is because adding a POI at the end of a route does not affect the traveling time between the previous POIs of the route, but reduces the chance of completing the route within the time budget due to the additional time of traveling to and visiting the new POI. We shall use this property to prune the trips that have their completion probability below the specified threshold θ.

Various studies and methods have been proposed to estimate travel time distributions in the literature. See [31] for a comparison of these methods. Our method does not depend on the choices of such distribution, provided that the probability $P(\chi < t)$ can be computed for a route $\mathcal{P}$. For concreteness, we adopt the log-normal

distribution in [27]. The traveling time t_z on the zth sub-route in a route independently follows the log-normal distribution with parameter μ_z, σ_z, that is, $t_z \sim \mathcal{LN}(\mu_z, \sigma_z^2)$. The expected traveling time $E[t_z]$ is given by $\exp(\mu_z + \sigma_z^2/2)$. The total traveling time χ of a route $\mathcal{P}$ made of multiple sub-routes is the sum of the traveling time t_z of each sub-route, i.e., $\chi = \sum_z t_z$. According to [19], χ can be approximated by another log-normal distribution $\mathcal{LN}(\mu_\chi, \sigma_\chi^2)$ with the following parameters:

$$\begin{aligned} \sigma_\chi^2 &= \log\left(\frac{\sum e^{2\mu_z+\sigma_z^2}(e^{\sigma_z^2}-1)}{(\sum e^{\mu_z+\sigma_z^2/2})^2}+1\right) \\ \mu_\chi &= \log\left(\sum e^{\mu_z+\sigma_z^2/2}\right) - \frac{\sigma_\chi^2}{2} \end{aligned} \tag{6}$$

With this distribution for the total traveling time χ, the probability of completing a trip $\mathcal{P}$ is $\psi(\mathcal{P}) = P(\chi < t)$, where t is the time available for traveling, that is, $t = b - \sum m_j$. For the log-normal distribution,

$$P(\chi < t) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{\ln t - \mu_\chi}{\sigma_\chi\sqrt{2}}\right)\right] \tag{7}$$

where $\mathrm{erf}(t) = \frac{2}{\sqrt{\pi}}\int_0^t e^{-\delta^2} d\delta$ [27].

4. STATE EXPANSION

In this section, we present a state expansion algorithm that guarantees to find an optimal route if it exists. The idea is to consider each partially generated route as a *state* associated with some ending POI i, representing a feasible trip route $x \to \cdots \to i \to y$ that has i as the ending POI before reaching the specified destination y. Each state is labeled by $s = (K, T, H, \rho, i)$, where K is the set of POIs already visited, excluding x and y. T is the starting time of visiting at i (i.e., π_i), H is the overall happiness collected (i.e., $F(K, u)$), ρ is the current route $x \to \cdots \to i$ (without the sub-route $i \to y$) and i is the ending POI. These parameters are denoted as s_K, s_T, s_H, s_ρ and s_i, respectively. Initially, there is only one state $s_0 = (\emptyset, T_0, 0, x, x)$, representing the trip route $x \to y$.

At the κth iteration ($\kappa > 0$), the state expansion algorithm extends each state of size $\kappa - 1$ into a new state of size κ by adding a new POI. Specifically, a state $s = (K, T, H, \rho, i)$ is extended into a new state s' associated with POI $j \notin s_K \cup \{x, y\}$ according to the following rules:

$$\begin{cases} s'_H = s_H + r^*_{uj} \\ s'_K = s_K \cup \{j\} \\ s'_T = \max\{s_T + m_i + E[t_{ij}], O_j\} \\ s'_\rho = s_\rho \to j \end{cases} \tag{8}$$

A new state s' is *feasible* if $\mathtt{Sat}(i, j, s_T)$ returns *true*. Intuitively, this means that the partial route of the state can be extended to j and then finished at the destination y within the time budget with the completion probability no less than the threshold θ.

4.1 Dominance of states

It is possible that the same ending POI s_i could be reached by different states s of the same POIs s_K, corresponding to different visiting orders. Not all such states need to be maintained because some do not lead to the optimal solution.

We say that a state s *dominates* a state s' if

$$(s_i = s'_i) \wedge (s_K = s'_K) \wedge (s_T \le s'_T) \wedge (\psi(\bar{s_\rho}) \ge \psi(\bar{s'_\rho})) \tag{9}$$

where $\bar{\cdot}$ forms the complete trip by adding y into the route, say $\bar{\rho} = \rho \to y$. Note that $s_K = s'_K$ implies $s_H = s'_H$, i.e., s and s' give the same user happiness. Intuitively, s dominates s' if all of the following conditions hold: the two states s and s' represent two routes $\rho \to y$ and $\rho' \to y$ containing the same set of POIs, the starting visit time of i in s is no later than that in s', and the completion probability of s is no less than that of s'. We assume that the procedure `Check` tests the dominance: `Check`(s, s') returns *true* if s dominates s' (i.e., Eqn 9) and *false* otherwise.

LEMMA 1. *If a state s dominates a state s' and let s_e and s'_e denote the states obtained from extending s and s' with a new POI j at the end, respectively, then s_e dominates s'_e.*

PROOF. Suppose that s and s' represent the routes $\rho \to y$ and $\rho' \to y$. Then s_e and s'_e represent the routes $\rho \to j \to y$ and $\rho' \to j \to y$. It is easy to see that the first three conditions in Eqn (9) remain true for s_e and s'_e. To see the last condition, since s dominates s', both ρ and ρ' have the same ending POI i. If we regard i as the new source of the following identical trip $i \to j \to y$ for both s_e and s'_e, the completion probability of this trip in s_e is no less than that in s'_e because it starts earlier in s_e. Combined with the previous trip ρ and ρ', this condition still holds. □

By repeatedly applying Lemma 1, we have the next theorem.

THEOREM 2 (**DOMINANCE BASED PRUNING**). *Assume that a state s dominates a state s'. If s' can be extended into an optimal trip by a sequence of POIs, so is s by the same sequence of POIs.*

From the above theorem, it suffices to consider only non-dominated states. We will use this property to remove all dominated states without affecting optimality.

4.2 Algorithm

Algorithm 1 summarizes the state expansion for TripRec. Starting at the initial state $S = \{s_0\}$, the algorithm extends the current set of states, S, by adding one new POI at the end of a route in S. If the states in S have the size κ, the new states in S' have the size $\kappa + 1$. The two **for** loops extend each state in S with an unvisited POI and only feasible states are kept. Meanwhile, **Line** 9-10 conducts the dominance test and removes dominated states. **Line** 12-13 records the optimal route with the maximal user happiness. The time complexity of Algorithm 1 is $\mathcal{O}(n^2 2^n)$, which is exponential but much faster than the brute-force search $\mathcal{O}(n!)$. However, due to the time budget and POI availability constraints, each trip typically consists of only a small fraction of all POIs. If the maximum number of POIs in a trip is τ, where $\tau \ll n$, 2^n is replaced with $C(n, \tau)$ in the above complexity.

4.3 State relaxing

The above dominance based pruning applies only to two states that have exactly the same set of POIs, i.e., $s_K = s'_K$. If we are willing to sacrifice optimality for efficiency, it is possible to have a more aggressive pruning by replacing the condition $s_K = s'_K$ with $|s_K| = |s'_K|$ (i.e., visiting the same number of POIs) and $s_H \ge s'_H$ (i.e., s representing a more preferred route than s'). So the dominance test condition in Eqn (9) is relaxed into

$$(s_i = s'_i) \wedge (|s_K| = |s'_K|) \wedge (s_T \le s'_T) \wedge (s_H \ge s'_H) \wedge (\psi(\bar{s_\rho}) \ge \psi(\bar{s'_\rho})) \tag{10}$$

Intuitively, with this relaxed dominance relationship, the route for s takes less time and generates a higher happiness than the route for s', while reaching the same ending POI i. In other words, the route represented by s gives the user more happiness and more remaining time than the route represented by s', thus, is preferred. We call the pruning based on this relaxed dominance relationship *state relaxing*. State relaxing applies to all states ending at the same

Algorithm 1: State expansion

input : POI map G, user u's specific preferences r^*_{ui} for each POI i, departure time T_0, time budget b
output: optimal TripRec trip route, $\mathcal{P}$

$s_0 \leftarrow (\emptyset, T_0, 0, x, x)$, $s^* \leftarrow s_0$;
$S \leftarrow \{s_0\}$, $S' \leftarrow \emptyset$;
while $S \neq \emptyset$ **do**
  for $s \in S$ **do**
    for $j \in V \setminus s_K$ **do**
      if `Sat` (i, j, s_T) **then** // $i \equiv s_i$
        $s'_T \leftarrow \max\{s_T + m_i + E[t_{ij}], O_j\}$;
        $s' \leftarrow (s_K \cup \{j\}, s'_T, s_H + r^*_{uj}, s_\rho \rightarrow j, j)$;
        if $\exists s'' \in S'$: `Check` (s', s'') =*true* **then**
          remove s'' from S';
        add s' to S';
        if $s'_H > s^*_H$ **then**
          $s^* \leftarrow s'$;
  $S \leftarrow S', S' \leftarrow \emptyset$;
return $s^*_\rho \rightarrow y$ as $\mathcal{P}$

POI through visiting the same number of POIs which significantly reduces the size of the set of states S in Algorithm 1 as each ending POI may only be associated with a few states. So the time complexity is decreased from $\mathcal{O}(n^2 2^n)$ to $\mathcal{O}(cn^2)$ for some constant c.

However, due to the POI availability constraint, state relaxing loses the optimality in some cases. For example, suppose $A \rightarrow D \rightarrow C$ dominates $A \rightarrow B \rightarrow C$ according to Eqn (10) (we omit the source x and destination y for simplicity), so the former is kept and the latter is eliminated. Now suppose that B only opens in the morning and D opens until midnight. Then the route $A \rightarrow D \rightarrow C \rightarrow B$ may be infeasible due to the late visit to B while the route $A \rightarrow B \rightarrow C \rightarrow D$ could be the optimal solution, but it cannot be generated because $A \rightarrow B \rightarrow C$ was pruned. Section 7 will study experimentally the trade-off between efficiency and user happiness for the state relaxing strategy.

5. PREFIX BASED DEPTH-FIRST SEARCH

If the states of size κ are represented by the nodes at level κ in a tree structure (with the root at level 0), Algorithm 1 generates the states in a *breadth-first* manner in that the states at level κ are generated before any state at level $\kappa+1$ is generated. For loose time budget and POI availability constraints, this approach may have to keep many "open" states in memory (i.e., all states of the same size), which imposes a bottleneck on the memory requirement.

5.1 Prefix based depth-first enumeration tree

To address this limitation, we present a prefix based depth-first enumeration of states in which a tree structure representing the states is searched in the *depth-first* manner so that only the current branch at any level is searched at any time. First, for the given user u, we order all POIs i by the estimated rating r^*_{ui}. This order together with our tree enumeration strategy below ensures that POIs with larger ratings are considered before those with smaller ratings in the construction of a route. For presentation, we consider $V = \{A, B, C, D, E\}$ of five POIs, excluding the source x and destination y, and we assume that these POIs are ordered in the

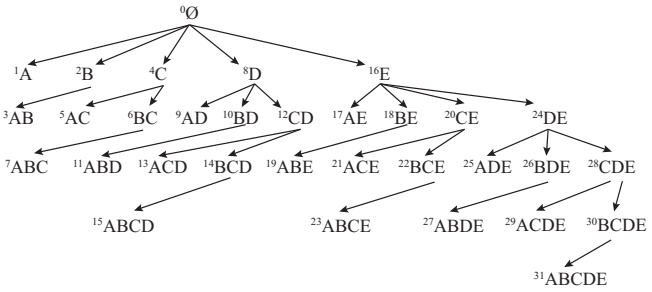


Figure 1: Prefix based depth-first enumeration tree

descending order of estimated ratings for u. The *prefix* of a POI i refers to the set of POIs that precede i in this order.

Figure 1 shows the tree structure for enumerating all the subsets of V with POIs arranged in the above order. The nodes in the tree are generated from left to right in a specific depth-first order, as indicated by the numbers aside the nodes. Each node is labeled by a set of visited POIs arranged in the above order, and the root is labeled by the empty set $\emptyset$. Intuitively, a node with the label K represents all the non-dominated feasible routes that visit exactly all the POIs in K. These routes are divided into $|K|$ groups according to each ending POI in K. To grow the tree, for a node v with a label ending at a POI i, a child node is generated by appending some POI j that precedes i in the above order to the front of the label of v. For example, Node 7 with the label ABC is generated as a child node of Node 6 with label BC by appending A to the front of BC.

Subset first property. An important property of the above depth-first enumeration is that *a label K is always enumerated before any of its supersets*. For example, the proper subsets of ABC are enumerated at Nodes 0-6 and ABC is enumerated at Node 7. This property ensures that, when computing the feasible routes at the node for a label K with the ending POI i, the feasible sub-routes visiting all the POIs in $K \setminus i$ have already been computed at the node with the label $K \setminus i$, so we can retrieve the stored information for such feasible sub-routes to construct the feasible routes at the node for K, by checking the time budget and POI availability constraints and pruning dominated states.

For example, Node 15 with the label $ABCD$ is a child node of Node 14 with the label BCD. There are 4! possible routes at Node 15, but many can be pruned since they either violate the time budget and POI availability constraints, which can be tested by the procedure `Sat`, or are dominated by other routes, which can be tested by the procedure `Check`. The feasible routes at Node 15 can be divided into 4 groups corresponding to the ending POIs A, B, C, and D, respectively. The group for the ending POI A can be constructed by retrieving the feasible sub-routes from the already computed Node 14 and appending A to the end by checking the constraints. With the dominance pruning discussed in Section 4 (i.e., Eqn 9), only the non-dominated routes will be kept for this group. The groups for the ending POIs B, C, D can be constructed similarly, by retrieving the feasible sub-routes from the nodes with the labels ACD, ABD, ABC, i.e., Node 13, Node 11, Node 7. Note that these nodes were already computed because their labels are subsets of $ABCD$.

The next theorem lays the foundation for our prefix based depth-first enumeration algorithm for computing the optimal route.

THEOREM 3. *(1) Every non-dominated feasible route stored at a node with the label K and ending POI i must have a prefix that is a non-dominated feasible route stored at the node with the label*

Algorithm 2: `PrefixDFS`(U, K)

```
1  for j ∈ U in order do
2      U⁻ ← prefix of j in U;
3      K⁺ ← K ∪ {j};
4      if |K⁺| = 1 then
5          if Sat(x, j, T0) = true then
6              Ω[K⁺]_H ← r*_uj;
7              add (j, x → j, max{E[t_xj] + T0, O_j}) to
               Ω[K⁺]_L;
8              PrefixDFS(U⁻, K⁺);
9      else
10         for k ∈ K⁺ do
11             K⁻ ← K⁺ \ k;
12             if Ω[K⁻] is not empty then
13                 find l = (i, ρ, T) in Ω[K⁻]_L such that
                   Sat(i, k, T) = true and ρ → k is
                   non-dominated;
14                 if l is found then
15                     T' ← max{T + m_i + E[t_ik], O_k}
16                     if Ω[K⁻]_H + r*_uk > H then
17                         update P and H;
18                     Ω[K⁺]_H ← Ω[K⁻]_H + r*_uk;
19                     add (k, ρ → k, T') to Ω[K⁺]_L;
20         if Ω[K⁺] is not empty then
21             PrefixDFS(U⁻, K⁺);
```

$K \setminus i$. (2) The node with the label $K \setminus i$ is enumerated prior to the node with the label K.

PROOF. (1) Consider a non-dominated feasible route at a node with the label K, written as $\rho' = \rho \rightarrow i$, where ρ is the prefix containing the POIs in $K \setminus i$. From Theorem 1, ρ must satisfy all the constraints, and from Theorem 2, it suffices to consider only non-dominated feasible route ρ at the node with the label $K \setminus i$. This proves Part (1). Part (2) follows from our discussion on the prefix based depth-first enumeration. □

5.2 Implementation

Based on the above discussions, we design a hash map Ω to store the computed results for each node in the tree, whose key is the label K of the corresponding node, and whose value, $\Omega[K]$, contains the information about the non-dominated feasible routes for the node. $\Omega[K]$ has two components, H and L. H is the total happiness for the POIs in K. L is a list $(l_1, l_2, \cdots)$, where each entry l in L has the form (i, ρ, T) and represents a non-dominated feasible route ρ for the node. $i \in K$ is the ending POI of ρ and T is the starting time of visiting i. Essentially, $\Omega[K]$ is a compact representation of all the states that have the same POI set K in Section 4. Let $\Omega[K]_H$ and $\Omega[K]_L$ denote the H and L components of $\Omega[K]$.

We implement the above prefix depth-first search in Algorithm 2 as a recursive procedure `PrefixDFS`(U, K) with a set U of ordered POIs and a node label K as the parameters. Intuitively, `PrefixDFS`(U, K) enumerates the subtree at the node with the label K and U is the set of POIs available for extending the label K within the subtree. The inputs to the algorithm are the POI map G, the departure time T_0, the time budget b and user-specific preferences r^*_{ui}. The output is the optimal route and its happiness, stored in the global variables $\mathcal{P}$ and $\mathcal{H}$. The main algorithm is the call `PrefixDFS`$(V, \emptyset)$ with the set of POIs V in the POI map G.

The algorithm extends the label K by each available POI j in U, creating the child node with the label $K^+ = K \cup \{j\}$ and U^- being the prefix of j in U. If K is empty, **Line** 4-8 adds the route $x \rightarrow j \rightarrow y$ to the hash entry for K^+ and recursively calls `PrefixDFS`(U^-, K^+). If K is not empty, **Line** 10-19 adds all non-dominated feasible routes having the POI set K^+ to the hash entry for K^+. In particular, for each $k \in K^+$, **Line** 13 searches for the non-dominated feasible route for the POI set K^+ and ending at k. This route consists of a non-dominated feasible route $l = (i, \rho, T)$ for the POI set $K^- = K^+ \setminus k$ and the ending POI k (Theorem 3) such that it satisfies all the constraints, i.e., `Sat`$(i, k, T) = true$, and the the route $\rho \rightarrow k$ is non-dominated (by the conditions in Eqn (9)). From Theorem 3, all non-dominated feasible routes for K^- are stored at the node K^- and were computed already. If there exists such an l, $(k, \rho \rightarrow k, T')$ for the extended route $\rho \rightarrow k$ is added to $\Omega[K^+]_L$. After considering every $k \in K^+$, if $\Omega[K^+]$ is not empty, the algorithm calls `PrefixDFS`(U^-, K^+) recursively.

Note that the algorithm does not actually materialize the entire enumeration tree; instead, it enumerates the nodes in the tree in the depth-first order. The result at each node is stored in the hash map.

6. HEURISTIC APPROXIMATION

In this section, we propose a simple heuristic algorithm that is essentially linear in the total number of POIs while maintaining the quality of the route. The idea is intuitive: starting with the initial trip route $x \rightarrow y$, we insert one POI at a time between two adjacent POIs in the current trip route so that (i) the insertion preserves the satisfaction of all the constraints and (ii) some score of the route is maximized (to be discussed shortly). For example, inserting a POI A into $x \rightarrow y$ gives the route $x \rightarrow A \rightarrow y$, then inserting a POI B before A gives $x \rightarrow B \rightarrow A \rightarrow y$, and so on. To avoid the local optimum, we generate some small number of routes (say 2-3) by applying this method to the set of remaining POIs not contained in the previously generated routes, and we choose the best route from all the routes generated. The time complexity of this algorithm is $\mathcal{O}(cn)$, where n is the number of POI and c is a constant, because each insertion considers at most n unvisited POIs. Note that the length of a route is usually small due to the time budget and POI availability constraints.

A remaining issue is to check whether inserting a POI k between two adjacent POIs i and j (i.e., the sub-route $i \rightarrow j$ already exists in the trip) preserves the satisfaction of the time budget constraint, the POI availability constraint and the completion probability constraint. We focus on the POI availability constraint because it is easy to check the other two constraints. We assume $\lambda_{ij} = 1$, that is, a visit to POI i is followed by a visit to POI j. Before the insertion of k, the arrival time at POI j, denoted by a_j, is computed by

$$a_j = \pi_i + m_i + E[t_{ij}] \tag{11}$$

where π_i is the starting time of visiting at POI i. The wait time at POI j, w_j, is computed by

$$w_j = \max\{0, O_j - a_j\} \tag{12}$$

The maximum allowed delay time at i to preserve the satisfaction of constraints, denoted by v_i, is computed by

$$v_i = \min\{C_i - \pi_i - m_i, w_j + v_j\} \tag{13}$$

where $C_i - \pi_i - m_i$ is the maximum allowed delay time to keep the visit to i available (before it closes), and $w_j + v_j$ is the maximum allowed delay time to keep the visit to j available.

For example, if $\pi_i = 10am, C_i = 2pm, m_i = 1h$, the maximum allowed delay time for i itself is $3h$, i.e., the user can at most delay to arrive at $1pm$. However, a delay at i may affect the visit to the next POI j. If $w_j + v_j = 2h$, that is, the visit to j can be delayed at most $2h$, then the maximum allowed delay time at i is $v_i = \min\{3h, 2h\} = 2h$.

The insertion of k between i and j is possible only if the new route satisfies the probability constraint according to Eqn (7) and the extra time caused by the insertion does not exceed the maximum allowed delay time at j, i.e., $w_j + v_j$. The extra time ε_k for inserting POI k is given by

$$\varepsilon_k = E[t_{ik}] + w_k + m_k + E[t_{kj}] - E[t_{ij}] \quad (14)$$

If $\varepsilon_k \leq w_j + v_j$, k can be inserted between i and j and the insertion transforms $i \to j$ into $i \to k \to j$, thus, $\lambda_{ik} = \lambda_{kj} = 1, \lambda_{ij} = 0$.

To determine the score of the insertion of k, we calculate the ratio γ_k as follows:

$$\gamma_k = (r^*_{uk})^2/\varepsilon_k \quad (15)$$

This ratio measures the gain of happiness per unit of extra time of visiting k. The square of r^*_{uk} places more emphasis on the rating. Since a smaller ε_k has less effect on the feasibility of the whole route, the POI k with a larger ratio γ_k is preferred. We try every adjacent (i, j) in the current route to find the best γ_k.

After each insertion, the arrival time, wait time, and maximum allowed delay time of all affected POIs in the route should be updated according to Eqn (11-13). For example, if k is inserted to form a new route $x \to i_1 \to i_2 \cdots \to k \to j_1 \to j_2 \cdots \to y$, the arrival time, the wait time and the maximum allowed delay time of any POIs after k $(j_1, j_2, \cdots)$ should be updated, and the maximum allowed delay time of any POIs before k $(i_1, i_2, \cdots)$ should be updated. Moreover, the updates must follow the orders imposed by the dependency in Eqn (11-13). For example, Eqn (13) requires first updating a later POI before updating an early POI in a route.

7. EXPERIMENTS

This section presents the empirical evaluation of the proposed methods.

7.1 Experimental setup

We adopt the Yelp[1] and Foursquare[2] data sets in our experiments. Both data sets were previously used for recommendation evaluation in [10]. The Yelp data set contains 45,981 users, 229,906 ratings of 1-5 scales, 11,537 POIs, plus text reviews on POIs. We preprocessed the reviews by removing stop words and infrequent words occurring in < 100 reviews, and using the remaining 8,519 keywords as the features. The feature set or content of a POI, B_i, consists of all keywords contained in the reviews about the POI. The Foursquare data set contains 20,784 users, 153,477 binary 0/1 ratings, 7,711 POIs, and user published tweets when checking-in at a POI. We obtained 1,377 features after preprocessing the tweets.

For each POI i, the touring time m_i is set to 1 hour, and the opening hours were generated from a Gaussian distribution, $(C_i - O_i) \sim \mathcal{N}(\mu, \delta)$ with the mean $\mu = 5$ hours and the standard error $\delta = 1$. The open time O_i was generated using a uniform distribution, $O_i \sim \mathcal{U}(8, 12)$. We set the departure time T_0 to 8*am*. The expected traveling time $E[t_{ij}]$ for a pair of POIs (i, j) is estimated using Google Maps[3] with the driving mode. All the experiments were run on a PC with 2.53 GHz Quad-Core CPU and 12G memory.

[1] http://www.yelp.com/dataset_challenge/
[2] https://foursquare.com/
[3] https://www.google.com/maps

7.2 Rating accuracy of individual POIs

First, we evaluate the first step of our approach, that is, the accuracy of estimated ratings of POIs produced by the feature-centric collaborative filtering. For both data sets, we keep 90% rating data for training to conduct matrix factorization and use the remaining 10% rating data for testing the accuracy of estimated ratings. As in the literature [22], we use the standard RMSE (root mean squared error) and MAE (mean absolute error) as the accuracy metrics for POI recommendation. The smaller these values are, the better the result is.

Many POI recommendation approaches are based on topic modeling, for example, STM [10] and LCA [29] predict the probability of visiting a POI, for which the error specific metrics such as RMSE/MAE are incomputable because probabilities are not comparable with ratings. For this reason, we evaluate the following methods.

Probabilistic matrix factorization (denoted PMF): This is the classic matrix factorization on the user-item rating utility matrix [21] where POIs are treated as items. In PMF, matrix factorization is generalized as a probabilistic model where a latent user vector $p_u \sim \mathcal{N}(0, \alpha_p^{-1} I_D)$, a latent item vector $q_i \sim \mathcal{N}(0, \alpha_q^{-1} I_D)$. The predicted user u's rating on item i is given by $r^*_{ui} = p_u^T q_i$. We adopt the default settings in [21] and set $D = 10$, the dimensionality of user and item latent factors.

Collaborative topic regression (denoted CTR): This is the matrix factorization with topic modeling applied to the content of items described in [26]. For our data sets, items are POIs and content of user reviews on POIs. LDA is employed on POI i's content to learn the latent topic vector θ_i, which is incorporated into the PMF framework to confine the search of latent item vectors by setting $q_i \sim \mathcal{N}(\theta_i, \alpha_q^{-1} I_D)$. We adopt the default settings in [26] and set $D = 10$.

Feature centric collaborative filtering (denoted FCF): This is the proposed algorithm in Section 3.1. All the parameter settings are the same as in PMF.

Table 1 shows the results of accuracy of the above three methods. FCF achieves the best performance and has a clear improvement in terms of RMSE/MAE on both data sets. So we believe that the estimated rating by FCF is closer to the true rating. In the rest of the experiments, we study the performance of trip recommendation with the estimated rating r^*_{ui} being generated by FCF.

Table 1: RMSE and MAE. Lower values are better.

	Yelp		Foursquare	
Method	RMSE	MAE	RMSE	MAE
PMF	1.3169	1.0491	0.6197	0.5160
CTR	1.2850	1.0277	0.6000	0.5018
FCF	**1.2152**	**0.9720**	**0.5154**	**0.4402**
improve over PMF	7.7%	7.3%	16.8%	14.7%
improve over CTR	5.4%	5.4%	14.1%	12.2%

7.3 The fixed traveling time model

In this section, we evaluate the trip route $\mathcal{P}$ found by TripRec under the fixed traveling time model where the traveling time t_{ij} for a pair of POIs i and j is fixed, because all the baselines consider fixed traveling time. In this deterministic setting, a feasible route always satisfies the completion probability constraint. The model for uncertain traveling time will be considered in Section 7.4.

We focus on three major cities for trip planning, Phoenix (PX) in Yelp, New York city (NY) and Los Angeles (LA) in Foursquare, and choose Central City, Central Park, and Hollywood as both the

source and the destination in these cities respectively. For each city, we randomly pick up 100 users from the testing data, and for each user, we select the top $n = 150$ unvisited POIs, ranked by their estimated ratings, for trip recommendation. This n is a suggested number in [1]. Even with this restriction, the number of trips that consist of 5 POIs can reach billions, which is certainly infeasible for a brute-force search. We compare the following methods in terms of user happiness $F(\mathcal{P}, u)$ and runtime. All the methods adopt the personalized estimated ratings for each POI, learnt by FCF as input.

Greedy algorithm (denoted Greedy): This is the greedy algorithm from the operation research literature [24], which iteratively picks up a POI j with the highest ratio of r_j^*/t_{ij}, where i is the location selected at the last step. Note that we have added the POI availability constraint, which is not considered by [24].

Dynamic programming (denoted DP): This is the dynamic programming approach proposed in [9]. We adapt to the order constraint by setting a "global" type to each POI and fix the visiting order that is from "global" type to "global" type. However, the dynamic programming by filling up a 2-dimensional array [9] still cannot deal with the POI availability constraint.

Heuristic approximation (denoted HA): This is the heuristic algorithm proposed in Section 6. HA is designed for fast approximation and does not guarantee the optimality of solution.

State expansion (denoted SE): This is Algorithm 1 proposed in Section 4. Let SE-SR denote SE with state relaxing. SE guarantees the optimality of solution, but SE-SR does not.

Prefix based depth-first search (denoted PDFS): This is Algorithm 2 in Section 5 that uses the prefix based depth-first enumeration of POIs. PDFS guarantees the optimality of solution.

7.3.1 User happiness

Figure 2 (left column) presents the user happiness score of the trips found by all methods, with y-axis being the happiness score averaged over all testing users and x-axis being the time budget b of a trip (hours). Note that SE and PDFS generate exactly the trips of the same happiness score due to their optimality.

Overall, the number of POIs in the recommended route varies from 3 to 7 depending on the setting of the time budget b. As the time budget increases, the happiness of users generally increases. PDFS/SE is the best performer since they guarantee the global optimum. Interestingly, SE-SR yields a nearly optimal solution as the happiness is only slightly ($< 1\%$) lower than that of the optimal PDFS/SE.

HA performs in the third place and there is an obvious gap between HA and the best two. This is because HA only maintains one route during search, which makes it easy to fall into a local optimum. We will further explain this in the case study below. Greedy performs about 10% worse than HA, as its search strategy is rather simple. DP performs poorly on on all the testing cities, because it cannot deal with the POI availability constraint. In fact, only partial happiness is gained for such routes that some of the POIs are already closed when the user arrives, thus, leading to the low happiness scores for many users. Meanwhile, DP cannot guarantee a better result for a larger time budget.

7.3.2 Runtime

Figure 2 (right column) presents the average runtime per user, with y-axis being the runtime (seconds) and x-axis being the time budget of a trip b (hours). HA and Greedy have a fast and stable runtime because both HA and Greedy only maintain one route, but this feature also overlooks other possible combinations of POIs, thus hardly finding optimal solutions. SE suffers the out of memory

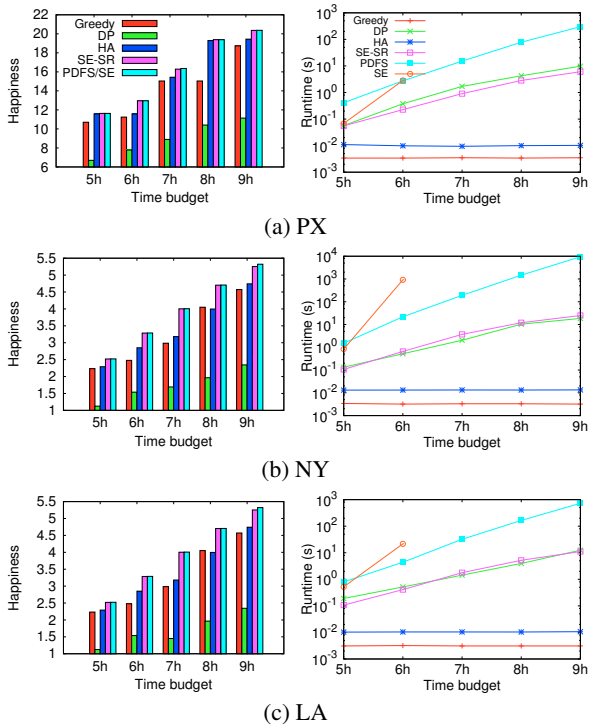


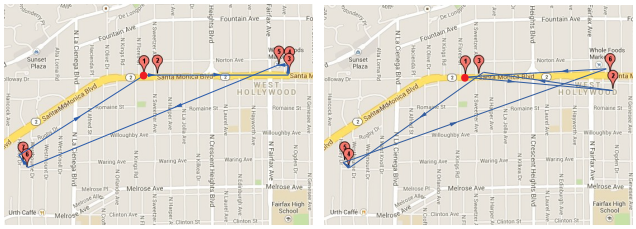
Figure 2: The fixed traveling time model: (left) happiness of trip routes found (y-axis) vs time budget (x-axis); (right) average runtime (y-axis) vs time budget (x-axis).

problem when the time budget is over 7 hours because there are too many "open" states in each iteration, which exhausts the memory when the time budget is large. PDFS avoids this problem by the prefix based depth-first enumeration. Although it takes the longest time at $b = 5h$, PDFS finds the optimal solution without having the steep increase of runtime encountered by SE. SE-SR takes substantially less time than SE by trading optimality for efficiency.

7.3.3 Case study

For a randomly selected user with $b = 8h$, Figure 3 shows the trip routes designed by PDFS and HA, on the local map of LA, where Location 1 is the source and the destination. The visit follows the increasing order $1 \rightarrow 2 \rightarrow \cdots \rightarrow 1$. PDFS and HA share many POIs in their recommended trip routes (e.g., POIs 2, 3, 6, 7 for PDFS) due to that both methods adopted the personalized preferences. However, HA maintains only one route and easily falls into a local optimum. For example, while POIs 1 and 3 are spatially far away from POIs 2 and 6 in Figure 3(b), HA visits these POIs in the order $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 1$, in which every sub-route is between two POIs that are far away, thus, too much time is spent on traveling. In contrast, PDFS designs the route in a circle, which reduces the number of sub-routes with long traveling time and allows the user to visit one more POI than HA within the same time budget.

In summary, PDFS finds the optimal solution with less runtime than SE; SE-SR is a very good trade-off for efficiency at a slightly lower happiness than the optimal solution; HA is very efficient but sometime has a significantly lower happiness. Overall, PDFS and SE-SR are two best performers considering both quality and efficiency.



(a) PDFS (5.72) (b) HA (4.89)

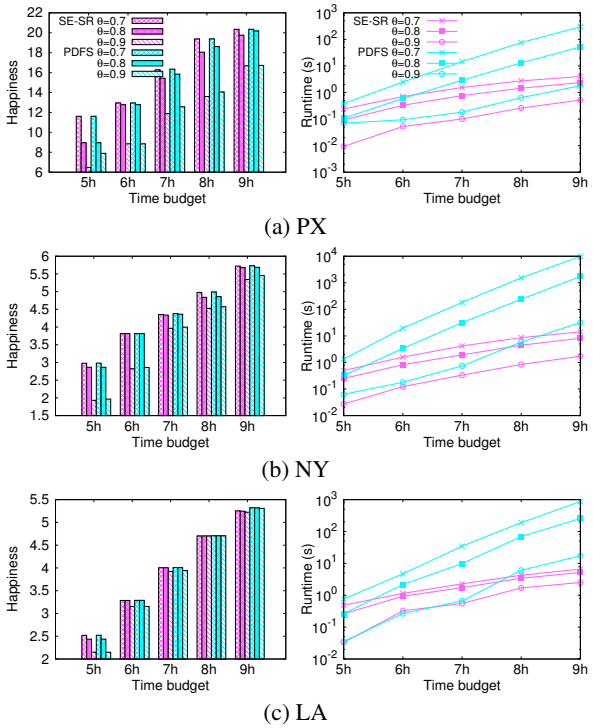
Figure 3: Case study of recommended trip routes for LA. The number in bracket is the happiness of the trip.

7.4 The uncertain traveling time model

In this section, we study the effect of the uncertain traveling time on SE-SR and PDFS. For the traveling time distribution of t_{ij} for a sub-route $i \rightarrow j$, we adopt the log-normal distribution $t_{ij} \sim \mathcal{LN}(\mu_{ij}, \sigma_{ij}^2)$ in Section 3.3. Note that $E[t_{ij}] = \exp(\mu_{ij}+\sigma_{ij}^2/2)$. σ_{ij} is generated from a uniform distribution to introduce the uncertainty, i.e., $\sigma_{ij} \sim \mathcal{U}(0.5, 2)$.

Figure 4 (left column) presents the happiness scores of SE-SR and PDFS with various threshold θ on completion probability. A color represents a method and a pattern represents a threshold θ on completion probability. Compared to the case for the fixed traveling time in Section 7.3, the happiness of both methods become lower given the same time budget. For example, there is about 20%-40% reduction of happiness from the fixed time cases at $\theta = 0.9$ and $b = 5h$. This is because the route designed in the previous section, although having a higher happiness, may violate the completion probability constraint due to the variance of traveling time, and a more strict constraint (i.e., higher threshold) results in less happiness. In practice, if the user prefers a more reliability of a trip, a route with higher completion probability but a bit less happiness is acceptable.

The uncertain traveling time model also accelerates the runtime of both methods, as shown in Figure 4 (right column). As the completion probability threshold θ increases, there are fewer feasible routes and both methods prune the routes with the probability below the threshold earlier. A cross examination with Figure 2 indicates that at $\theta = 0.7$, the runtime of these methods with modeling uncertain traveling time is close to that with the fixed traveling time model. However, it is almost an order of magnitude less in runtime at $\theta = 0.9$.



(a) PX

(b) NY

(c) LA

Figure 4: The uncertain traveling time model: (left) happiness of trip routes found (y-axis) by SE-SR and PDFS vs time budget b (x-axis); (right) average runtime (y-axis) vs time budget b (x-axis).

8. RELATED WORK

8.1 POI recommendation

Most location-based recommendation falls into this category, which scores each POI individually and recommends top-k POIs to a user. Some examples in this category include [4], [13], which consider no content information, [10], [16], and [29], which consider content as side information. The key difference between trip recommendation and POI recommendation is that POI recommendation considers neither the order of visiting POIs nor the time budget of users and the POI availability constraint. Recommending a visiting order of several POIs to maximize user satisfaction under such time constraints is the main focus of trip recommendation.

8.2 Travel package recommendation

Travel package or itinerary recommendation focuses on a tour of POIs instead of isolated POIs, which is similar to trip recommendation. [14, 30] applied spatio-temporal streams such as tagged photo streams or GPS trajectories to seek for a feasible path by topic modeling or Markov models. [8, 17] developed several probabilistic models to generate possible packages by considering cost, season, area, etc. [1] used user feedback to improve results on interactive tour recommendation. None of these works focus on the objective to maximize user's happiness.

[7, 9, 18] maximized user's happiness in travel recommendation. [9] assumes a fixed order on the types of POIs visited, which is not suitable in the presence of constraints such as opening hours of POIs. The greedy algorithm proposed in [7] cannot guarantee the global optimality. [18] adopted memory-based collaborative filtering to estimate user specific preferences, which are dynamical with time. All these works do not consider the POI availability and probability constraints. The first two works do not consider user specific preferences, thus, generate the same itinerary to all users.

8.3 Operation research and scheduling

The orienteering problem (OP) [25] studied in operation research and theoretical computer science is related to our problem. In OP, a set of vertices is given, each with a score. The goal is to determine a path, limited in length, that visits some vertices and maximizes the sum of the collected scores. However, there are some important differences between trip recommendation and OP. First, OP does not consider personalized user preferences so only a global trip is planned. Second, OP has no touring time for each location, which is an important factor affecting the number of POIs visited. Finally, we consider the uncertain traveling time between POIs through the completion probability constraint, which is absent in OP. While most works on OP focus on heuristic approaches [23, 24] to estimate the global optimum of OP, we present an optimal solution to trip recommendation through a prefix based depth-first search strat-

egy with a focus on efficiency through incremental reconstruction and dominance based pruning of routes.

Other works on real life scheduling problem are related to our modeling of the uncertain traveling time. For example, [2] considered multiple types of transport within a single trip and adopted Monte-Carlo simulation to estimate the probability of catching the trip in non-deterministic transport networks. [27] introduced a Bayesian model to estimate the distribution of ambulance traveling time on the road in a city.

9. CONCLUSION AND EXTENSION

We formulated the personalized trip recommendation problem, which is NP-hard, to retrieve a sequence of POIs that maximizes user's satisfaction according to user's historic activities with various constraints including user's time budget, POI availability and uncertain traveling time. We presented both optimal solutions and heuristic solutions to this problem. Our evaluation on real life data sets suggested that PDFS is the most efficient algorithm for optimal solutions and SE-SR improves efficiency at a slightly lower quality than optimal solutions.

Several variations are possible in the presented trip recommendation model. One variation is to factor the touring time of a POI in the happiness score, that is, it is more important for a POI with a longer staying time to be preferred by the user than a POI with a shorter staying time. We can also factor the completion probability of a trip in the score, in addition to a threshold on the probability. Another variation is adding a financial budget constraint of a user, in addition to the time budget, assuming a cost for traveling and a cost for visiting a POI. If the type of a POI is given and if the diversity of POIs is a desirable goal, we can enforce the specific types of POIs in a trip, or a minimum number of different types in a trip, on a feasible state to prune the search space. These variations or extensions require only a minor modification to our current algorithms.

Acknowledgments. This work is partially supported by a Discovery Grant from Natural Sciences and Engineering Research Council of Canada, and partially supported by China Knowledge Centre for Engineering Sciences and Technology (No. CKCEST-2014-1-5). The work of the third author was partially done during a visit to SA Center for Big Data Research hosted in Renmin University of China. This Center is partially funded by a Chinese National 111 Project "Attracting International Talents in Data Engineering and Knowledge Engineering Research".

10. REFERENCES

[1] S. Basu Roy, G. Das, S. Amer-Yahia, and C. Yu. Interactive itinerary planning. In *ICDE*, pages 15–26, 2011.

[2] A. Botea, E. Nikolova, and M. Berlingerio. Multi-modal journey planning in the presence of uncertainty. In *ICAPS*, 2013.

[3] Y.-Y. Chen, A.-J. Cheng, and W. Hsu. Travel recommendation by mining people attributes and travel group types from community-contributed photos. *IEEE Transactions on Multimedia*, 15(6):1283–1295, Oct 2013.

[4] C. Cheng, H. Yang, I. King, and M. R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*, volume 12, pages 17–23, 2012.

[5] C. Cheng, H. Yang, M. R. Lyu, and I. King. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*, pages 2605–2611. AAAI Press, 2013.

[6] Z. Cheng, J. Caverlee, K. Y. Kamath, and K. Lee. Toward traffic-driven location-based web search. In *CIKM*, pages 805–814, 2011.

[7] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 35–44, 2010.

[8] Y. Ge, Q. Liu, H. Xiong, A. Tuzhilin, and J. Chen. Cost-aware travel tour recommendation. In *KDD*, pages 983–991, 2011.

[9] A. Gionis, T. Lappas, K. Pelechrinis, and E. Terzi. Customized tour recommendations in urban areas. In *WSDM*, pages 313–322, 2014.

[10] B. Hu and M. Ester. Spatial topic modeling in online social media for location recommendation. In *RecSys*, pages 25–32, 2013.

[11] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer, 2004.

[12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 30–37, 2009.

[13] T. Kurashima, T. Iwata, T. Hoshide, N. Takaya, and K. Fujimura. Geo topic model: joint modeling of user's activity area and interests for location recommendation. In *WSDM*, pages 375–384, 2013.

[14] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *CIKM*, pages 579–588, 2010.

[15] K. W.-T. Leung, D. L. Lee, and W.-C. Lee. Clr: a collaborative location recommendation framework based on co-clustering. In *SIGIR*, pages 305–314, 2011.

[16] B. Liu, Y. Fu, Z. Yao, and H. Xiong. Learning geographical preferences for point-of-interest recommendation. In *KDD*, pages 1043–1051, 2013.

[17] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong. Personalized travel package recommendation. In *ICDM*, pages 407–416, 2011.

[18] E. H.-C. Lu, C.-Y. Chen, and V. S. Tseng. Personalized trip recommendation with multiple constraints by mining user check-in behaviors. In *SIGSPATIAL*, pages 209–218, 2012.

[19] N. Marlow. A normal limit theorem for power sums of independent random variables. *Bell System Technical Journal*, 46(9):2081–2089, 1967.

[20] M. Qu, H. Zhu, J. Liu, G. Liu, and H. Xiong. A cost-effective recommender system for taxi drivers. In *KDD*, pages 45–54, 2014.

[21] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *NIPS*, pages 1257–1264, 2008.

[22] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[23] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. V. Berghe, and D. V. Oudheusden. A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, 22(10):964–985, 2008.

[24] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, pages 797–809, 1984.

[25] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[26] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.

[27] B. S. Westgate, D. B. Woodard, D. S. Matteson, S. G. Henderson, et al. Travel time estimation for ambulances using bayesian data augmentation. *The Annals of Applied Statistics*, 7(2):1139–1161, 2013.

[28] M. Ye, P. Yin, and W.-C. Lee. Location recommendation for location-based social networks. In *GIS*, pages 458–461, 2010.

[29] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen. Lcars: a location-content-aware recommender system. In *KDD*, pages 221–229, 2013.

[30] H. Yoon, Y. Zheng, X. Xie, and W. Woo. Social itinerary recommendation from user-generated digital trails. *Personal and Ubiquitous Computing*, 16(5):469–484, 2012.

[31] N. B. S. C. Younes Guessousa, Maurice Aron. Estimating travel time distribution under different traffic conditions. *17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain*, 2014.

[32] J. Yuan, Y. Zheng, and X. Xie. Discovering regions of different functions in a city using human mobility and pois. In *KDD*, pages 186–194, 2012.

[33] C. Zhang, K. Wang, E.-p. Lim, Q. Xu, J. Sun, and H. Yu. Are features equally representative? a feature-centric recommendation. In *AAAI*, 2015.

[34] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W.-Y. Ma. Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, 5(1):1–44, 2011.