Trip Recommendation Meets Real-World Constraints: POI Availability, Diversity, and Traveling Time Uncertainty

CHENYI ZHANG*, HONGWEI LIANG*, and KE WANG, Simon Fraser University

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 three 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. In addition, the *diversity* of the POIs included in the trip plays an important role in user's final adoptions. This work presents efficient solutions to personalized trip recommendation by incorporating these constraints and leveraging them to prune the search space. We evaluated the efficiency and effectiveness of our solutions on real-life LBSN datasets.

CCS Concepts: • **Computing methodologies** → **Planning and scheduling** • **Information systems** → *Recommender systems* Data mining • **Theory of computation** → *Dynamic programming;*

Additional Key Words and Phrases: Trip plan, location-based social network, recommender systems

ACM Reference Format:
Chenyi Zhang, Hongwei Liang, and Ke Wang. 2016. Trip recommendation meets real-world constraints: POI availability, diversity, and traveling time uncertainty. ACM Trans. Inf. Syst. 35, 1, Article 5 (September 2016), 28 pages.
DOI: http://dx.doi.org/10.1145/2948065

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 an array of new research problems in both academia and industry, such as user behavior analysis, movement pattern study [Cheng et al. 2013;

*Chenyi Zhang and Hongwei Liang (corresponding authors) are both first authors and contributing equally. This work was primarily done when they were PhD students supervised by Ke Wang.
This work is supported by a Discovery Grant from Natural Sciences and Engineering Research Council of Canada. 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."
This work is an extension of the conference paper [Zhang et al. 2015a]. The main changes are listed in Section 8.2.
Authors' addresses: C. Zhang, R&D Group, Mogujie.com; H. Liang and K. Wang, School of Computing Science, Simon Fraser University; email: {chenyiz, hongweil}@sfu.ca, wangk@cs.sfu.ca.

ACM 1046-8188/2016/09-ART5 $15.00
DOI: http://dx.doi.org/10.1145/2948065

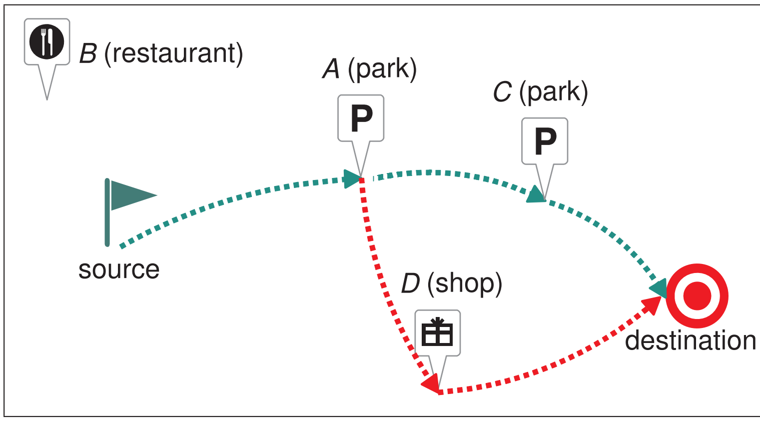


Fig. 1. Example of trip recommendation. Both the trips *source* → *A* → *C* → *destination* and *source* → *A* → *D* → *destination* satisify the spatial and time constraints. However, if the user wants to visit at least two categories of POIs, *source* → *A* → *D* → *destination* is feasible while *source* → *A* → *C* → *destination* cannot meet user's needs.

Leung et al. 2011], and various real-world applications [Cheng et al. 2011; Yuan et al. 2012; Zheng et al. 2011]. Among them, POI recommendation and trip recommendation [Kurashima et al. 2010; Ye et al. 2010] are hot topics and require a location sensitive solution. For example, recommending a highestrated 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 article, 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.

—(*Order and spatial constraints 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. For example, as illustrated in Figure 1, though *A* (a national park) and *B* (a famous restaurant) have the highest scores individually, it is not feasible to visit both *A* and *B*, and end at the specified destination, due to the user's time constraint and the long travel distance between the two. In this case, recommending *A* and *C* (another park) is likely more suitable, even if *C* may have a slightly lower score than *B*.

—(*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.

—(*POI diversity*) Fourth, the diversity of POIs included in a trip also affects user satisfaction since users are usually interested in different types of attractions or expect a variety of activities such that they want to explore multiple categories of POIs during the trip; otherwise, a trip consisting of too few (even single) categories of POIs is boring. The category of POIs could be museum, park, shop, restaurant, and so on. For example, if the user wants to visit at least two categories of POIs, in the case of Figure 1, the trip with the green line cannot meet the user's needs, while the trip with the red line is feasible. Similar arguments are also suggested by a few previous works, such as Ardissono et al. [2003] and Gionis et al. [2014]. However, they either expect the user to manually select a POI from each desired category or assume a fixed order on the categories of POIs, of which the restrictions are too strong. Specifying a minimum number instead of (a fixed order of) exact POI categories is more practical. One important fact is that users often do not have an idea about which categories of POIs to visit before departure. Besides, specifying exact categories does not allow surprise. Actually, specifying exact POI categories is even simpler. The reason is that the user knows the exact categories of POIs to visit in advance so that we can simply filter the POIs belonging to the undesired categories in a pre-processing step and then recommend a trip using the remaining POIs.

—(*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 limit. A strategy that prunes such infeasible and non-optimal trips based on user preferences, POI availability, and traveling time uncertainty is essential for scaling a solution to large applications.

Trip recommendation has been studied recently. Liu et al. [2011] 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. Chen et al. [2013] developed a Bayesian learning model to extract travel paths from photos and conducted personalized travel recommendations 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 [Gionis et al. 2014] 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 categories and the user knows the order of visiting POI categories and likes to visit POIs of each category exactly once in a pre-determined order. The restriction of visiting each category exactly once in a pre-determined order significantly reduces the search space. For example, for 150 POIs falling into five categories 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 article, 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; and (5) the diversity of POIs that constrains the minimum number of POI categories. 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 constraints that all the POIs in the trip can be visited and the trip can be completed within the user time budget with a probability no less than a user specified threshold; moreover, the trip covers a user-specified minimum number of POI categories. This problem is NP-hard, as it is a special case of either the Knapsack problem [Kellerer et al. 2004] or the Orienteering problem [Vansteenwegen et al. 2011].

We solve this problem by using the information and constraints in (1)–(5) to prune unpromising candidate trips. Our algorithm has an offline step and an online step. In the *offline step*, we apply collaborative filtering 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 (PDFS) 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 datasets, Yelp and Foursquare, and demonstrated the superiority over previous trip recommendation algorithms.

1.3. The Road Map

The rest of the article 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 3.2–3.4 presents our modeling of POI availability, uncertain traveling time, and diversity of POIs. 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 sacrifices optimality for efficiency. Section 5 introduces the second optimal solution based on the PDFS 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 article with a discussion on extension to other scenarios of trip recommendation.

2. PRELIMINARY

This section describes our data model and the trip recommendation problem. We begin by summarizing the main notations and their corresponding interpretations to be used throughout the article in Table I for easy reference.

2.1. Data Model

POI map: We assume that there are n POIs in a directed graph $G = (V, E)$. $V = \{1, \dots, n\}$ is a set of POIs. Each POI $i \in V$ is associated with the following information:

Table I. Frequently Used Notations

Notation	Gloss
$G = (V, E)$	POI map G with POI set V and edge set E
m_i	touring time for POI i
$[O_i, C_i]$	opening and closing time for POI i
t_{ij}	traveling time from POI i to j
r_{ui} or r^*_{ui}	observed rating or estimated rating of user u for POI i
$\mathcal{P}$	a trip route
x, y	source and destination location of a trip
T_0	departure time of a trip
b	time budget of a trip
$F(\mathcal{P}, u)$	score of a trip route $\mathcal{P}$ for user u
$\psi(\mathcal{P})$	completion probability of trip route $\mathcal{P}$
θ	completion probability threshold
π_i	expected starting time of visiting POI i
$\mathtt{Sat}(i, j, \pi_i)$	a function to check the accessibility of j followed by visiting i
ϕ_i	category of POI i
β	POI diversity threshold

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 $E[t_{ij}]$, where t_{ij} follows a distribution with probability density function $f_{ij}(\cdot)$ and $E[t_{ij}]$ is its expectation. 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 $x \to \cdots i \cdots \to y$, that starts from x at the time T_0, visits each POI i listed in the route in order, and ends at y. We 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: We have four types of 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. We do not consider "passing by" an intermediate POI i in order to visit the next POI as visiting the POI because it does not need to spend m_i time at i. For this reason, a trip route visits each POI at most once, and all POIs that are passed by but not visited will not be included in a trip route.

—*Time budget constraint:* the whole trip is completed within a period of time b, including traveling time $E[t_{ij}]$ between POIs and touring time m_i 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 $\theta \in [0, 1)$.
—*POI diversity constraint:* an integer specifies that the user wishes to visit at least β different categories of POIs in a trip for touring diversity. Note that the threshold β controls the minimum number of POI categories, not the minimum number of POIs. As is discussed in the fourth point of Section 1.1, this diversity constraint is a more general setting than specifying exact POI categories.

2.2. Problem Definition

PROBLEM 1. *[TripRec] Given a POI map with graph $G = (V, E)$, user u with the source x and the destination y, a departure time T_0, a time budget b, a completion probability threshold $\theta \in [0, 1)$, and a diversity constraint β, we want to find an optimal trip route $\mathcal{P}$ that maximizes user happiness $F(\mathcal{P}, u)$ under the following constraints: (1) it starts at location x and ends at location y; (2) it satisfies the POI availability constraint; (3) it completes within the time budget; (4) it satisfies the completion probability constraint; and (5) it satisfies the POI diversity constraint.*

We show the following simplified decision version of the *TripRec* problem, by ignoring the touring time, the POI availability constraint, the uncertain traveling time, the completion probability constraint and the POI diversity constraint, is NP-complete. In particular, we assume that $T_0 = 0$, $m_i = 0$, $[O_i, C_i] = [0, \infty]$, $r^*_{ui} = 1$ for all POIs $i \in V$ (except the source and destination), $x = y$, the traveling time t_{ij} is a fixed constant, the completion probability threshold $\theta = 0$, and the diversity threshold $\beta = 1$.

The *simplified decision TripRec problem* is: Given a POI map with graph $G = (V, E)$, a user u with the source and destination x, and a time budget b, the task is to decide whether there is any trip route $\mathcal{P}$ such that $F(\mathcal{P}, u) \geq \mathcal{B}$ and the trip is completed within a period of time b.

THEOREM 1. *The simplified decision version of the TripRec problem is NP-complete.*

PROOF. First, we show that the simplified decision TripRec is in NP. This is so because given a trip $\mathcal{P}$, we can verify in polynomial time whether $F(\mathcal{P}, u) \geq \mathcal{B}$ and the trip is completed within a period of time b.

Then, we show that the simplified decision TripRec is NP-hard. For this, we reduce the decision version of the traveling salesman problem (TSP) [Applegate et al. 2007], which is known to be NP-hard, to the simplified decision version of TripRec. The *instance of the decision TSP* is as follows: Given an input graph $G = (V, E)$, consisting of n cities and edges with the weights representing the distance between cities, a distance bound D_{max} and the origin city x, the task is to decide whether there exists any route with length no longer than D_{max} that visits each city exactly once and returns to the origin city.

For an instance of the decision TSP, we can create an instance of the simplified decision TripRec: the graph G is the same as for the decision TSP. Let the traveling time t_{ij} be equal to the weight on the edge from i to j, and let $b = D_{max}$ and $\mathcal{B} = n$.

If the decision TSP is a "yes" instance, the user has a tour of the n cities with length no more than D_{max}. Since $D_{max} = b$ and $\mathcal{B} = n$, this tour is a "yes" instance of the corresponding simplified decision TripRec because $F(\mathcal{P}, u) \geq \mathcal{B}$ and the tour is completed within a period of time b.

On the other hand, if the simplified decision TripRec instance is a "yes" instance, there is a trip route $\mathcal{P}$ such that $F(\mathcal{P}, u) \geq \mathcal{B}$ and the trip is completed within the time budget b. Since $\mathcal{B} = n$ and $b = D_{max}$, these constraints imply that the trip must visit each city at least once and the total length is no more than D_{max}. Since a trip does not visit a node more than once, this trip visits each city exactly once. Thus, the instance of decision TSP must be a "yes" instance. □

Since the simplified decision TripRec is NP-complete, the optimization TripRec problem is at least NP-hard.

In the rest of the article, 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 Zhang et al. [2015b]. 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. We assume that the user may select some features of the POI when she rates it. If the user rates the POI j but does not specifically select any feature of j, it is assumed that all features of j are selected by default. 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 and are rated by u:

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

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 [Koren et al. 2009] to R' to extract the latent user vectors p_u for users u and the latent feature vectors q_f for features f. The goal is to minimize the regularized squared error loss between the rating g_{uf} and the predicted rating $p_u^T q_f$:

$$\Theta = \frac{1}{2}\sum_{u,f} I_{uf}\left(g_{uf} - p_u^T q_f\right)^2 + \frac{\alpha_p}{2}\sum_u \|p_u\|^2 + \frac{\alpha_q}{2}\sum_f \|q_f\|^2, \tag{2}$$

where α_p and α_q are regularization parameters; I_{uf} is a binary indicator that is equal to 1 if g_{uf} is defined, and equal to 0 otherwise. Taking the gradient of E respectively to the variables p_u and q_f, we get

$$\frac{\partial \Theta}{\partial p_u} = \alpha_p p_u - \sum_f \left(g_{uf} - p_u^T q_f\right) q_f \tag{3}$$

$$\frac{\partial \Theta}{\partial q_f} = \alpha_q q_f - \sum_i \left(g_{uf} - p_u^T q_f\right) p_u. \tag{4}$$

A local minimum of E is found by iteratively updating each variable with a step proportional to the negative of the gradient based on the recent values with the learning rate η:

$$p_u^{\kappa+1} = p_u^{\kappa} - \eta \frac{\partial \Theta}{\partial p_u^{\kappa}}, \;\; q_f^{\kappa+1} = q_f^{\kappa} - \eta \frac{\partial \Theta}{\partial q_f^{\kappa}}. \tag{5}$$

This iterative process stops until convergence. The outcome is the latent user vectors p_u for all users u and the latent feature vectors q_f for all features f. The predicted rating of a user u on a feature f is given by the inner product $p_u^T q_f$. 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:

$$r_{ui}^* = agg\big(\{p_u^T q_f | f \in B_i\}\big). \tag{6}$$

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

3.2. Testing Time Constraints

We first assume the traveling time t_{ij} between two POIs is deterministic. Note that there may be no direct edge between two POIs in the graph. To obtain any t_{ij} in advance, we use any existing shortest-path algorithm, such as the Floyd-Warshal algorithm [Floyd 1962], to compute the pair-wise traveling time in a pre-processing step. This is a one-time computation and the results are stored for further usage. It deals with general graphing; even the triangle inequality is not enforced on the graph.

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 $\texttt{Sat}$ 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 $\pi_j + m_j \le C_j$ and $\pi_j + m_j + t_{jy} \le T_0 + b$, 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 [Qu et al. 2014]; 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 the probability density function $f_{ij}(\cdot)$. Let $E[t_{ij}]$ denote the expectation of t_{ij}. 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 $\psi(\mathcal{P}) \ge \theta$. Note that the pre-processing step of computing the shortest paths, as in Section 3.2, is also required, where $E[t_{ij}]$ is the weight of each edge of the graph.

We can modify the above constraint testing function $\texttt{Sat}$ for uncertain traveling time as follows. $\texttt{Sat}(i, j, \pi_i)$ returns *true* if $\pi_j + m_j \le C_j$, $\pi_j + m_j + E[t_{jy}] \le T_0 + b$, and $\psi(\mathcal{P}) \ge \theta$, 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 $\mathcal{P}$. Considering 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} is 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:

$$P(\chi < t) = \int_0^t f_{ij}(\delta) d\delta. \tag{7}$$

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:

$$P(\chi < t) = \iint_D f_{ij}(\delta) f_{jk}(\gamma) d\delta d\gamma, \tag{8}$$

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

$$P(\chi < t) = \iint \cdots \int_D f_{ij}(\delta_1) \cdots f_{j'k}(\delta_c) d\delta_1 d \cdots d\delta_c, \tag{9}$$

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

THEOREM 2 (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 to the route $\mathcal{P}$. 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 Guessous et al. [2014] 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 Westgate et al. [2013]. 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 Marlow [1967], χ can be approximated by another log-normal distribution $\mathcal{LN}(\mu_\chi, \sigma_\chi^2)$ with the following parameters:

$$\begin{aligned} \sigma_\chi^2 &= \log\left(\frac{\sum e^{2\mu_z + \sigma_z^2}(e^{\sigma_z^2} - 1)}{\left(\sum e^{\mu_z + \sigma_z^2/2}\right)^2} + 1\right) \\ \mu_\chi &= \log\left(\sum e^{\mu_z + \sigma_z^2/2}\right) - \frac{\sigma_\chi^2}{2} \end{aligned}. \tag{10}$$

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,

$$P(\chi < t) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{\ln t - \mu_\chi}{\sigma_\chi\sqrt{2}}\right)\right], \tag{11}$$

where $\mathrm{erf}(t) = \frac{2}{\sqrt{\pi}}\int_0^t e^{-\delta^2} d\delta$ [Westgate et al. 2013].

Note that the estimation of the probability of completing a trip, $P(\chi < t)$, is not costly. Because we assume the parameters μ_z and σ_z of the distribution for t_z are given, the estimation of the parameters μ_χ and σ_χ of the other distribution for χ, in Equation (10), takes no time. Then, with the result of μ_χ and σ_χ, the computation of Equation (11) takes no time.

3.4. Modeling POI Categories

In some scenarios, each POI has an explicit category. For example, some tourism websites may categorize POIs into different classes. We can simply apply this category information. However, in most cases, these explicit categories are not available, therefore, requiring us to infer the category information from the contend description of each POI. Let's first discuss how the POI categories are inferred.

We classify the POI categories by Latent Dirichlet Allocation (LDA) [Blei et al. 2003], a basic topic modeling method. The intuition behind LDA is that documents exhibit multiple topics which are represented by distributions over words. Since each POI i associates with a bag of words W_i, we can directly adopt LDA to infer the topic distribution θ_i for each POI i. Refer to Blei et al. [2003] for more details.

θ_i is a probabilistic mixture of latent topics and each dimension $\theta_i(z)$ represents the probability for a certain topic z. Then, the category of POI i, denoted as ϕ_i, is defined as follows:

$$\phi_i = \arg\max_z\{\theta_i(z)\}. \tag{12}$$

The POI diversity constraint requires that a feasible route $\mathcal{P}$ consists of at least β categories of POIs. Namely, $|\bigcup_{i\in\mathcal{P}}\phi_i| \geq \beta$. When $\beta = 1$, every route always satisfies the diversity constraint.

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 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 $s = (K, H, Z, T, \rho, i)$, where K is the set of POIs already visited, excluding x and y, H is the overall happiness collected (i.e., $F(K, u)$), Z is the set of categories covered by the POIs in K, T is the starting time of visiting at i (i.e., π_i), 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_H, s_Z s_T, s_ρ, and s_i, respectively. Initially, there is only one state $s_0 = (\emptyset, 0, \emptyset, T_0, x, x)$, 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 $s = (K, H, Z, T, \rho, i)$ is extended into a new state s' associated with POI $j \notin s_K \cup \{x, y\}$

according to the following rules:

$$\begin{cases} s'_K = s_K \cup \{j\} \\ s'_H = s_H + r^*_{uj} \\ s'_Z = s_Z \cup \{\phi_j\} \\ s'_T = \max\{s_T + m_i + E[t_{ij}], O_j\} \\ s'_\rho = s_\rho \to j \end{cases}. \tag{13}$$

A new state s' is *feasible* if $\texttt{Sat}(i, j, s_T)$ 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

$$(s_i = s'_i) \wedge (s_K = s'_K) \wedge (s_T \leq s'_T) \wedge (\psi(\bar{s_\rho}) \geq \psi(\bar{s'_\rho})), \tag{14}$$

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 $s_H = s'_H$ and $s_Z = s'_Z$, i.e., s and s' give the same user happiness and set of categories. 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'. Please note that the dominance applies also for the case of a "tight", i.e., all the terms in Equation (14) are equations. Thus, in the case of a "tight," the later extended state dominates an earlier extended one. We assume that the procedure Check tests the dominance: $\texttt{Check}(s, s')$ returns *true* if s dominates s' (i.e., Equation (14)) 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 Equation (14) 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 3 (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. Note that it's possible that there are more than one optimal solutions.

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, and the optimal route must fulfil the diversity threshold β. 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. The diversity constraint β is one of the necessary conditions used to check whether a trip is considered to be optimal. If β is not checked while the trips are constructed, we need to first keep any trips who meet the other constraints and rank them in descending order as the overall happiness, and then check the satisfaction of diversity constraint for each kept trip from the top ranked to the bottom ranked in an additional post-processing phase until one trip meeting constraint β is found. In this case, unknown numbers of trips (could be very large) need to be maintained during constructing the trips, which is a huge overhead.

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, diversity threshold β
output: optimal TripRec trip route, $\mathcal{P}$

1 $s_0 \leftarrow (\emptyset, 0, \emptyset, T_0, x, x)$, $s^* \leftarrow s_0$;
2 $S \leftarrow \{s_0\}$, $S' \leftarrow \emptyset$;
3 **while** $S \neq \emptyset$ **do**
4   **for** $s \in S$ **do**
5   **for** $j \in V \backslash s_K$ **do**
6   **if** $\texttt{Sat}(i, j, s_T)$ **then** // $i \equiv s_i$
7                 $s'_T \leftarrow \max\{s_T + m_i + E[t_{ij}], O_j\}$;
8                 $s' \leftarrow (s_K \cup \{j\}, s_H + r^*_{uj}, s_Z \cup \{\phi_j\}, s'_T, s_\rho \rightarrow j, j)$;
9   **if** $\exists s'' \in S' : \texttt{Check}(s', s'') = true$ **then**
10   remove s'' from S';
11   add s' to S';
12   **if** $s'_H > s^*_H$ *and* $|s'_Z| \geq \beta$ **then**
13                     $s^* \leftarrow s'$;
14     $S \leftarrow S'$, $S' \leftarrow \emptyset$;
15 return $s^*_\rho \rightarrow y$ as $\mathcal{P}$

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), $s_H \geq s'_H$ and $s_Z \geq s'_Z$ (i.e., s representing a more preferred route than s'). So, the dominance test condition in Equation (14) is relaxed into

$$(s_i = s'_i) \wedge (|s_K| = |s'_K|) \wedge (s_H \geq s'_H) \wedge (s_Z \geq s'_Z) \wedge (s_T \leq s'_T) \wedge (\psi(\bar{s_\rho}) \geq \psi(\bar{s'_\rho})). \qquad (15)$$

Intuitively, with this relaxed dominance relationship, the route for s takes less time, generates a higher happiness, and covers more POI categories than the route for s',

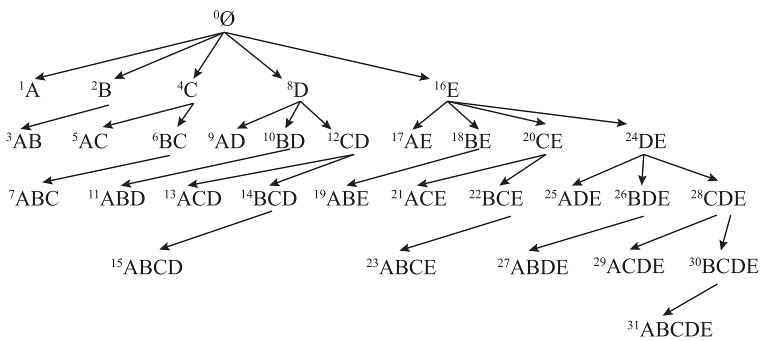


Fig. 2. Prefix-based depth-first enumeration tree.

while reaching the same ending POI i. In other words, the route represented by s gives the user more happiness, more remaining time, and more diversity 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 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 Equation (15) (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 is open 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 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 2 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 ∅. 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 $K\backslash i$ have already been computed at the node with the label $K\backslash i$; 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 four 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., Equation (14)), 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 4. *(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 $K\backslash i$. (2) The node with the label $K\backslash i$ 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 $\rho' = \rho \to i$, where ρ is the prefix containing the POIs in $K\backslash i$. From Theorem 2, ρ must satisfy all the constraints, and from Theorem 3, it suffices to consider only non-dominated feasible route ρ at the node with the label $K\backslash i$. This proves Part (1). Part (2) follows from our discussion on the prefix-based depth-first enumeration. □

This prefix based depth-first search (PDFS) tree is somewhat inspired by the subset-first depth-first (SFDF) tree in Liang et al. [2016], which is used to partition attributes to generate group relationships. In addition to the difference of applications, one major difference is that the PDFS tree is proposed to deal with trips consisting of POIs in different orders; however, the SFDF tree deals with a partition of attributes and the order does not matter. Besides, the ways of materializing each node and the pruning methods used are completely different.

ALGORITHM 2: `PrefixDFS`(U, K)

1 **for** $j \in U$ *in order* **do**
2     $U^- \leftarrow$ prefix of j in U;
3     $K^+ \leftarrow K \cup \{j\}$;
4   **if** $|K^+| = 1$ **then**
5   **if** $\mathtt{Sat}(x, j, T_0) = \mathit{true}$ **then**
6             $\Omega[K^+]_H \leftarrow r^*_{uj}$;
7             $\Omega[K^+]_Z \leftarrow \phi_j$;
8   add $(\max\{E[t_{xj}] + T_0, O_j\}, x \to j, j)$ to $\Omega[K^+]_L$;
9   `PrefixDFS`(U^-,K^+);
10   **else**
11   **for** $k \in K^+$ **do**
12             $K^- \leftarrow K^+ \backslash k$;
13   **if** $\Omega[K^-]$ *is not empty* **then**
14   find $l = (T, \rho, i)$ in $\Omega[K^-]_L$ such that $\mathtt{Sat}(i, k, T) = \mathit{true}$ and $\rho \to k$ is non-dominated;
15   **if** l *is found* **then**
16                     $T' \leftarrow \max\{T + m_i + E[t_{ik}], O_k\}$
17   **if** $\Omega[K^-]_H + r^*_{uk} > \mathcal{H}$ *and* $|\Omega[K^-]_Z \cup \{\phi_k\}| \geq \beta)$ **then**
18   update $\mathcal{P}$ and $\mathcal{H}$;
19                     $\Omega[K^+]_H \leftarrow \Omega[K^-]_H + r^*_{uk}$;
20                     $\Omega[K^+]_Z \leftarrow \Omega[K^-]_Z \cup \{\phi_k\}$;
21   add $(T', \rho \to k, k)$ to $\Omega[K^+]_L$;
22   **if** $\Omega[K^+]$ *is not empty* **then**
23   `PrefixDFS`(U^-,K^+);

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 three components, H, Z and L. H is the total happiness for the POIs in K, and Z is the set of categories covered by the POIs in K. L is a list $(l_1, l_2, \ldots)$, where each entry l in L has the form (T, ρ, i) 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 $\Omega[K]_H$, $\Omega[K]_Z$, and $\Omega[K]_L$ denote the H, Z, and L components of $\Omega[K]$.

We implement the above PDFS 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, the diversity threshold β, 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** 4–9 adds the route $x \to j \to y$ to the hash entry for K^+ and recursively calls `PrefixDFS`(U^-, K^+). If K is not empty, **Lines** 11–21 add all non-dominated feasible routes having the POI set K^+ to the hash entry for K^+. In particular, for each $k \in K^+$, **Line** 14 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 $l = (T, \rho, i)$ for the POI set $K^- = K^+ \backslash k$ and the ending POI k (Theorem 4) such that it satisfies all the constraints, i.e., $\texttt{Sat}(i, k, T) = \mathit{true}$, and the route $\rho \rightarrow k$ is non-dominated (by the conditions in Equation (14)). From Theorem 4, all non-dominated feasible routes for K^- are stored at the node K^- and were computed already. If there exists such an l, $(T', \rho \rightarrow k, k)$ 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 $\texttt{PrefixDFS}(U^-, K^+)$, 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. Let us first ignore the POI diversity constraint, since it is never violated by inserting a POI into the current trip route. We will discuss whether this algorithm can deal with the diversity constraint later. The procedure is illustrated in Algorithm 3. Each calling of `insert` results in one additional POI in the route, until it is impossible to add any new POI into the route without violating the constraints. To avoid the local optimum, we generate some small numbers 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.

ALGORITHM 3: Heuristic Approximation

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

1 initialize the route $\rho : x \rightarrow y$;
2 **repeat**
3     $\rho' \leftarrow \texttt{Insert}(\rho)$;
4     $\rho \leftarrow \rho'$
5 **until** *no more POIs in V can be inserted*;
6 $\mathcal{P}_1 \leftarrow \rho$;
7 remove the POIs in the route $\mathcal{P}_1$ from V;
8 generate another route $\mathcal{P}_2$ by repeating **Line** 2–5;
9 **if** $F(\mathcal{P}_1, u) > F(\mathcal{P}_2, u)$ **then**
10     $\mathcal{P} \leftarrow \mathcal{P}_1$
11 **else**
12     $\mathcal{P} \leftarrow \mathcal{P}_2$

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

$$a_j = \pi_i + m_i + E[t_{ij}], \tag{16}$$

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

$$w_j = \max\{0, O_j - a_j\}. \tag{17}$$

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

$$v_i = \min\{C_i - \pi_i - m_i, w_j + v_j\}, \tag{18}$$

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 Equation (11) 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

$$\varepsilon_k = E[t_{ik}] + w_k + m_k + E[t_{kj}] - E[t_{ij}]. \tag{19}$$

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:

$$\gamma_k = (r^*_{uk})^2 / \varepsilon_k. \tag{20}$$

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 Equations (16)–(18). 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 Equations (16) through (18). For example, Equation (18) requires first updating a later POI before updating an early POI in a route.

Then, we discuss whether the heuristic approximation algorithm can deal with the diversity constraint. The answer is no. In each iteration, the algorithm chooses a POI to insert to maximize the ratio γ_k, as in Equation (20), while satisfying all the other constraints. But this ratio is not conditioned on the POI categories. In some cases, the POIs to be inserted (have the maximum value of γ_k) in each iteration belong to the same category. As a result, the final recommended trip includes very few, even only one category of POIs. This trip probably fails to satisfy the diversity constraint.

7. EXPERIMENTS

This section presents the empirical evaluation of the proposed methods.

7.1. Experimental Setup

We adopt the Yelp[1] and Foursquare[2] datasets in our experiments. Both datasets were previously used for recommendation evaluation in Hu and Ester [2013]. The Yelp dataset contains 45,981 users; 229,906 ratings of 1–5 scales; 11,537 POIs; plus, text reviews on POIs. We pre-processed 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 dataset 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 pre-processing 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 $8am$. The expected traveling time $E[t_{ij}]$ for a pair of POIs (i, j) is estimated using Google Maps[3] with the driving mode. We assume that Google Maps produces shortest paths between POIs; therefore, the pre-processing step of computing the shortest paths, as stated in Section 3.2, is unnecessary. All the experiments were run on a PC with 2.53GHz Quad-Core CPU and 12G memory.

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 datasets, 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 [Shani and Gunawardana 2011], we use the standard root mean squared error (RMSE) and mean absolute error (MAE) as the accuracy metrics for POI recommendation. These two metrics are defined as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(r_{ui} - r^*_{ui})^2} \tag{21}$$

$$MAE = \frac{1}{N}\sum_{u,i}|r_{ui} - r^*_{ui}|, \tag{22}$$

where r_{ui} is the true rating value, r^*_{ui} is the predicted rating value, and N is the number of ratings in the testing set. The smaller these values are, the better the result is.

Many POI recommendation approaches are based on topic modeling; for example, STM [Hu and Ester 2013] and LCA [Yin et al. 2013] 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 (PMF): This is the classic matrix factorization on the user-item rating utility matrix [Salakhutdinov and Mnih 2008] where POIs are

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

Table II. 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**
improvement of FCF over PMF	7.7%	7.3%	16.8%	14.7%
improvement of FCF over CTR	5.4%	5.4%	14.1%	12.2%

Table III. Paired t-Test (2-Tail) of FCF and Baselines

	Yelp		Foursquare	
Method	RMSE	MAE	RMSE	MAE
FCF/PMF	7.4×10^{-6}	4.2×10^{-6}	6.7×10^{-7}	2.1×10^{-6}
FCF/CTR	1.1×10^{-5}	1.2×10^{-5}	2.0×10^{-6}	3.8×10^{-6}

treated as items. In PMF, matrix factorization is generalized as a probabilistic model where a latent user vector $p_u \sim \mathcal{N}(0, \alpha_p^{-1} I_D)$ and 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 Salakhutdinov and Mnih [2008] and set $D = 10$, the dimensionality of user and item latent factors.

Collaborative topic regression (CTR): This is the matrix factorization with topic modeling applied to the content of items described in Wang and Blei [2011]. 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 Wang and Blei [2011] and set $D = 10$.

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

Table II shows the results of accuracy of the above three methods. FCF achieves the best performance and has a significant improvement in terms of RMSE/MAE on both data sets. The improvement of FCF against any baseline on RMSE is measured by the following equation $(RMSE(baseline) - RMSE(FCF))/RMSE(baseline) * 100\%$. The improvement on MAE is computed in the same way. So, we believe that the estimated rating by FCF is closer to the true rating.

t-Test. To further verify the statistical significance of the improvement introduced by FCF, we conducted the paired t-Test (2-tail) on FCF and the two baselines. Table III shows that all p-values in the t-Test results are less than 0.01, which suggests that the improvement of FCF over PMF and CTR is statistically significant. In the rest of the experiments, we study the performance of trip recommendations with the estimated rating r^*_{ui} being generated by FCF.

7.3. The Fixed Traveling Time Model without Diversity Constraint

In this section, we evaluate the trip route $\mathcal{P}$ found by TripRec under the fixed traveling time model without considering the diversity constraint, where the traveling time t_{ij} for a pair of POIs i and j is fixed and the diversity threshold is set as a special case $\beta = 1$. The reason we use this deterministic setting is that all the baselines consider fixed traveling time and none of them has a constraint on the minimum number of categories covered by the trip. In this deterministic setting, a feasible route always satisfies the completion probability constraint and the diversity constraint. The model for uncertain traveling time will be considered in Section 7.4, and the effect of diversity constraint is shown in Section 7.5.

We focus on three major cities for trip planning—Phoenix (PX) in Yelp, and 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 Basu Roy et al. [2011]. Even with this restriction, the number of trips that consist of five 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, learned by FCF as input.

Greedy algorithm (Greedy): This is the greedy algorithm from the operation research literature [Tsiligirides 1984], 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 Tsiligirides [1984].

Dynamic programming (DP): This is the dynamic programming approach proposed in Gionis et al. [2014]. We adapt to the order constraint by setting a "global" category to each POI and fix the visiting order that is from "global" category to "global" category. However, by filling up a 2-dimensional array, the dynamic programming [Gionis et al. 2014] still cannot deal with the POI availability constraint.

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

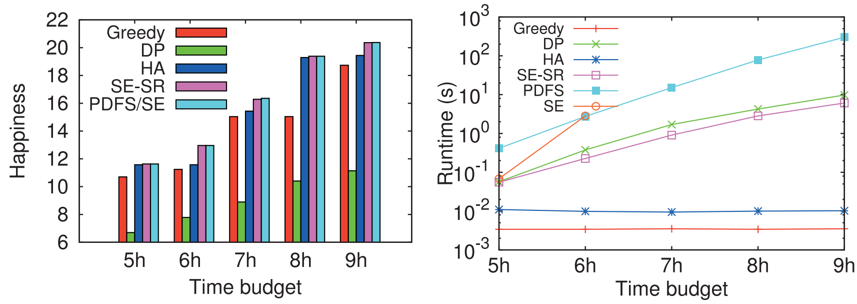
State expansion (SE): This is Algorithm 1 proposed in Section 4. Let SE-SR denote SE with state relaxing. SE guarantees the optimality of solution, but SE-SR does not.

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

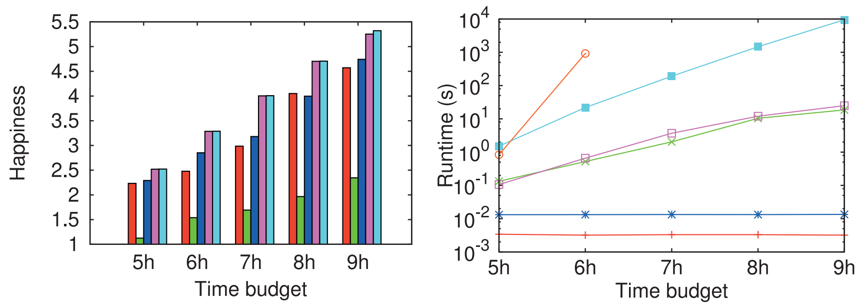
7.3.1. User Happiness. Let us recall that the happiness of user u with respect to route $\mathcal{P}$ is defined as $F(\mathcal{P}, u) = \sum_{i \in \mathcal{P}} r_{ui}^*$, which sums up the estimated ratings r_{ui}^* for all POIs in route $\mathcal{P}$. Figure 3 (left column) presents the user happiness 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 three to seven 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. SE-SR appears close to the optimal solutions because we select the top 150 POIs for each user and the ratings on these 150 POIs likely have minor differences; therefore, in many cases, the happiness for the trips consisting of the same number of POIs and the same ending POI are close.

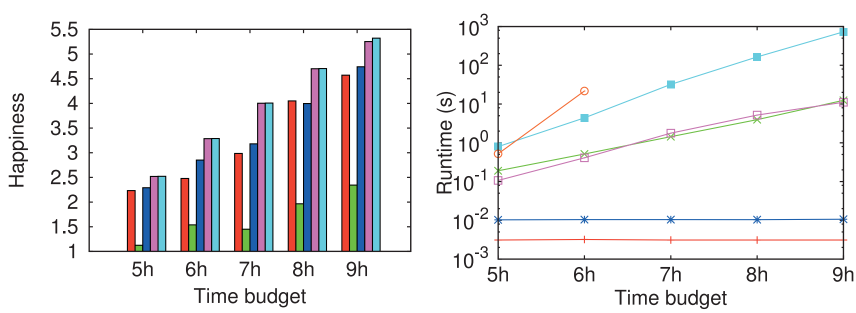
HA performs in the third place and there is an obvious gap between HA and the best two. This is because HA only maintains one route during search, which makes it easy to fall into a local optimum. We will further explain this in the case study below. Greedy performs about 10% worse than HA, as its search 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.



(a) PX



(b) NY



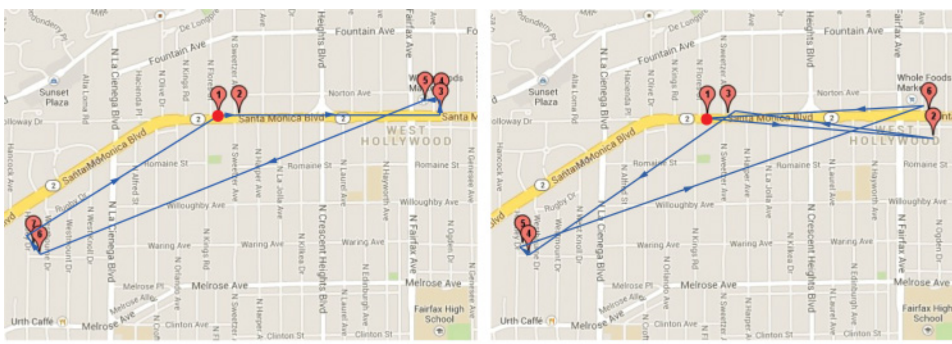
(c) LA

Fig. 3. 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).

7.3.2. Runtime. Figure 3 (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 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.

Table IV. Pruning Power of the SE/PDFS Algorithms in Various Time Budget Settings, the Numbers are the Percentage of the Pruned States by the Algorithms

	ι		
b	PX	NY	LA
5h	99.30%	99.29%	99.25%
6h	99.23%	99.26%	99.20%
7h	99.20%	99.22%	99.17%
8h	99.18%	99.21%	99.15%
9h	99.16%	99.19%	99.11%



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

Fig. 4. Case study of recommended trips for LA. The number in bracket is the happiness of the trip.

7.3.3. Pruning Power. The algorithms' SE and PDFS apply the same pruning strategy, i.e., dominance-based pruning as in Theorem 4, to reduce the potential state number when computing the optimal trip. In this section, we investigate how powerful the pruning strategy is. We take the SE algorithm as an example for illustration. Let S_{all} denote all the states that can be enumerated using the POI set in each iteration, i.e., the enumeration space, and let S_{kept} denote the set of the states that are feasible (by checking whether the function `Sat()` returns true) and are non-dominated. Explicitly, S_{all} represents the states enumerated by **Line 3–Line 5** in Algorithm 1, and S_{kept} represents only the states that go deeper and pass the check of **Line 9**. Then, the pruning power, ι, is defined as:

$$\iota = \frac{|S_{all}| - |S_{kept}|}{|S_{all}|} \times 100\%. \tag{23}$$

Table IV shows the experimental study of the pruning power over the three datasets. As we can see from the results, for all the datasets, over 99% of the states are pruned by our SE/PDFS algorithm; in other words, only around 0.8% of the states are kept. Therefore, the search space and runtime of searching the optimal solution is being greatly reduced. Another fact we can observe is that as the time budget (b) increases, the pruning power becomes weaker. For instance, for the PX dataset, $\iota = 99.16\%$ when $b = 9h$, which is smaller than $\iota = 99.30\%$ when $b = 5h$. The reason is that a longer time budget allows for more POIs being visited in a trip and more trip routes become feasible and cannot be dominated owing to the longer budget.

7.3.4. Case Study. For a randomly selected user with $b = 8h$, Figure 4 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 the fact 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 4(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 tradeoff for efficiency at a slightly lower happiness than the optimal solution; HA is very efficient, but sometimes has a significantly lower happiness. Overall, PDFS and SE-SR are the two best performers considering both quality and efficiency.

7.4. The Uncertain Traveling Time Model

In this section, we study the effect of the uncertain traveling time on SE-SR and PDFS. The diversity threshold, β, is still set as 1. 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., $\sigma_{ij} \sim \mathcal{U}(0.5, 2)$.

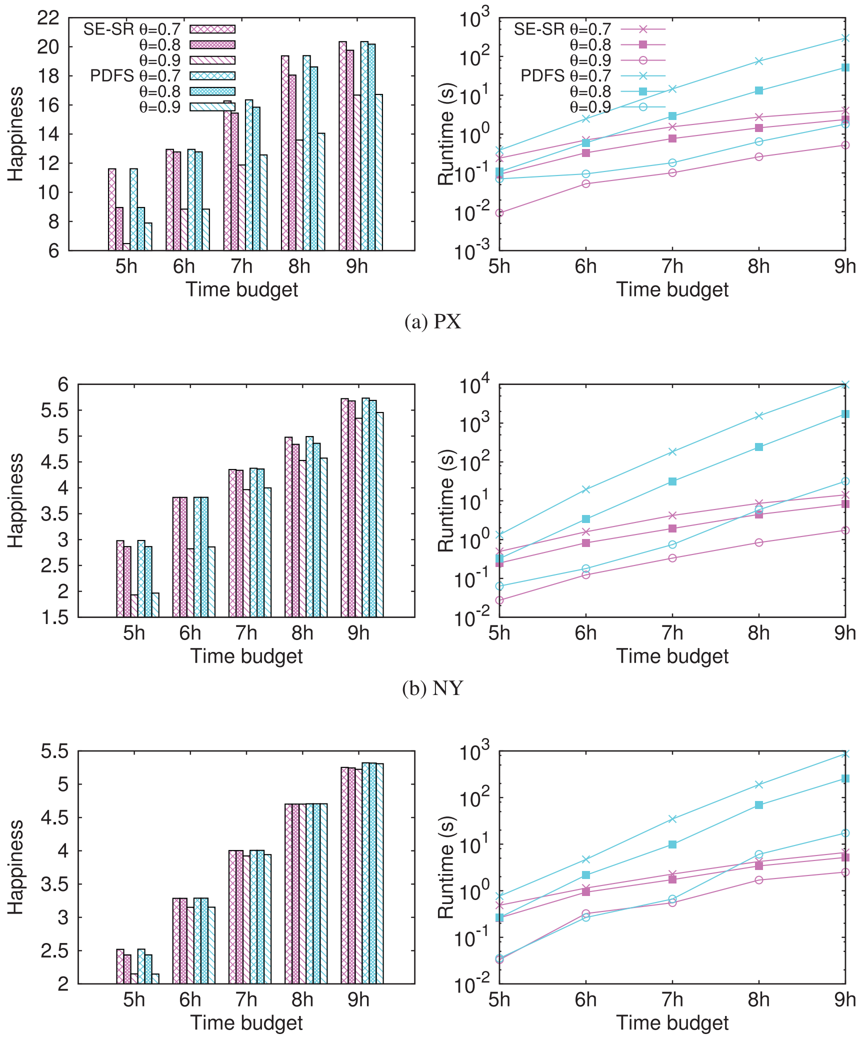
Figure 5 (left column) presents the happiness scores of SE-SR and PDFS with various threshold θ on completion probability. A color represents a method and a pattern represents a threshold θ on completion probability. Compared to the case for the fixed traveling time in Section 7.3, the happiness of both methods becomes lower given the same time budget. For example, there is about a 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 5 (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 3 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 $\theta = 0.9$.

7.5. Effect of Diversity Constraint

The previous sections evaluated the methods for TripRec in the special case of $\beta = 1$, where β is the minimum number of categories of POIs in a trip, specified by the diversity constraint. In this section, we study the effect of the diversity constraint by comparing the results for the special case of $\beta = 1$ and the more general case of $\beta > 1$. The fixed travelling time model is applied. We consider the results of PDFS only because SE generates the same results. Greedy, DP, and HA cannot deal with the diversity constraint. As there are a total of five different categories of POIs, we vary β from two to four.

Table V shows the averaged happiness values of PDFS for both the case of $\beta = 1$ and $\beta > 1$. The numbers in brackets indicate, among the recommended routes for the



(a) PX

(b) NY

(c) LA

Fig. 5. 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).

100 testing users with the setting $\beta = 1$, how many routes also satisfy the specified diversity constraint, with β equal to 2, 3, or 4. Note that if a route is optimal with the setting $\beta = 1$, but it actually contains k ($k > 1$) categories, then the route is also optimal for the case of $\beta = k$. For example, for PX with $\beta = 1$, the entry 16.35 (73) means that the averaged happiness of the 100 routes that have at least 1 category (i.e., $\beta = 1$) is 16.35, and out of them, 73 consist of 2 or more categories of POIs. As we can see in Table V, for $\beta = 2$, there are many overlapped routes for $\beta = 1$ and $\beta = 2$, since a route has a high chance to contain two categories of POIs. As β increases, there exists fewer overlapped routes, so more gap on the happiness can be observed. At $b = 5h$, no route satisfies the diversity constraint of $\beta = 4$, since at most three POIs can be visited within 5 hours.

Table V. Comparison of the Averaged Happiness Over the 100 Testing Users for the Case of $\beta = 1$ and $\beta > 1$, Gained from the Optimal Routes Recommended by PDFS

	PX		NY		LA	
b	$\beta = 1$	$\beta = 2$	$\beta = 1$	$\beta = 2$	$\beta = 1$	$\beta = 2$
5h	11.63 (100)	11.63	2.98 (99)	2.98	2.50 (55)	2.49
6h	12.96 (100)	12.96	3.81 (99)	3.81	3.26 (97)	3.25
7h	16.35 (73)	16.32	4.37 (96)	4.35	3.97 (97)	3.96
8h	19.39 (99)	19.38	4.99 (99)	4.99	4.66 (100)	4.66
9h	20.36 (100)	20.36	5.73 (100)	5.73	5.26 (100)	5.26
	PX		NY		LA	
b	$\beta = 1$	$\beta = 3$	$\beta = 1$	$\beta = 3$	$\beta = 1$	$\beta = 3$
5h	11.63 (38)	11.61	2.98 (8)	2.91	2.50 (11)	2.47
6h	12.96 (26)	12.93	3.81 (71)	3.79	3.26 (4)	3.19
7h	16.35 (38)	16.30	4.37 (84)	4.34	3.97 (5)	3.92
8h	19.39 (97)	19.38	4.99 (6)	4.94	4.66 (13)	4.62
9h	20.36 (89)	20.35	5.73 (78)	5.72	5.26 (96)	5.23
	PX		NY		LA	
b	$\beta = 1$	$\beta = 4$	$\beta = 1$	$\beta = 4$	$\beta = 1$	$\beta = 4$
5h	11.63 (0)	-	2.98 (0)	-	2.50 (0)	-
6h	12.96 (4)	12.17	3.81 (0)	3.77	3.26 (1)	3.19
7h	16.35 (0)	15.51	4.37 (3)	3.91	3.97 (1)	3.91
8h	19.39 (40)	19.37	4.99 (0)	4.83	4.66 (4)	4.59
9h	20.36 (19)	20.33	5.73 (23)	5.69	5.26 (38)	5.19

The numbers in brackets indicate that among the 100 generated optimal results with the constraint $\beta = 1$ for the testing users how many are also optimal with the constraint β Equal to 2, 3, or 4.

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 Cheng et al. [2012] and Kurashima et al. [2013], which consider no content information, and Hu and Ester [2013], Liu et al. [2013], and Yin et al. [2013], 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. Ge et al. [2011] and Liu et al. [2011] developed several probabilistic models to generate possible packages by considering cost, season, area, and so on. A major difference between trip recommendation and travel package recommendation is that travel package recommendation considers neither the order of visiting POIs nor the realistic constraints, such as travelling time between two location and time budget. Kurashima et al. [2010] and Yoon et al. [2012] applied spatio-temporal streams, such as tagged photo streams or GPS trajectories, to seek for a feasible path by topic modeling or Markov models. Basu Roy et al. [2011] used user feedback to improve results on interactive tour recommendation. None of these works focuses on the objective to maximize user's happiness.

De Choudhury et al. [2010], Gionis et al. [2014], and Lu et al. [2012] maximized user's happiness in travel recommendations. Gionis et al. [2014] assumes a fixed order on the categories of POIs visited, which is not suitable in the presence of constraints such as opening hours of POIs. The greedy algorithm proposed in De Choudhury et al. [2010] cannot guarantee the global optimality. Lu et al. [2012] adopted memory-based collaborative filtering to estimate user-specific preferences, which are dynamic with time. All these works do not consider the POI availability and uncertain time constraints. The first two works do not consider user specific preferences, thus, generate the same itinerary to all users. The POI diversity or multiple categories of POIs is considered by Ardissono et al. [2003] and Gionis et al. [2014]. However, Ardissono et al. [2003] expect the user to manually select a POI from each desired category and Gionis et al. [2014] assumes a fixed order on the categories of POIs. A discussion about whether to specify a minimum number of POI categories or to specify exact POI categories is presented in the fourth point of Section 1.1.

This article is primarily extended from the previous work [Zhang et al. 2015a]. One major addition is that the POI diversity constraint is considered in this work by allowing users to specify a threshold on the minimum number of categories of POIs that the recommended trip route covers. In particular, the motivation of the POI diversity, the methods used to classify POI categories, and the empirical evaluation of the POI diversity constraint are presented. Besides, more details, such as formal proof of the NP-hardness of the problem and experiment studies about the pruning power are included in this extension.

8.3. Operation Research and Scheduling

The orienteering problem (OP) [Golden et al. 1987; Vansteenwegen et al. 2011] 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 [Souffriau et al. 2008; Tsiligirides 1984] to estimate the global optimum of OP, we present an optimal solution to trip recommendation through a PDFS strategy with a focus on efficiency through incremental reconstruction and dominance-based pruning of routes.

Other works on real-life scheduling problems are related to our modeling of the uncertain traveling time. For example, Botea et al. [2013] considered multiple types of transport within a single trip and adopted the Monte-Carlo simulation to estimate the probability of catching the trip in non-deterministic transport networks. Westgate et al. [2013] 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 diversity, and uncertain traveling time. We presented both optimal solutions and heuristic solutions to this problem. Our evaluation on real-life datasets 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. Besides, the opening/closing hours can also depend on the day of the week. This can be done by simply using the opening/closing hours at the time when the POI is visited. In addition to the uncertainty of traveling time, we can also model the uncertainty of the POI touring time via a separate random variable by the same way we did for the traveling time t_{ij}, then can merge the uncertainty of the POI touring time and the uncertainty of the travelling time together to estimate the completion probability of the whole trip. These variations or extensions require only a minor modification to our current algorithms.

REFERENCES

David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. 2007. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press.

Liliana Ardissono, Anna Goy, Giovanna Petrone, Marino Segnan, and Pietro Torasso. 2003. Intrigue: Personalized recommendation of tourist attractions for desktop and hand held devices. *Applied Artificial Intelligence* 17, 8–9 (2003), 687–714.

Senjuti Basu Roy, Gautam Das, Sihem Amer-Yahia, and Cong Yu. 2011. Interactive itinerary planning. In *ICDE*. 15–26.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research* 3 (2003), 993–1022.

Adi Botea, Evdokia Nikolova, and Michele Berlingerio. 2013. Multi-modal journey planning in the presence of uncertainty. In *ICAPS*.

Yan-Ying Chen, An-Jung Cheng, and W. H. Hsu. 2013. Travel recommendation by mining people attributes and travel group types from community-contributed photos. *IEEE Transactions on Multimedia* 15, 6 (Oct 2013), 1283–1295.

Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. 2012. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*, Vol. 12. 17–23.

Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*. AAAI Press, 2605–2611.

Zhiyuan Cheng, James Caverlee, Krishna Yeswanth Kamath, and Kyumin Lee. 2011. Toward traffic-driven location-based web search. In *CIKM*. 805–814.

Munmun De Choudhury, Moran Feldman, Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel, and Cong Yu. 2010. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*. 35–44.

Robert W. Floyd. 1962. Algorithm 97: Shortest path. *Communications of the ACM* 5, 6 (1962), 345.

Yong Ge, Qi Liu, Hui Xiong, Alexander Tuzhilin, and Jian Chen. 2011. Cost-aware travel tour recommendation. In *KDD*. 983–991.

Aristides Gionis, Theodoros Lappas, Konstantinos Pelechrinis, and Evimaria Terzi. 2014. Customized tour recommendations in urban areas. In *WSDM*. 313–322.

Bruce L. Golden, Larry Levy, and Rakesh Vohra. 1987. The orienteering problem. *Naval Research Logistics* 34, 3 (1987), 307–318.

Younes Guessous, Maurice Aron, Neila Bhouri, and Simon Cohen. 2014. Estimating travel time distribution under different traffic conditions. *Transportation Research Procedia* 3 (2014), 339–348.

Bo Hu and Martin Ester. 2013. Spatial topic modeling in online social media for location recommendation. In *RecSys*. 25–32.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack Problems*. Springer.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009), 30–37.

Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. 2013. Geo topic model: Joint modeling of user's activity area and interests for location recommendation. In *WSDM*. 375–384.

Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. 2010. Travel route recommendation using geotags in photo sharing sites. In *CIKM*. 579–588.

Kenneth Wai-Ting Leung, Dik Lun Lee, and Wang-Chien Lee. 2011. CLR: A collaborative location recommendation framework based on co-clustering. In *SIGIR*. 305–314.

Hongwei Liang, Ke Wang, and Feida Zhu. 2016. Mining social ties beyond homophily. In *Proceedings of the 32nd IEEE International Conference on Data Engineering (ICDE)*. IEEE.

Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *KDD*. 1043–1051.

Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. 2011. Personalized travel package recommendation. In *ICDM*. 407–416.

Eric Hsueh-Chan Lu, Ching-Yu Chen, and Vincent S. Tseng. 2012. Personalized trip recommendation with multiple constraints by mining user check-in behaviors. In *SIGSPATIAL*. 209–218.

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

Meng Qu, Hengshu Zhu, Junming Liu, Guannan Liu, and Hui Xiong. 2014. A cost-effective recommender system for taxi drivers. In *KDD*. 45–54.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Probabilistic matrix factorization. *NIPS* (2008), 1257–1264.

Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender Systems Handbook*. Springer, 257–297.

Wouter Souffriau, Pieter Vansteenwegen, Joris Vertommen, Greet Vanden Berghe, and Dirk Van Oudheusden. 2008. A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence* 22, 10 (2008), 964–985.

Theodore Tsiligirides. 1984. Heuristic methods applied to orienteering. *Journal of the Operational Research Society* (1984), 797–809.

Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209, 1 (2011), 1–10.

Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD*. 448–456.

Bradford S. Westgate, Dawn B. Woodard, David S. Matteson, Shane G. Henderson, and others. 2013. Travel time estimation for ambulances using Bayesian data augmentation. *The Annals of Applied Statistics* 7, 2 (2013), 1139–1161.

Mao Ye, Peifeng Yin, and Wang-Chien Lee. 2010. Location recommendation for location-based social networks. In *GIS*. 458–461.

Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. 2013. LCARS: A location-content-aware recommender system. In *KDD*. 221–229.

Hyoseok Yoon, Yu Zheng, Xing Xie, and Woontack Woo. 2012. Social itinerary recommendation from user-generated digital trails. *Personal and Ubiquitous Computing* 16, 5 (2012), 469–484.

Jing Yuan, Yu Zheng, and Xing Xie. 2012. Discovering regions of different functions in a city using human mobility and POIs. In *KDD*. 186–194.

Chenyi Zhang, Hongwei Liang, Ke Wang, and Jianling Sun. 2015a. Personalized trip recommendation with POI availability and uncertain traveling time. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 911–920.

Chenyi Zhang, Ke Wang, Ee-peng Lim, Qinneng Xu, Jianling Sun, and Hongkun Yu. 2015b. Are features equally representative? A feature-centric recommendation. In *AAAI*.

Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. 2011. Recommending friends and locations based on individual location history. *ACM Transactions on the Web* 5, 1 (2011), 1–44.

Received December 2015; revised April 2016; accepted May 2016