Measuring and Recommending Time-Sensitive Routes from Location-Based Data

HSUN-PING HSIEH, CHENG-TE LI, and SHOU-DE LIN, National Taiwan University

Location-based services allow users to perform geospatial recording actions, which facilitates the mining of the moving activities of human beings. This article proposes to recommend time-sensitive trip routes consisting of a sequence of locations with associated timestamps based on knowledge extracted from large-scale timestamped location sequence data (e.g., check-ins and GPS traces). We argue that a good route should consider (a) the popularity of places, (b) the visiting order of places, (c) the proper visiting time of each place, and (d) the proper transit time from one place to another. By devising a statistical model, we integrate these four factors into a route goodness function that aims to measure the quality of a route. Equipped with the route goodness, we recommend time-sensitive routes for two scenarios. The first is about constructing the route based on the user-specified source location with the starting time. The second is about composing the route between the specified source location and the destination location given a starting time. To handle these queries, we propose a search method, *Guidance Search*, which consists of a novel heuristic satisfaction function that guides the search toward the destination location and a backward checking mechanism to boost the effectiveness of the constructed route. Experiments on the Gowalla check-in datasets demonstrate the effectiveness of our model on detecting real routes and performing cloze test of routes, comparing with other baseline methods. We also develop a system *TripRouter* as a real-time demo platform.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining, web mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Time-sensitive route, trip recommendation, location-based data

ACM Reference Format:
Hsun-Ping Hsieh, Cheng-Te Li, and Shou-De Lin. 2014. Measuring and recommending time-sensitive routes from location-based data. ACM Trans. Intell. Syst. Technol. 5, 3, Article 45 (June 2014), 27 pages.
DOI: http://dx.doi.org/10.1145/2542668

1. INTRODUCTION

Location-Based Services (LBS), such as Foursquare and Gowalla, allow users to perform the action of location recording that pins the geographical information of current locations and timestamps onto their personal pages. By continuously recording such actions by users, a large-scale location sequences dataset can be generated. The rapid accumulation of location sequence data can not only collectively represent real-world human activities, but can also serve as a handy resource for constructing location-based recommendation systems. Since the user-moving records implicitly reveal how people

Authors' addresses: H.-P. Hsieh, Graduate Institute of Networking and Multimedia, National Taiwan University; email: d98944006@ntu.edu.tw; C.-T. Li, Research Center for Information Technology Innovation, Academia Sinica; email: ctli@citi.sinica.edu.tw; S.-D. Lin, Department of Computer Science and Information Engineering, National Taiwan University; email: sdlin@csie.ntu.edu.tw.

DOI: http://dx.doi.org/10.1145/2542668

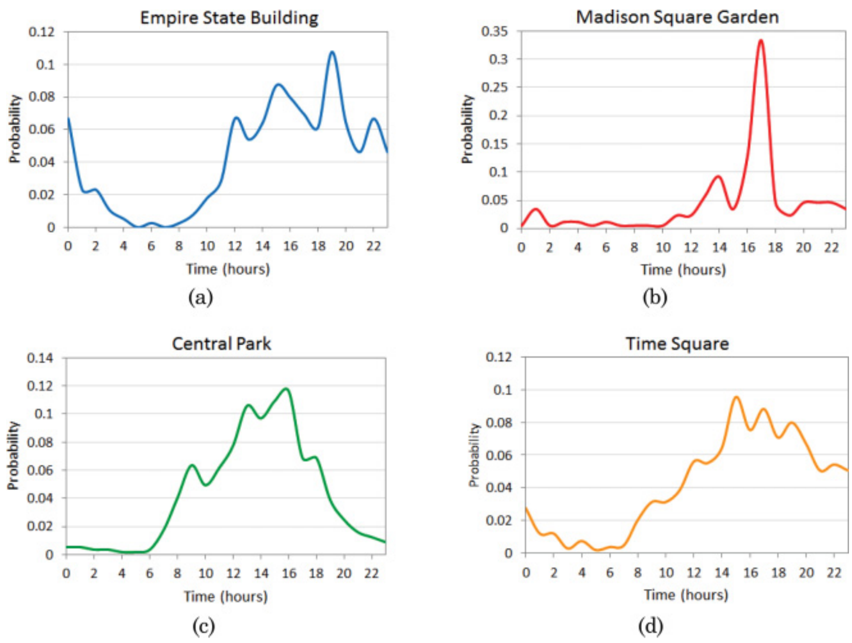


Fig. 1. The distribution of the visiting probability at each time unit (hour) for (a) Empire State Building, (b) Madison Square Garden, (c) Central Park, and (d) Time Square. These distributions are derived from Gowalla check-in data.

travel around an area, displaying rich spatial and temporal information including longitude, latitude, and recording timestamp, one reasonable application leveraging such user-generated location sequence data is to recommend travel routes. Indeed, many of existing works recommend and construct travel routes using GPS trajectories [Chen et al. 2011; Wei et al. 2012] or geo-tagged photos [Arase et al. 2010; Cheng et al. 2011; Yin et al. 2011].

In this article, instead of relying on past moving trajectories to recommend traveling paths, we propose a novel time-sensitive trip route recommendation framework using the timestamped route data. To do so, we argue that a good route recommendation system should consider the following factors when constructing a route:

—**The popularity of a place.** Popular landmarks by definition should attract more visitors.
—**The proper time to visit a place.** In general, the pleasure of visiting a place can be significantly diminished if one arrives at the wrong time. Some places have a wide range of preferred visiting times while others are constrained to certain particular time slots. For example, most people do not want to visit a beach during boiling hot noon, but rather will arrive in the late afternoon to enjoy the sunset scene. Or certain sports events usually take place at a particular time (e.g., in the evening). As shown in Figure 1, derived from the Gowalla check-in data described in Section 5, visitors visit some places with higher probability during certain time slots. For example, (a) people usually visit the Empire State Building from about noon to midnight (note that this place is famous for its night view), (b) people tend to visit Madison Square Garden in the early evening for a basketball game, (c) the proper time to visit Central Park is during the daytime, and (d) Time Square is preferred from afternoon to midnight.

—**The amount of time required to transit from one place to another.** The transit time between places is critical to the design of a suitable route. To recommend the next place to visit with the proper visiting time, we should consider the amount of time spent on traveling. For example, if one has bought tickets to a football game at a stadium 2 hours away, then he or she shall logically choose to start traveling toward the stadium 2 hours ahead of the official kick-off time instead of first going to a nearby museum 30 minutes away.
—**The visiting order of places.** The visiting order of places is highly correlated to the nature of places and human preference, and it can affect the quality of the recommended trip to the user. For example, going to the gym first then going to nearby restaurant for dinner might be a better plan than the other way around because it might be unhealthy to exercise right after a meal.

While some places are extremely sensitive to visiting time, others (e.g., movie theaters) might not possess such strict constraints. An intelligent route recommendation system should consider such diversity and be able to create a route that has a high chance of satisfying users' needs. This article argues that by exploring the timestamped location sequence dataset, it is possible to design a statistical model to achieve such a goal.

We employ the timestamped location sequence data for measuring and constructing time-sensitive routes. Currently, there are two major resources that can provide such location sequence data. One is user moving trajectories from GPS trace recorders (with famous spots annotated), whereas the other is the online check-in data. Although from this point on we assume it is the check-in data that are used, our model can in fact be trained to use any given timestamped route dataset.

In our experiment, we use online check-in data from LBS to acquire the timestamped route information to train our model. The online check-in data provide plenty of explicit or implicit information that allows us to fulfill the above-mentioned requirements for planning a proper trip route. First, we can distill from the check-in data the number of people who have visited a certain place and thus derive the popularity of that place. Second, users in LBS tend to perform check-in actions to keep track of their trips. As a result, we can obtain and consider the visiting order of places. Third, the check-in records contain the visiting timestamps of locations. Users in LBS are able to collectively reveal the proper visiting time of places. Fourth, followed by the check-in timestamps from existing routes, we are able to hypothesize the transit time between places. Equipped with such elements, we utilize the check-in data to recommend trip routes. Figure 2 is an example outcome of our time-sensitive trip route recommendation. Assume that, at 10:00 AM, a certain user starts to travel from his hotel, marked with a star in Figure 2. According to the distribution of the locations to be visited at each time unit in Figure 1, a possible trip route consists of going to Central Park first, followed by the Empire State Building, Madison Square Garden, and finally Times Square.

Formally, the goal of this article is to construct a time-sensitive route from the timestamped location sequence data. We propose to tackle two real-world demands of recommending time-sensitive routes. The first is to construct a route given a *source* location, and the second is to create a route given the *source-destination* pair of locations. Both queries consider the starting time of the trip. Given a query, our model finds a sequence of recommended places in which each location can be visited at a proper time with a reasonable transit time from one place to another in the route. A time-sensitive route is supposed to be more effective than a simple route without timestamp because it allows the users to better manage their time during the trip.

We propose statistical approaches to construct the time-sensitive routes with respect to the proposed queries. In general, our model consists of two phases. In phase

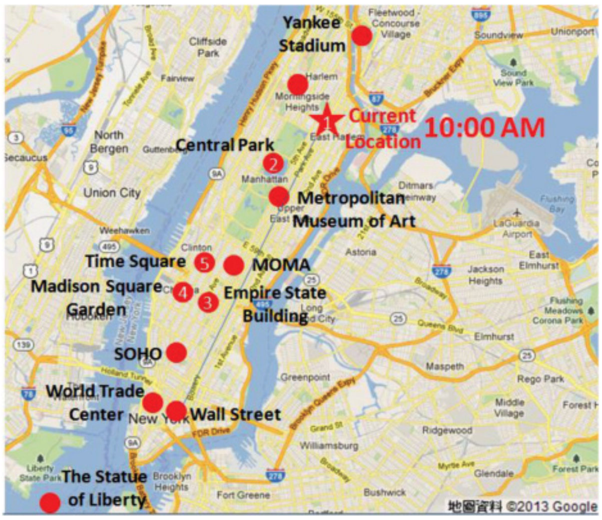


Fig. 2. An example of a recommended trip route starting from a given position at 10:00 AM, in Manhattan, New York City.

one, the quality of a route is measured using a goodness function that integrates the above-mentioned four requirements for a proper trip route. In phase two, given the user-specified query, we design an effective and efficient search method, the Guidance Search, to identify the places to be visited by optimizing the route goodness function.

We summarize the contributions of this article as such:

—We propose a novel time-sensitive trip route recommendation problem. We fulfill the idea by developing a *TripRouter* system based on real-world Gowalla check-in data.
—Conceptually, we propose that a good route should consider four elements: (a) the popularity of a place, (b) the visiting order of places, (c) the proper visiting time of a place, and (d) the proper transit time between places.
—Technically, we devise a goodness function to measure the quality of a route. By exploiting certain statistical methods, we model the four requirements into the design of the goodness function.
—We propose two time-sensitive query scenarios about route recommendation. To deal with these queries, we develop the *Guidance Search* algorithm to construct the route by optimizing the goodness function.

This article is organized as follows, with the accompaniment of the flowchart shown in Figure 3. The input and output of our route recommendation are the Time-Sensitive Query and the Time-Sensitive Route, respectively. After describing the related work in Section 2, we formally describe these terms and the problem definition in Section 3.1. The main proposed method consists of two parts: Route Goodness Function and Route Construction. The former aims to measure the goodness of a given route, whereas the latter is to construct the route to satisfy the query. As the route construction algorithm proceeds, the route goodness function is exploited to intelligently determine the next location in the route. We first describe how to design the route goodness function in Section 3, and we elaborate the proposed route construction algorithm in Section 4. We evaluate the proposed method in Section 5 and demonstrate the *TripRouter* system in Section 6. Section 7 concludes this work.

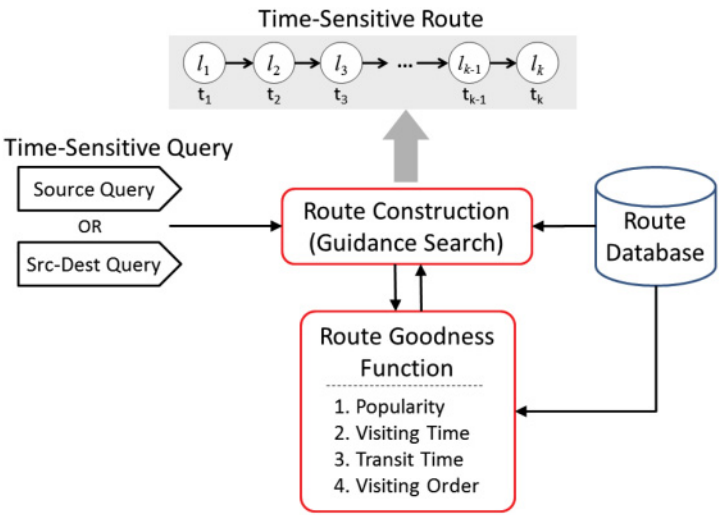


Fig. 3. The flowchart of the proposed time-sensitive route recommendation framework.

2. RELATED WORK

2.1. Route Planning by GPS Trajectory Data

There are lots of related works about route planning using the GPS trajectories. Yuan et al. [2011a, 2011b] find the fastest routes to a destination. Chen et al. [2011] and Wei et al. [2012] search for popular and attractive trajectories for recommendation. Chen et al. [2010] find the top-k trajectories connecting some user-given locations. Yoon et al. [2011] and Zheng et al. [2011] propose the itinerary recommendation by considering user preference based on mined trajectory attributes. Zheng et al. [2009, 2011] aim to discover interesting and classical travel sequences. Tang et al. [2011] find the top-k nearest neighboring trajectories with the minimum aggregated distance to some query locations. Wei et al. [2010] construct the top-k routes that sequentially pass through the query locations within the specified time span. Considering the travel cost (i.e., the financial cost and the time cost), Ge et al. [2011] provide a focused study of cost-aware tour recommendation. Liu et al. [2011] develop a Tourist-Area-Season Topic (TAST) model, which extract the topics conditioned on both tourists and intrinsic features of the landscape to recommend topic-dependent and/or season-based travel packages. Tang et al. [2012a, 2012b] investigate the problem of discovering groups that travel together from trajectory data streams. Their system can discover companions without accessing the object details. Bao et al. [2012] consider both user preference and social opinions to develop a location recommender system that allows users to specify a set of venues within a geospatial range.

Although there are many successful proposals to solve different kinds of route planning problems, the issues of proper visiting time for places and proper transit time between places are not tackled adequately. This work proposes to soundly measure and construct the time-sensitive trip routes using large-scale timestamped location sequence data.

We use Table I to summarize the differences between our work and other relevant studies. Here, we list some important issues about route planning, including whether it allows the Query of certain Locations (QL) and whether it considers the following ideas: Popularity (PO), Visiting Order (VO), Visiting Time (VT), Transit Time (TT), User

Table I. Summarization of Differences between This Paper and Other Related Works

	QL	PO	VO	VT	TT	UP	DI	TD	TK	CO	GS
Yuan et al. [2011a, 2011b]		■	■			■	■	■			
Chen et al. [2011]	■	■	■								
Wei et al. [2012]	■	■	■						■		
Chen et al. [2010]	■		■						■		
Yoon et al. [2011]	■	■	■		■	■		■	■		
Zheng et al. [2009, 2011]	■	■	■			■			■		
Tang et al. [2011]	■						■		■		
Wei et al. [2010]	■	■						■	■		
Ge et al. [2011], Liu et al. [2011]		■				■		■	■	■	
Tang et al. [2012]			■			■		■			■
Bao et al. [2012]	■	■				■	■		■		■
This work	■	■	■	■	■		■	■			

QL, Query of Location; PO, Popularity; VO, Visiting Order; VT, Visiting Time; TT, Transit Time; UP, User Preference; DI, Distance; TD, Travel Duration; TK, Top-*k* retrieval; CO, financial costs; GS, group and/or social consideration.

Preference (UP), Distance (DI), Travel Duration (TD), Top-*k* retrieval (TK), financial costs (CO), and group and/or social consideration (GS).

2.2. Route Recommendation Using Social Media

The rapid rise of social media applications generates a huge volume of geospatial data of human activities, such as geo-tagged photos in Flickr and check-in records in Foursquare. Both geo-tagged photos and check-in data can reveal how people sequentially visit places in an area. Using geo-tagged photos, Arase et al. [2010] mine frequent route patterns for recommendation. Cheng et al. [2011] propose personalized travel recommendation based on personal profiles and visual attributes of geo-tagged photos. Lu et al. [2010] and Kurashima et al. [2010] construct routes based on user preference of must-go destinations, visiting time, and travel duration. Yin et al. [2011] mine and rank trajectory patterns from geo-tagged photos and diversify the ranking results. Wei et al. [2012] infer the top-*k* routes traveling a given location sequence within a specified travel time from uncertain check-in data. In contrast to these works, we aim to perform knowledge discovery to construct the time-sensitive routes.

3. ROUTE GOODNESS MEASURE

3.1. Basic Definitions

Definition 3.1 (*Location*). A location l_i is a tuple, $l_i = (x_i, y_i)$, where x_i is the longitude, y_i is the latitude.

Definition 3.2 (*Route of Timestamped Locations*). A route is a sequence of locations with the corresponding timestamps, denoted by s, $s = <(l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n)>$, where n is the number of locations. Throughout this article, we focus on recommending single-day routes, which implies $t_n - t_1$ is no more than 24 hours.

Definition 3.3 (*Time-sensitive Query*). We define the *time-sensitive query* as $Q = (l_q, t_q)$, where l_q is the initial location of a user, and t_q is the starting time for this trip. Note that in the following we further define the source query and source-destination query (in Section 4), which are two kinds of time-sensitive query.

Definition 3.4 (*Time-sensitive Route*). Given a time-sensitive query, we define the *time-sensitive route* as a sequence of timestamped locations $s_r = <(l_1, t_1), (l_2, t_2), \ldots, (l_k, t_k)>$, which can have a higher score under the following defined route goodness function (in Section 3.2), where $l_1 = l_q$, $t_1 = t_q$, and k is the number of locations in the route, which can be either specified by users or be determined automatically based on the proposed algorithm (in Section 4). Note that the time-sensitive route is simply a common spatial-temporal sequence, just like a regular route. The main novelty lies in that the generation of these routes considers temporal information as constraints to optimize.

In the following sections, we describe how to measure the quality of a time-sensitive trip route. Based on the proposed goodness definition, we are able to search and recommend better time-sensitive routes given an initial time-sensitive query.

3.2. Measuring the Quality of a Trip Route

To construct a high-quality route for recommendation, we need to first design a proper metric to measure the quality of any given route. We propose that a good trip route should consider the following four factors: (a) the popularity of a place, (b) the proper visiting time of a location, (c) the proper transit time traveling from one location to another, and (d) the visiting order of places on the route. We attempt to model these factors into the goodness function and utilize this function to greedily select locations for the construction of the final trip route.

3.2.1. Route Popularity. A popular place, by definition, should be somewhere that attracts a number of visitors. If a route contains more popular places, it has higher potential to satisfy a user. The popularity of a place can be represented by the number of recording actions performed at that place. In our goodness measure of a route, we first consider the popularity of places on the route. We define the relative popularity of a location l_i as:

$$pop(l_i) = \frac{N(l_i)}{N_{max}},$$

where $N(l_i)$ is the number of recording actions performed on location l_i, and N_{max} is the maximum number of recording actions among all the locations in the location sequence data. Given a route $s = <(l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n)>$, we define the popularity-based goodness function $f_{pop}(s)$ as:

$$f_{pop}(s) = \left(\prod_{i=1}^{n} pop(l_i)\right)^{\frac{1}{n}}.$$

3.2.2. Proper Visiting Time. The timestamped location sequence data reveal that although some locations (e.g., park and movie theater) are popular regardless of the visiting time in a given day, other locations (e.g., stadium and beach) are more attractive during certain time periods of the day. We propose to learn such time-dependent popularity of each location from the location sequence data. We begin from defining the *Temporal Visiting Distribution* as the following.

Definition 3.5 (*Temporal Visiting Distribution [TVD] of a Location*). We define a TVD for a location l, $TVD_l(t_i)$, as the probability distribution of a randomly picked recording action of location l occurring at time t_i. For example, in a 24-hour span, *TVD* can be a legal probability distribution, as shown in Figure 4. *TVD* can easily be learned from the timestamped location sequence data, representing how popular a place is at a given time.

Using *TVD*, we can determine whether it is proper to visit a place at a given time. For example, assume that we want to know how well a decision is to visit a place at 8 AM,

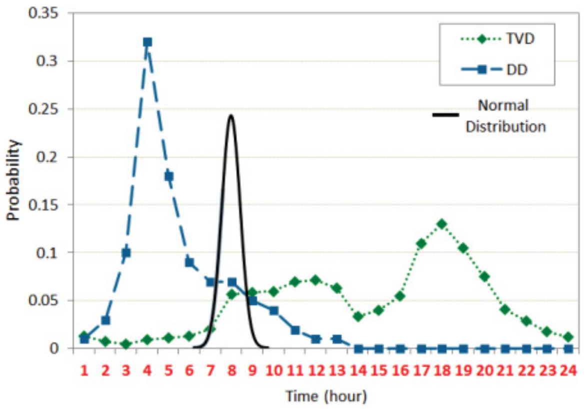


Fig. 4. Examples of the Temporal Visiting Distribution (*TVD*; the green dotted curve) for a certain location l_i, and the Duration Distribution (*DD*; the blue dashed curve) between location l_i and l_j. The black solid curve represents a normal distribution of a particular time assignment to measure the fitness values.

given the location's *TVD* as represented by the green dotted line in Figure 4. To do that, we propose to first generate a thin Gaussian distribution $G(t; \mu, \sigma^2)$ whose mean value μ is 8 with a very small variance σ^2 (e.g., standard deviation is 1). Then we can transform the original task into measuring the difference between the Gaussian distribution with the learned *TVD* of this location. Here, we use the *symmetric Kullback-Leibler* (*KL*) *Divergence* between $G(t; \mu, \sigma^2)$ and $TVD_l(t)$ to represent the fitness of the assignment. The formal mathematical definition of a fitness score between a place l and a time t can be defined as:

$$D_{KL}(G(t; \mu, \sigma^2)||TVD_l(t)) = \sum_x G(x; \mu, \sigma^2) log \frac{G(x; \mu, \sigma^2)}{TVD_l(x)} + \sum_x TVD_l(x) log \frac{TVD_l(x)}{G(x; \mu, \sigma^2)}.$$

Conceivably, a smaller KL value indicates a better match between the assignment and the distribution learned from data. Consequently, we formally define the temporal visiting goodness function $f_{visit}(s)$ of a route $s = <(l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n)>$, as a combination of the popularity of places together with the fitness of each location over time, in the following equation:

$$f_{visit}(s) = \left(\prod_{i=1}^{n} D_{KL}\big(G(t; t_i, \sigma^2)||TVD_{l_i}(t)\big) \times \frac{1}{pop(l_i)} \right)^{\frac{-1}{n}}.$$

If the places in a route s are visited during the proper time period, the $f_{visit}(s)$ value would become higher.

3.2.3. Proper Transit Time Duration. To schedule a good trip route, another key element to be considered is the visiting time of each place as well as the transit time from one place to another. Although the timestamped location sequence data (user check-in data in this article) cannot explicitly tell us these two kinds of information, we can simply treat the duration between two checked-in places as the summation of the visiting time of the first place plus the transportation time from the first to the second place. Such duration can further be utilized to evaluate the quality of a trip. Here, we propose the *Duration Distribution* (DD), as defined in the following paragraph, to model such "visiting plus transit time" between places.

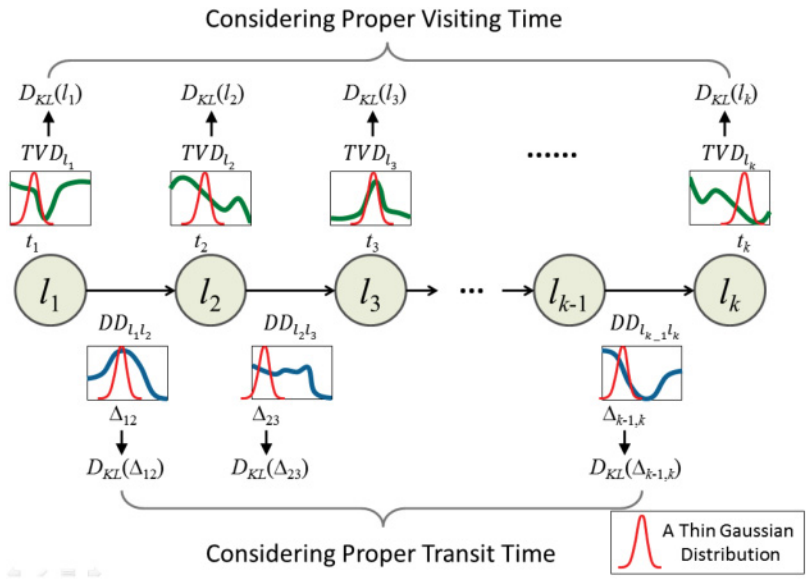


Fig. 5. For a route $s = <(l_1, t_1), (l_2, t_2), \ldots, (l_k, t_k)>$, we compute $2k - 1$ values of KL-divergence and then take the geometric mean of such values as the time-dependent goodness of a route.

Definition 3.6 (*Duration Distribution [DD] between Two Locations*). We define the *DD* between locations l_i and l_j as the probability distribution over time duration Δ, $DD_{l_il_j}(\Delta)$, which can be obtained from the following random experiment: randomly pick two consecutive location recording actions (l_i, t_i), (l_j, t_j) of a person and calculate the probability that $t_j - t_i = t$.

Again, we consider only one-day trips and therefore treat the outcome space of *DD* between hours 0 through 24. For example, any legal probability distribution between hours 0 through 24 can be a *DD* (e.g., the blue dashed line in Figure 4).

Similar to what we do to obtain *TVD*, given a pair of locations l_i and l_j together with an assignment of a given time duration Δ among them, we can model Δ as a thin Gaussian distribution and compare it with $DD_{l_il_j}(\Delta)$ using symmetric KL divergence. Consequently, for a route $s = <(l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n)>$, it is possible to know how good the route is based on the durations between places by defining a goodness function of duration:

$$f_{duration}(s) = \left(\prod_{i=1}^{n-1} D_{KL}\big(g(t; \Delta_{i,i+1}; \sigma^2)||DD_{l_il_j}(\Delta)\big)\right)^{\frac{-1}{n-1}}.$$

A route s with a higher value of $f_{duration}(s)$ indicates that such a route can be visited with proper "transit + visiting" time between places.

Here, we use Figure 5 as an illustration to summarize our idea of utilizing *TVD* and *DD* to measure the goodness of a trip route. Given a route $s = <(l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n)>$. We use symmetric KL divergence to measure the visiting fitness of each location l_i by calculating a $D_{KL}(l_i)$ value between TVD_{li} and a narrow Gaussian distribution. We also use KL divergence to measure the fitness of each transition $l_i \rightarrow l_j$ and derive a $D_{KL}(\Delta_{ij})$ between $DD_{l_il_j}$ and a thin Gaussian distribution. Eventually, we compute the geometric mean of such D_{KL} values to be the time-related route goodness.

3.2.4. Proper Visiting Order. Due to the characteristic of each place, there might be certain latent patterns about the order of the places to be visited. With the timestamped location sequence data, we are able to learn such orders and exploit them to evaluate the quality of a route. For example, going to restaurant for dinner and then going back to the hotel is better than the other way around. In this section, we propose to exploit the idea of the *n*-gram language model to measure the quality of the order of visits in a trip route. Using the location sequence corpus, we can first generate the n-gram probabilities of locations. Then, given a route $s = <(l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n)>$, we can compute its *n*-gram probability. We consider such *n*-gram probability as the goodness of visiting order. Technically, we use the average value of the probabilities of uni-gram, bi-gram, and tri-gram to estimate the goodness of orders. Note that the uni-gram probability corresponds to the popularity-based route goodness. We can formally write the probabilities as follows:

$$\begin{aligned} P_{uni}(s) &= f_{pop}(s). \\ P_{bi}(s) &= (P(l_1)P(l_2|l_1)P(l_3|l_2)\cdots P(l_n|l_{n-1}))^{\frac{1}{n}}. \\ P_{tri}(s) &= (P(l_1)P(l_2|l_1)P(l_3|l_1l_2)\cdots P(l_n|l_{n-2}l_{n-1}))^{\frac{1}{n}}. \end{aligned}$$

Therefore, the goodness of visiting order of a route can be defined:

$$f_{order}(s) = \frac{(P_{uni}(s) + P_{bi}(s) + P_{tri}(s))}{3}.$$

Higher $f_{order}(s)$ value represents better quality of route. Note that we utilize the add-one technique for smoothing. Also note that since the trip route would not be too long, and visiting a certain place usually depends on only the previous and the next locations, we consider only up to tri-gram.

3.2.5. Final Goodness Function. Here, we integrate the goodness measures of the proper visiting time, the proper transit time duration, and the proper visiting order into the final goodness function $f(s)$. Our design of the route goodness function integrates these factors accordingly. We divide the goodness function into two parts and provide a weighting parameter for users to determine the significance and balance of these two parts. The first part is the average value of the temporal visiting goodness $f_{visit}(s)$ and the location transition goodness $f_{duration}(s)$. The second part is the visiting order goodness $f_{order}(s)$. Note that the first part $f_{visit}(s)$ is time-sensitive whereas the second part is not. We use a parameter $\alpha \in [0, 1]$ to devise a linear combination of these two parts. This parameter provides users the flexibility to specify whether they prefer the time-sensitive routes. The final goodness function $f(s)$ is defined in the following:

$$f(s) = \alpha \times \left(\frac{f_{visit}(s) + f_{duration}(s)}{2}\right) + (1 - \alpha) \times f_{order}(s).$$

A route s with higher value of $f(s)$ will be considered a better route. Experiments in Section 5.3 suggest $\alpha \approx 0.9$ as being more effective on measuring route quality. Such a result exhibits the usefulness of the proposed time-sensitive route recommendation.

4. ROUTE CONSTRUCTION ALGORITHM

In this section, we define the time-sensitive trip route recommendation problem based on the proposed route goodness measure. We design two types of queries, *source* query and *source-destination* query, which correspond to the two real-life route recommendation tasks. To solve such problems, we propose a route search algorithm, *Guidance Search*.

4.1. Problem Statement

We first provide the definitions of the source and source-destination queries in the following paragraphs.

Definition 4.1 (*Source Query*). The input consists of (a) the source/starting location $Q_s = (l_s, t_s)$, where l_s contains the longitude and latitude of such a location, and t_s is the starting timestamp (e.g., 8 AM); and (b) the number k of locations in the final route to be recommended.

Definition 4.2 (*Source-Destination Query*). The input consists of (a) the source/starting location $Q_d = (l_s, t_s, l_d)$, where l_s contains the longitude and latitude of such location, t_s is the starting timestamp (e.g., 8 AM), and l_d is the destination/end location; and/or (b) the number k of locations in the final route to be recommended.

Both queries are very common for real-world trip planning. For source query, people might want to know where they can visit in a city given their available starting time. For the source-destination query, people might need to arrive at a destination place starting from the current place at a certain time, but want to know along where along the line of the route they can visit.

Therefore, for our trip recommendation system, given (a) the routes extracted from the timestamped location sequence data and (b) either the source query $Q_s = (l_s, t_s)$ or the source-destination query $Q_d = (l_s, t_s, l_d)$, the goal is to construct a route $s_r = <(l_1 = l_s, t_1 = t_s), (l_2, t_2), \ldots, (l_k, t_k)>$ that optimizes $f(s_r)$. Note that l_k is required to be l_d for the source-destination query.

THEOREM 4.3. *The time-sensitive route construction problem with the source-destination query is NP-hard.*

PROOF. We prove this theorem by a reduction from the *Longest Path Problem* (LPP) [Cormen et al. 2009]. In the decision version of LPP, we are given a weighted graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes and E is the set of edges, as well as a source node v_s and a destination node v_d. Each edge $(v_i, v_j) \in E$ is associated with a non-negative weight c_{ij}. The length of a path $p = v_1 = v_s, v_2, \ldots, v_k = v_d$ is defined by $c(p) = \sum_{i=1}^{k-1} c_{i,i+1}, \ldots, k$. The objective is to find a simple path $LP(v_s, v_d)$ (i.e., path without any repeated nodes) with the maximum length among all paths from v_s to v_d in G.

Now we transform an instance of the LPP to an instance of our Time-Sensitive Route Construction problem with Source-Destination (TRCSD) query $Q_d = (l_s, t_s, l_d)$ as follows. Based on the route database constructed from user check-in sequences (the details are given in Section 5), we can build a graph $H = (L, A)$, where L is the set of check-in locations and A is the set of edges connected by two consecutively visited locations (l_i, l_j). Let $s_{x \to y}$ be the route starting from location l_x to l_y in H. Consider a special case of TRCSD: all the locations and all the routes between locations are visited once (which leads to $f_{order}(s) = 1$ for any given route s), and both the temporal visiting distributions on all the locations and the duration distributions between locations are generated by any kind of monotonically increasing functions with respect to time (which leads to $f_{visit}(s)$ and $f_{duration}(s)$, which are nondecreasing for any given route s; i.e., $f_{visit}(s_{s \to \cdots \to i}) \leq f_{visit}(s_{s \to \cdots \to i \to j})$ and $f_{duration}(s_{s \to \cdots \to i}) \leq f_{duration}(s_{s \to \cdots \to i \to j})$). Therefore, the route goodness function $f(s)$ is monotonically increasing. Based on such a special case, for every edge $(v_i, v_j) \in E$ in LPP, we use the route goodness function $f(s)$ of TRCSD to create the corresponding edge weight c_{ij}. By assigning the starting locations l_s and l_d in TRCSD to nodes v_s and v_d in LPP, respectively, we compute the edge cost of edge (v_i, v_j) by $c_{ij} = f(s_{s \to \cdots \to i \to j}) - f(s_{s \to \cdots \to i})$. Hence, the graph H in TRCSD is identical to the graph G in LPP. Resulting from this mapping, we can find that there exists a path solution to the LPP problem with maximum path length if and only if there exists a route solution to the TRCSD problem with maximum route goodness. Therefore, the proof is complete. □

4.2. Route Construction Algorithm: *Guidance Search*

We develop the *Guidance Search* algorithm to deal with the route recommendation task for the source and source-destination queries. The general idea aims to find a proper time-sensitive route (whose locations possess proper visiting time, transition time, visiting order, and higher popularity) and requires the constructed route to start from the source location for the source query (and finally reach the destination location for the source-destination query). Therefore, in addition to constructing the route up to length k from the source location, we further need to implement a certain guiding mechanism to direct the search algorithm to move toward the destination if the destination is specified. To fulfill such a goal, we develop a novel search algorithm, *Guidance Search*, which takes advantage of the idea of a *best-first search* that attempts to find a least-cost path from a given initial node to the goal. The central idea of *Guidance Search* consists of two main parts. The first is to design the *heuristic satisfaction function*, which is in charge of the guidance to determine the next most promising location toward the destination. The second is the *backward checking mechanism* that keeps the historical search tree (all the expanded routes starting from the source location) for reconstructing the route with a higher *satisfaction* score.

We first describe the deployment of the *heuristic satisfaction function* $f^*(l)$. Because our central goal is to recommend a trip route with great time-sensitive satisfaction, we consider the time-sensitive route goodness function $f(s)$ (described in Section 3.2.5) as the fundamental to design the heuristic satisfaction function $f^*(l)$. The input of the heuristic satisfaction $f^*(l)$ is a location l. The goal of $f^*(l)$ is to measure the satisfaction of selecting location l as the next visiting location considering the subroute from the source location l_s to the location l and from the location l to the destination location l_d. On one hand, if only the former part (i.e., from l_s to l) is considered, the source query can be tackled. On the other hand, we can address the source-destination query when both the former and the latter parts (i.e., from l to l_d) are considered.

$f^*(l)$ can be decomposed into two parts. The first is the time-sensitive route goodness function $g(l) = f(l_s \to l)$. $g(l)$ computes the goodness score of the subroute starting from the source location l_s to the location l. The second is the *heuristic function* $h(l)$. $h(l)$ considers both the route goodness $f(l \to l_d)$ and the geographical distance $d(l \to l_d)$ of a subroute from the location l to the destination l_d. The former part $f(l \to l_d)$ aims to estimate the route goodness of a certain subroute from l to l_d. The latter part $d(l \to l_d)$ serves as the role of guidance. Specifically, we use the geographical distance between l and l_d as the steering force to direct the search process to move toward the destination location. When selecting the next visiting location during route construction, those locations with shorter distances to the destination have a higher chance to be picked if the rest of the criteria are equally satisfied. Moreover, because there could be many subroutes from l to l_d in the route database, we will compute all the scores and take the best one as the final $h(l)$ value. Consequently, the heuristic function can be formally written as:

$$h(l) = max_{(l \to l_d) \in S(l \to l_d)} \left\{ \sqrt{f(l \to l_d) \times (1 - d(l \to l_d))} \right\},$$

where $S(l \to l_d)$ is the set of subroutes starting from l to l_d extracted from all the routes derived from the timestamped location sequence data. Note that the $d(l \to l_d)$ is

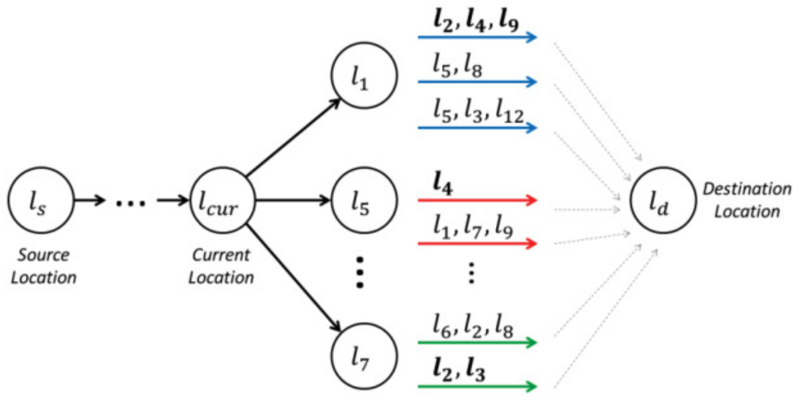


Fig. 6. An example to elaborate the idea of the proposed *heuristic satisfaction function* for tackling the source-destination query.

normalized to [0, 1]. Eventually, we write down the final heuristic satisfaction function as:

$$f^*(l) = (1 - \beta) \times g(l) + \beta \times h(l),$$

where $\beta \in [0, 1]$ is the weighting parameter to control the strength of the guidance to the destination. Higher β values indicate stronger guidance. When $\beta = 0$, $f^*(l)$ is considered as a greedy search with backward checking mechanism, and it can be utilized to tackle the source query (because in the $g(l)$ function, no destination information is needed).

Here, we give an example to elaborate the idea of the proposed *heuristic satisfaction function* $f^*(l)$ for selecting the next visiting location toward the destination location in Figure 6. Assume our search is current at location l_{cur} and has to find the next best location to expand from l_{cur}. In the route database, assuming there are some candidate locations l_1, $l_5, \ldots,$ and l_7 that have ever been visited after l_{cur}, then the satisfaction scores $f^*(l_1)$, $f^*(l_5), \ldots,$ and $f^*(l_7)$ can be generated. For instance, to derive $f^*(l_1)$, all the subroutes starting from l_1 and ending at l_d (such as $<l_1, l_2, l_4, l_9, l_d>$, $<l_1, l_5, l_8, l_d>$, and $<l_1, l_5, l_3, l_{12}, l_d>$) are considered to find the one with the maximum score among them as the $h(l_1)$ value. On the other hand, we can derive $g(l_1) = f(l_s \rightarrow l_1)$. Eventually, we have $f^*(l_1) = (1 - \beta) \times g(l_1) + \beta \times h(l_1)$. By applying such a computation process to l_1, $l_5, \ldots,$ and l_7, it is possible to choose the next visiting location with the highest satisfaction.

The *backward checking mechanism* is the key to the best-first search and can be applied to our route construction algorithm. Specifically, while exploiting the *heuristic satisfaction function* to choose the next location to visit, it is necessary to expand all neighbor locations to generate the corresponding satisfaction scores. We keep track of such scores in the expansion tree. When it is necessary to select the next visiting location, not only the expanded locations from the current location, but also those that had ever been expanded during the previous rounds can be considered. In other words, in addition to continuing to expand the current location, the algorithm can possibly go backward to consider the previously expanded nodes and find those with the highest satisfaction score.

The *Guidance Search* algorithm to construct the route for tackling either the source query or the source-destination query is given in Algorithm 1. We first construct the initial route s_0 by including the source location l_s (line 1). A *PriorityQueue* is employed for the purpose of the backward checking mechanism (line 2). Each element in the

ALGORITHM 1: *Guidance Search*

```
Input: (a) RouteDB: routes extracted from the timestamped location sequences data;
(b) $Q_d = (l_s, t_s, l_d)$: if $l_d = \emptyset$: the source query,
if $l_d \neq \emptyset$: the source-destination query;
(c) $k$: the number of locations in the final route $s_r$.
Output: a time-sensitive route $s_r = \langle (l_1, t_1), (l_2, t_2), \ldots (l_d, t_k) \rangle$.
$s_0 = \langle (l_1 = l_s, t_1 = t_s) \rangle$.
$PriotityQueue = \{(s_0, 0)\}$.
$s_r = s_0$. $l_{last} = null$.
while $|s_r| < k$ OR $l_{last} \neq l_d$ do:
    $l_{last} = sr.EndLocation$.
    $C = \{l_{next} | l_{last} \rightarrow l_{next}$ in $RouteDB\}$.
    for each $l_c \in C$ do:
        $s_{tmp} = s_r + \langle (l_c, t_c) \rangle$.
        $score = f^*(l_c)$.
        $PriorityQueue.Insert((s_{tmp}, score))$.
    end
    $s_r = PriorityQueue.Pull()$.
Return: $s_r$.
```

PriorityQueue consists of a route s and the corresponding heuristic satisfaction score. The *PriorityQueue* automatically sorts its elements according to their satisfaction scores. We add s_0 to initialize the *PriorityQueue*. After setting the final route s_r as the initial one s_0 (line 3), we perform the iterative expansion search process until the route s_r is constructed up to length k (lines 4–12). For each iteration, the last location l_{last} in the route s_r with the highest satisfaction score is identified (line 5 and line 12) and each possible next visiting location l_{next} is put into a candidate set C (line 6). Then, for each candidate for the next location l_c, we can derive the heuristic satisfaction score $f^*(l_c)$ (if $l_d = \emptyset$, we set the weighting parameter $\beta = 0$ for the function f^*; otherwise $0 < \beta \leq 1$). We put $f^*(l_c)$ together with the corresponding route s_{tmp} into the *PriorityQueue* (lines 7–11). The *PriorityQueue* will then pick the next best route and location to conduct the further expansions (line 12). Finally (line 13), the route s_r is reported as the result of recommendation.

4.2.1. Time Complexity. The time complexity of *Guidance Search* mainly depends on three parts: the number of search expansion N, the number of candidates of next location $|C|$, and the operation of the *PriorityQueue* $O(\log(|E|))$, where E is the set of elements in the *PriorityQueue*. We can simply write down the complexity as $O(N \times |C| \times \log(|E|))$. In the worst case, both the candidate set C_i of each next location and the number of search expansion could be all the locations in the route database. By denoting the set of all locations in the database as L, the complexity is $O(|L|^2 \times \log(|E|))$ for the worst case.

5. EXPERIMENTS

5.1. Dataset and Data Analysis

We utilize the Gowalla dataset [Cho et al. 2011] which has been exploited for location-based analysis in several places (such as Scellato et al. [2010, 2011]) to construct our time-sensitive route recommendation. The Gowalla dataset contains 6,442,890

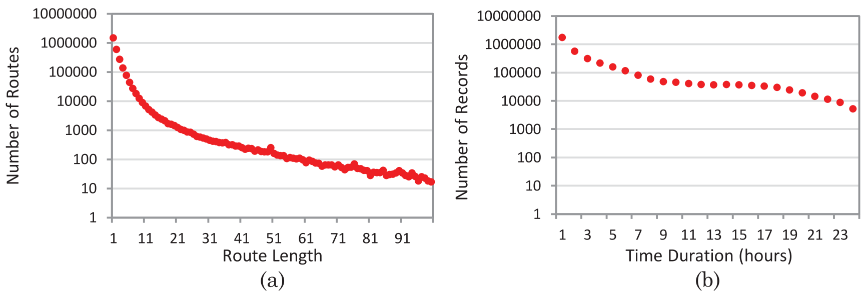


Fig. 7. (a) Distribution of route length. (b) Distributions of time duration in *RouteDB*.

Table II. The Statistics of *RouteDB* and the Three Subsets

	Total Number of Check-ins	Avg. Route Length	Variance of Route Length
RouteDB (all data)	6,442,890	4.09	48.04
New York	103,174	4.46	71.24
San Francisco	187,568	4.09	58.36
Paris	14,224	4.45	75.73

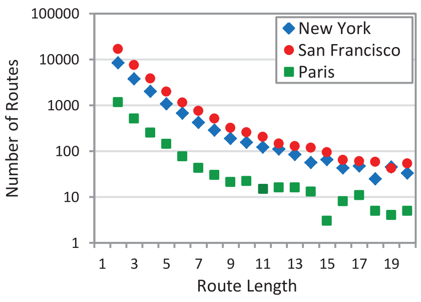


Fig. 8. Distribution of route length for three cities.

check-in records from February 2009 to October 2010. The total number of check-in locations is 1,280,969. Considering a route as a sequence of a user's check-in locations within a day, we construct the route database *RouteDB* containing 1,136,737 routes whose length is more than one (the average length is 4.09). Figure 7(a) shows the distribution of the route length, which is highly skewed and heavily tailed. Figure 7(a) also shows that people usually do not prefer visiting too many locations in a day, but with some exceptions. Figure 7(b) shows the distribution of the time difference between the visiting of two places. This indicates that people consider places closer to where they are while planning the next destination.

We extract three subsets of the check-in data that correspond to the cities of New York, San Francisco, and Paris. Some statistics are reported in Table II. Figure 8 shows the distribution of route length in the three subsets of check-in data, whereas Figure 9 shows the distribution of the time duration.

5.2. Evaluation Plan

In this section, we introduce two experiments to test the effectiveness of our goodness model and demonstrate the performance of the proposed search methods. In the first experiment, we conduct **pairwise time-sensitive route detection** to compare the quality of our goodness model with several baseline methods. In the second experiment, we design the **time-sensitive cloze test of locations in routes** to validate the quality

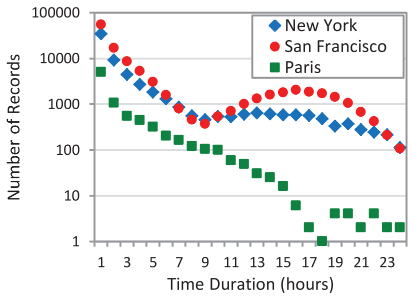


Fig. 9. Distribution of time duration for three cities.

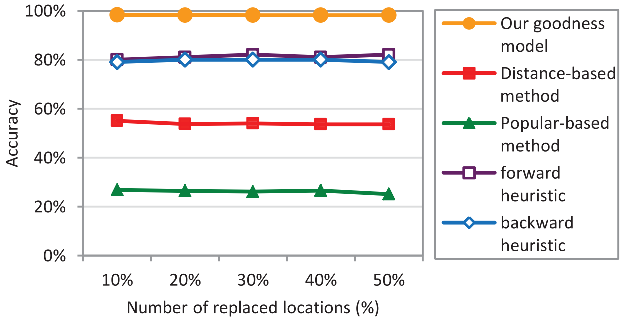


Fig. 10. Accuracy by varying the number of replaced locations in San Francisco.

of the recommended routes by our search algorithm, comparing the route to not only our previously proposed greedy method [Hsieh et al. 2012], but also to a series of baseline methods. We show the results in Section 5.3 and provide some discussions in Section 5.4.

Experiment 1: **Pair-wise Time-sensitive Route Detection.** In this experiment, we would like to verify whether our goodness model can rank the existing routes higher than the nonexisting ones. We first randomly choose 1,000 real routes from the check-in data. Note that the timestamp is associated with each location l. For each route, we replace a portion of the locations with other locations in the same city to generate a pseudoroute. Note that each existing route will be paired with a pseudoroute. To make the task nontrivial, we adopt a replacing strategy to replace a location with a "plausible" one instead of a randomly selected one. That is, to replace a location at position i of a route, we only choose from candidate locations that have ever appear right after the location at position i-1 (e.g., the bi-gram probability of them is non-zero). Furthermore, after the replacement, we double-check to make sure the generated pseudo-routes do not exist in the database. That is, there is no such route in the database of the same location sequences together with the same associated timestamps. As can be seen in Figures 10–12, the amount of replaced locations occupies from 10% to 50% of a route. That is, for each route, at least half of the locations are correct. We then use our goodness model to examine each pair of existing route and its pseudo-route, and we record how frequently our method ranks the correct one higher. Finally, we report the accuracy of our method and compare it with the baseline results. The accuracy is defined as the number of correctly ranked pairs (i.e., an algorithm assigns higher score to the existing route than to the pseudo-route) divided by the total number of pairs.

Similarly, we can generate another kind of pseudo-route by perturbing the timestamps of certain locations in an existing route. For example, given an existing route $s = <(l_1, t_1), (l_2, t_2), \ldots (l_{i-1}, t_{i-1}), (l_i, t_i), (l_{i-1}, t_{i-1}), \ldots, (l_n, t_n)>$, we change t_i to a different time t_j, where $t_{i-1} < t_j < t_{i+1}$. We expect a proper fitness function to assign lower scores to such pseudo-routes. In general, the baseline approaches cannot handle the case of time perturbation so that the identification rate is 50%.

Experiment 2: Time-sensitive Cloze Test of Locations in Routes. Given some real trip routes with timestamps in each location, through randomly removing m consecutive locations in each route, it is possible to test whether a method can successfully recover the missing locations. With increasing m, consecutive removal of locations does impose a decent level of difficulty to this cloze test, because when m increases, the information that can be utilized becomes sparse, and mistakes in the earlier position can lead to follow-up errors in the next positions to be predicted. Here, we use the *Hit Rate* as the accuracy measure for this cloze test. Given that there are a total of N removals of locations over all routes, and assuming that M places out of N are successfully predicted, the hit rate is defined as M/N. A higher hit rate indicates a better quality of recommendation. For each city, we extract all the existing routes of length larger than 6 for our experiments. There are a total 3,702 routes with average route length 10.26 for this experiment. Note that when there are multiple missing places in the cloze test, we only consider the fully recovered sequences as hits. For instance, if the system fills (A X Y Z E) in a cloze route (A_ _ _ E) while the original sequence is (A B C D E), all X Y Z will be considered as missing even if another sequence (A B Y D E) exists in the database.

Baseline Approaches. To evaluate the effectiveness of our method, we design the following four baseline methods.

—**Distance-Based Approach.** This method chooses the closest location to the current spot as the next spot to move to. It rates a route using the goodness function $f_d(s) = (\prod_{i=1}^{n} \frac{1}{D(l_i, l_{i-1})})^{\frac{1}{n}}$, where $D(l_i, l_{i-1})$ is the geographical distance between two consecutive locations.

—**Popularity-Based Approach.** This method chooses the most popular spot of a given time in that city as the next spot to move to. It rates the path using the goodness function $f_{pop}(s)$ as defined previously in Section 3.2.1.

—**Forward Heuristic Approach.** The forward heuristic chooses a location l_i that possesses the largest bi-gram probability with the previous location $P(l_i|l_{i-1})$ as the next location to move to. Its goodness function is $f_{forw}(s) = P_{bi}(s)$, as defined previously in Section 3.2.4.

—**Backward Heuristic Approach.** The backward heuristic chooses a location l_i that possesses the largest bi-gram probability with the next location $P(l_i|l_{i+1})$ as the next location to move to. The fitness function can be described as $f_{backw}(s) = (P(l_1|l_2)P(l_2|l_3) \cdots P(l_{n-1}|l_n))^{\frac{1}{n}}$.

5.3. Experimental Results

Section 5.3.1 shows the results of pairwise route detection and Section 5.3.2 presents the outcome of cloze test. For both experiments, we implement four baseline methods to compare with. We also compare our method with the greedy search method of our previous work [Hsieh et al. 2012].

5.3.1. Pairwise Time-Sensitive Route Detection. We first vary the number of replaced locations from 10% to 50% and report the accuracy of different methods. We set α of our goodness function as 0.5. Figure 10 contains the results for San Francisco. Our fitness

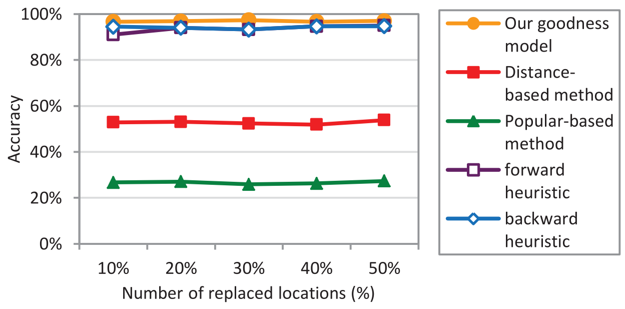


Fig. 11. Accuracy by varying the number of replaced locations in New York.

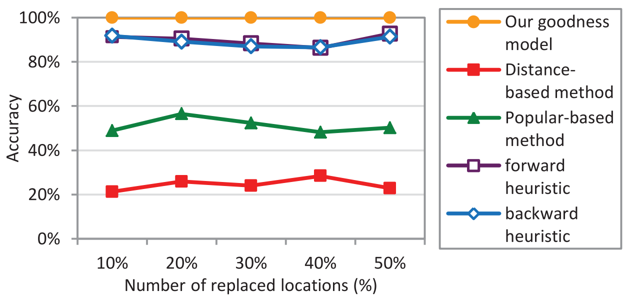
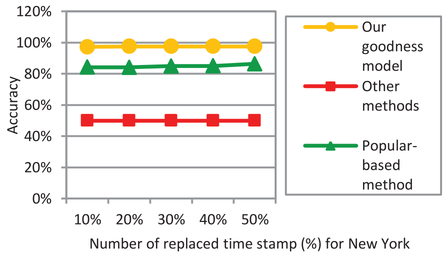


Fig. 12. Accuracy by varying the number of replaced locations in Paris.

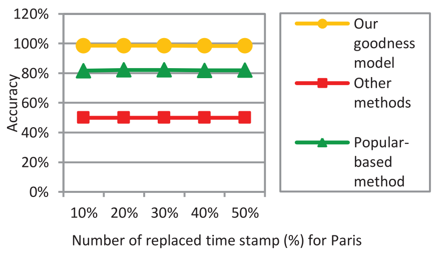
model achieves around 98% accuracy in distinguishing the real routes from replaced ones. The accuracy scores of the forward and backward heuristics reaches about 80%. The popular-based and distance-based methods do not do a good job because they only reach 50% or lower in accuracy. Similar trends occur in New York and Paris (Figures 11 and 12). The results are not surprising because our method does consider the location preference over time and location order.

Figure 13 shows the results of creating pseudo-paths by shifting timestamps for some locations in three cities. Again, we vary the ratio of change from 10% to 50%. The results show that our model can almost perfectly detect such change, better than the popularity-based method (around 15% accuracy). The other competitors, such as distance-based, forward heuristic, and backward heuristic methods do not have the capability to distinguish such pairs because they do not consider time information during route generation and therefore can only achieve 50% for such pairs of routes as shown in Figure 13.

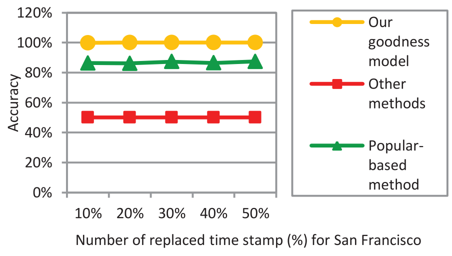
5.3.2. Time-Sensitive Cloze Test in Routes. In cloze experiment of locations in routes, we first report the hit rate in three cities by varying β (i.e., the weight for $h(l)$) from 0 to 1, as shown in Figures 14–16. Our goal is to investigate whether the heuristic function $h(l)$ mentioned in Section 4.3 can improve the hit rate. We set the number of missing locations as 3 and α (for goodness function) as 0.5. The results indicate that increasing the weight of a heuristic function $h(l)$ provides a positive influence to the performance. When $\beta = 0.5$, it can consistently produce the best (or close to the best) results. The hit rates of the four baseline models are lower than 3% in these three cities. Moreover, we compare this with a greedy-based approach we proposed previously [Hsieh et al. 2012], and we found that our search can consistently outperform the previous method. It is not surprising because the previous greedy method does not embed a heuristic function to consider the quality of selection to the destination.



(a) New York



(b) Paris



(c) San Francisco

Fig. 13. Accuracy by varying the number of replaced timestamps for our method in (a) New York, (b) Paris, and (c) San Francisco.

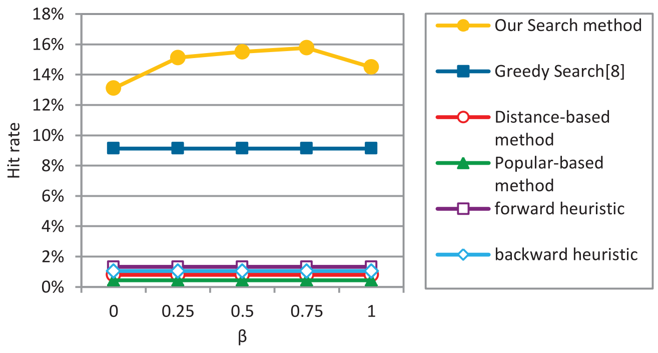


Fig. 14. Hit rate by varying the weight of β in New York.

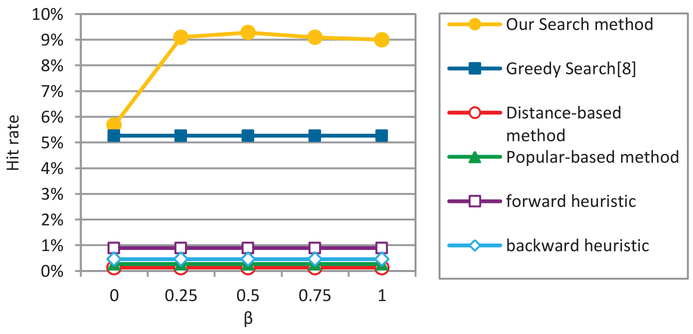


Fig. 15. Hit rate by varying the weight of β in San Francisco.

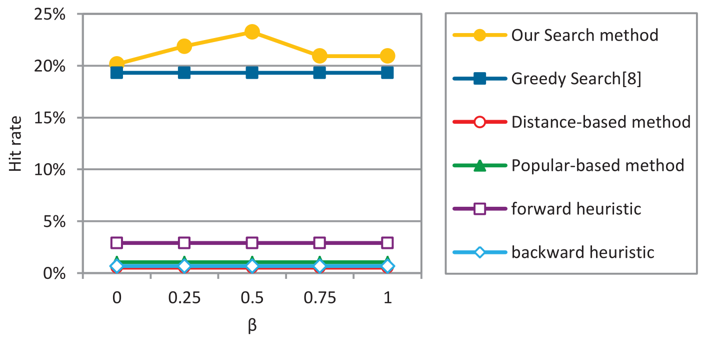


Fig. 16. Hit rate by varying the weight of β in Paris.

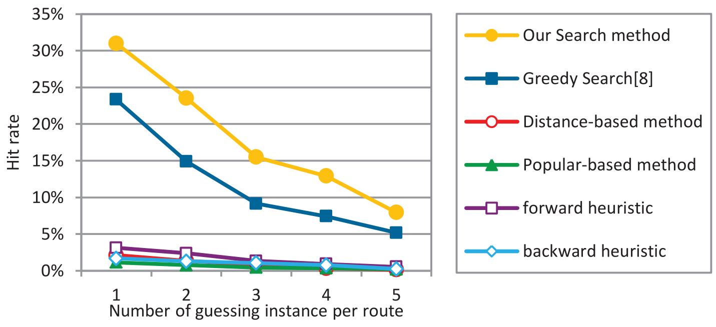


Fig. 17. Hit rate by varying the number of guessing instances per route in New York.

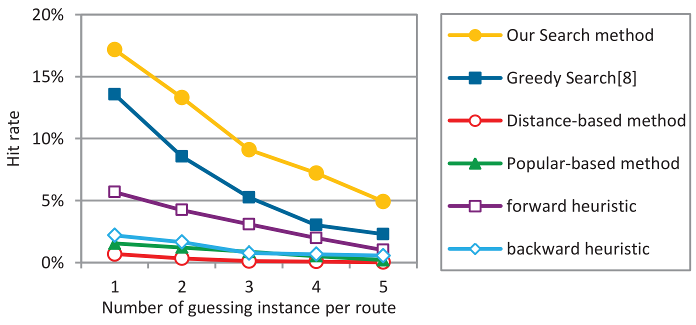


Fig. 18. Hit rate by varying the number of guessing instances per route in San Francisco.

Next, we vary the number of missing instances per route and report the hit rate in three cities. Here, we set the weight of β as 0.5 (i.e., the $g(l)$ function and $h(l)$ function are considered as equally important). General speaking, the hit rate of each methods is decreasing while the number of missing instances increases. Our proposed method still outperforms the greedy search method [Hsieh et al. 2012] and other baselines significantly. The results in three cities are shown in Figures 17–19.

5.4. Discussions

5.4.1. The Relationship between $g(l)$ and $h(l)$. While sequentially recommending the route, our search model utilizes destination information $h(l)$ as heuristics. In this section, we investigate the interaction between $h(l)$ and $g(l)$ through varying β.

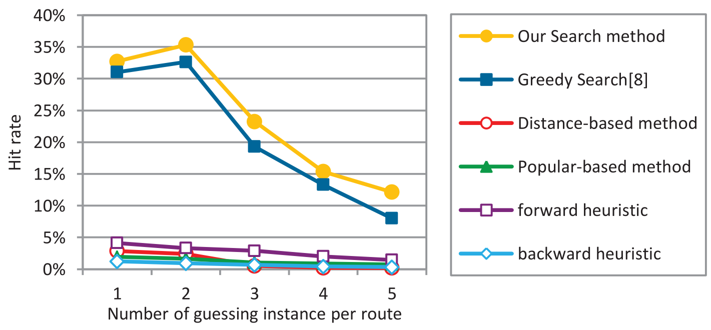


Fig. 19. Hit rate by varying the number of guessing instances per route in Paris.

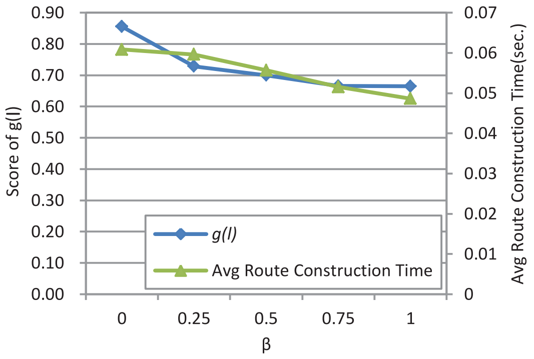


Fig. 20. Score of $g(l)$ for *Guidance Search* and the corresponding average construction time by varying β in New York.

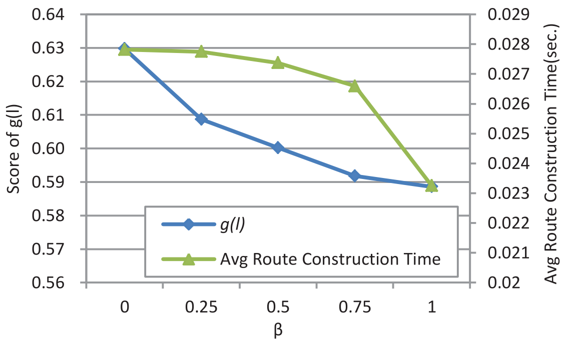


Fig. 21. Score of $g(l)$ for *Guidance Search* and the corresponding average construction time by varying β in San Francisco.

Given a random chosen source location, destination location, and starting time, we exploit the *Guidance Search* to recommend a route r. In this discussion, we generate a total of 1,000 routes and control the route length so that there are 200 routes each for the length from 4 to 8.

In Figure 20, we vary the parameter β for *Guidance Search* and examine the corresponding $g(l)$ score, as well as the time needed to produce a route in New York City. Similarly, same settings are applied to San Francisco and Paris. The results are reported in Figures 21 and 22. These show that when β becomes larger, it considers

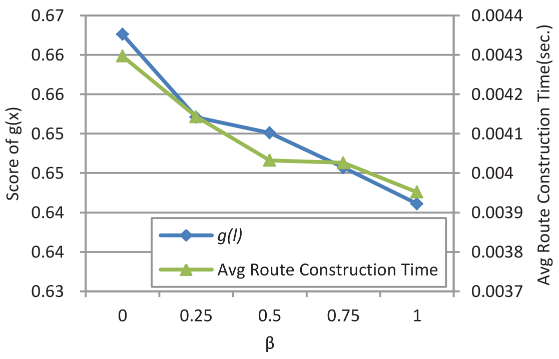


Fig. 22. Score of $g(l)$ for *Guidance Search* and the corresponding average construction time by varying β in Paris.

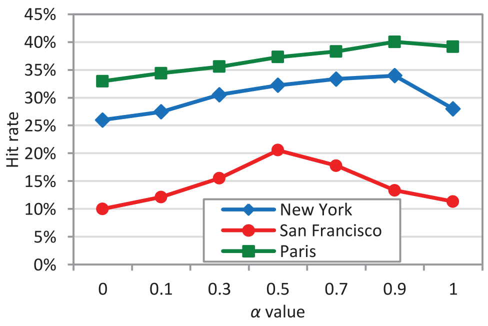


Fig. 23. The impact of α for our *Guidance Search* method on the time-sensitive cloze test for the three cities.

paying more attention to the heuristic satisfaction score $h(l)$, rather than to simply optimize $g(l)$. Therefore, $g(l)$ decreases as β increases.

Because the $h(l)$ function utilizes the destination information to guide the search, increasing β (i.e., emphasizing h) allows us to reach the destination more quickly. However, it pays the cost of not optimizing the fitness function $g(l)$. Figures 20–22 demonstrate that by setting for an inferior $g(l)$ value, we can indeed improve the efficiency through increasing β.

5.4.2. Impact of α for Cloze Experiment. We examine how sensitive *Guidance Search* is to the parameter α (higher α means we pay more attention to time-sensitive routes), ranging from 0 to 1. We evaluate on the hit rate of the cloze test and display the results in Figure 23. In New York and Paris, the best α value is around 0.9. That is, time-sensitive information is preferable to the visiting order on the cloze test task. In San Francisco City, α performs well at 0.5.

5.4.3. The Approximation Ratio of Guidance Search *Compared with Exhaustive Search.* In this section, we conduct an evaluation to measure the closeness of *Guidance Search* and other baseline methods toward the optimal route (i.e., the route that maximizes the goodness function) generated by exhaustive search. We randomly generate 1,000 queries from each city and compare the goodness scores of the proposed *Guidance Search*, f(sGS) and other baseline methods. Because the optimization problem is NP-hard, we obtain the optimal solution relying on the exhaustive search method. We can then calculate the approximation ratio (from 0 to 1) to determine how close the goodness measure of each method is toward the optimal goodness value. In Figure 24,

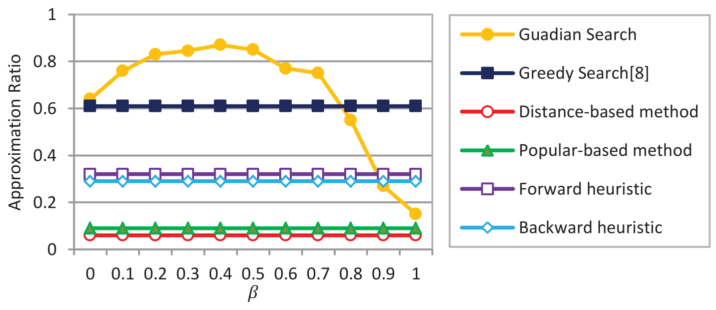


Fig. 24. The approximation ratio of all methods compared with the exhaustive search.

Table III. The Criteria and the Corresponding Questions for User Study

Criteria	Question
Location Popularity (LP)	Do you think these recommended locations are popular ones?
Visiting Order (VO)	Is the visiting order of locations in the route suitable/acceptable?
Visiting Time (VT)	How do you feel about the visiting time of locations in the route?
Transition Time (TT)	Do you think the transition time between locations is rational?
Towards Destination (TD)	As traveling along the route, does it guide you to the destination?
Overall Acceptance (OA)	If you are a traveler, do you want to adopt this route?

we vary the parameter β for *Guidance Search* and examine the corresponding approximation ratio. In general, the approximation rate for our *Guidance Search* outperforms the other methods when $\beta \leq 0.8$ and reaches as high as 87% optimal when β is around 0.4. In addition, the results show that the proposed *Guidance Search* is actually an efficient search process (average route construction time $\leq$0.0043 seconds, which is 65,814 times faster than exhaustive search, which takes on average 283 seconds) due to the imposition of the heuristic function $h(l)$ on the best-first search mechanism.

6. USER STUDY

We conduct a user study to test whether the time-sensitive routes recommended by the proposed algorithm are rational, useful, and acceptable to users. For each city, we randomly select two paired locations as source-destination to construct the routes for a user study. The route length varies from 4 to 7. For comparison, we produce four routes for the user study. The first three routes are generated by our *Guidance Search* method with $\beta = 0$, $\beta = 0.5$, and $\beta = 1$. The fourth is the popularity-based route construction that sequentially selects the most popular neighbors and the corresponding most popular visiting time slots. We invited 10 people to conduct the user study. The evaluation criteria for the user study includes the location popularity, visiting order, visiting time, transition time, whether moving toward destination, and the overall acceptance of the recommended route (see Table III for details). We asked each user to assign a 0–5 score to each criterion/question for each constructed route. Higher scores represent better satisfaction for the criteria. Each user provided 6 (criteria) $*$ 3(cities) $*$ 2(routes) $*$ 4(models) $=$ 144 scores. Finally, we report the average value for each criterion in Figure 25.

In general, our method with $\beta = 0.5$ produced scores greater than 4 for all the criteria. For the case of $\beta = 0$, it also received high scores for LP, VO, VT, and TT. However, it does not consider the distance between current location and destination and therefore cannot guide users toward a destination. The proposed method with $\beta = 1$ has the highest score for TD since it emphasizes the heuristics that utilize destination information. However, its scores for other criteria are lower than those of $\beta = 0$ and $\beta = 0.5$. The baseline method, the popularity-based route, ensures that the

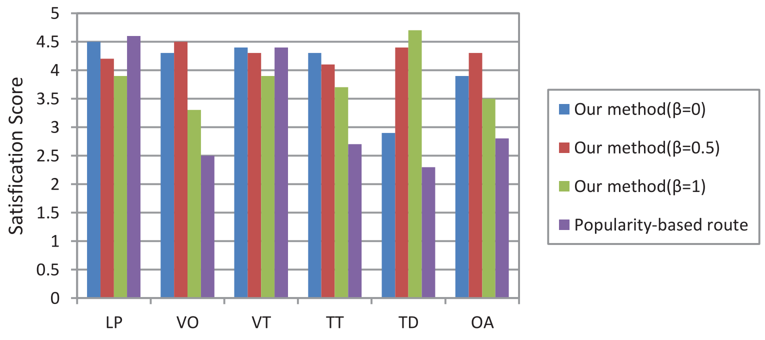
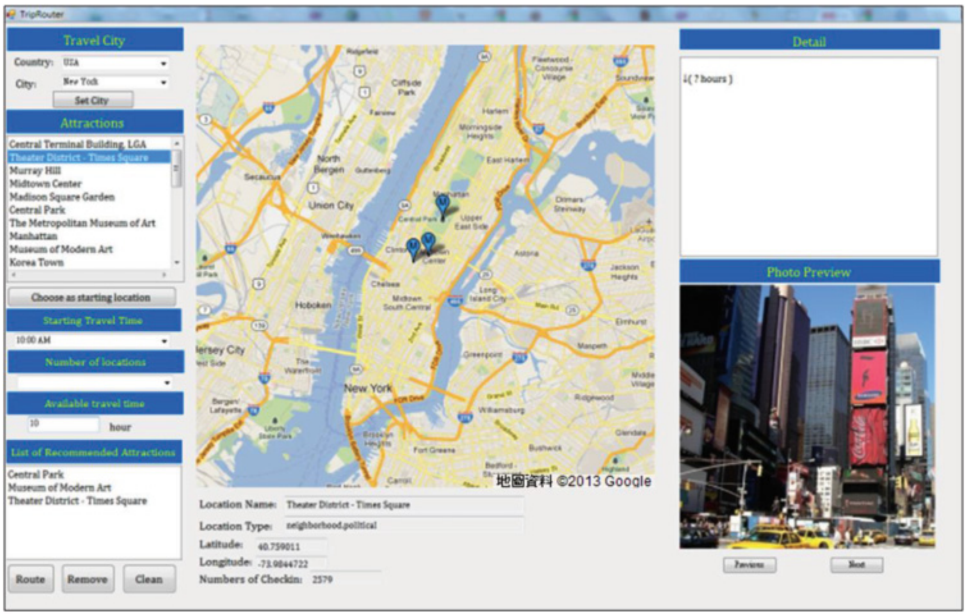


Fig. 25. Results of subjective evaluation for four methods.



Country: USA
City: New York

Central Terminal Building, LGA
Theater District - Times Square
Murray Hill
Midtown Center
Madison Square Garden
Central Park
The Metropolitan Museum of Art
Manhattan
Museum of Modern Art
Korea Town

10:00 AM

10 hour

Central Park
Museum of Modern Art
Theater District - Times Square

Location Name: Theater District - Times Square
Location Type: neighborhood,political
Latitude: 40.759011
Longitude: -73.9844722
Numbers of Checkin: 2579

i(? hours)

Fig. 26. The system interface of *TripRouter*.

visiting locations associated with visiting timestamps are popular, but the scores for the remaining four criteria are significantly lower. The user study gives us a quick view that the β can be set to 0.5 to achieve a better satisfaction score.

7. SYSTEM DEMONSTRATION

Using our model, we develop an online time-sensitive trip route recommendation system called *TripRouter*. The system snapshot is shown in Figure 26. In *TripRouter,* users first determine the city they intend to travel to and then select one location as their starting location, together with the starting time. *TripRouter* allows users to specify their destination and the desired number of places to visit during a trip. We list the three major functions of *TripRouter* as (a) time-sensitive route recommendation; (b) display of diverse information on locations and routes, such as location attributes, route statistics, and some geo-tagged photos obtained from Flickr; and (c) recommendation

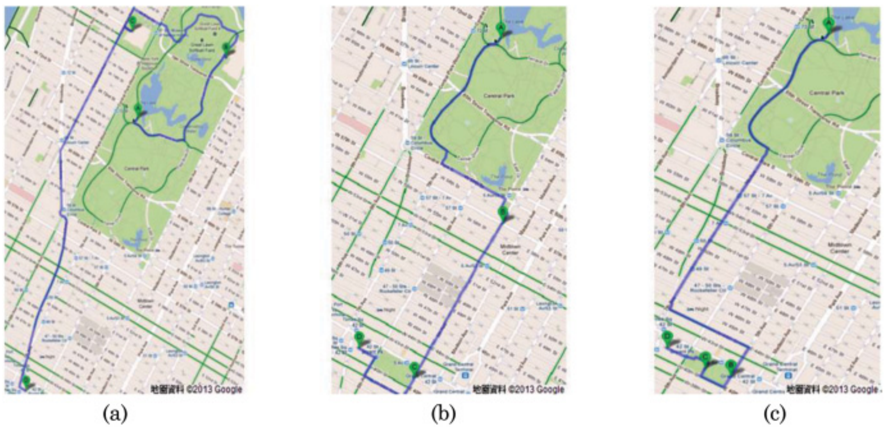


Fig. 27. Three recommended cases for varying β. (a) $\beta = 0$. (b) $\beta = 0.5$. (c) $\beta = 1$.

of the transportation mode by querying Google Map API according to mined transit time duration.

Here, we vary β as 0, 0.5, and 1 to show three recommended routes querying from Central Park to Time Square starting at 10 AM using the *Guidance Search* method, where the route length k is set to 4. The corresponding paths are highlighted with blue and shown in Figure 27(a), (b), (c).

Route 1. ($\beta = 0$): Central Park (10 AM) $\rightarrow$ Metropolitan Museum of Art (1 PM) $\rightarrow$ American Museum of Natural History (4 PM) $\rightarrow$ Time Square (9 PM).

Route 2. ($\beta = 0.5$): Central Park (10 AM) $\rightarrow$ 5th Avenue (1 PM) $\rightarrow$ New York Public Library (5 PM) $\rightarrow$ Times Square (7 PM).

Route 3. (**$\beta = 1$**): Central Park (10 AM) $\rightarrow$ New York Public Library (2 PM) $\rightarrow$ Bryant Park (7 PM) $\rightarrow$ Times Square (9 PM).

For $\beta = 0$, the program pays most attention to fitting the time-sensitive score, $g(l)$. Therefore, for every visiting location, it will try to find a fitting score considering proper visiting time, transit time, visiting order, and popularity. However, it can produce longer transit times to the final destination. The overall distance for route 1 is relatively large (3.9 miles, almost twice as much as the other cases). The $\beta = 0$ case provides routes for users who have much traveling time and care about the time-sensitivity of locations.

$\beta = 1$ finds a shorter route to the destination However, the visiting time, transit time, visiting order, and popularity are not considered because it does not include $g(l)$ to optimize. The overall distance for route 2 is 2.1 miles. $\beta = 1$ is more suitable for people who prefer to go to a destination earlier rather than later, but who still want to visit some places along the route.

The case of $\beta = 0.5$ is suitable for general travelers because it tries to strike a balance between $g(l)$ and $h(l)$. The overall distance for route 3 is 2.6 miles and the time-sensitive factors are considered.

Moreover, our system allows users to dynamically specify where they would like to go during the journey. Such user feedback functions can easily be achieved by combining the source and source-destination queries.

8. CONCLUSION

This article tries to address an important research question: can we measure and construct a time-sensitive trip route that considers the visiting time of each place for trip recommendations? Note that our approach is mostly data-driven, which assures that diverse results can be learned from different cities in which visiting patterns may vary with the differing cultures and characteristics of the city. Moreover, despite the fact that we emphasized check-in data in our experiments, in fact any kinds of route data can be exploited in our framework. For example, the GPS trajectory data can be used if we can perform some preprocessing in advance to identify the main locations, to compensate for some of the drawbacks from the check-in data (e.g., missing records, coarse granularity, and noise). Our future works are focusing on three directions: (a) improving the quality of the check-in routes by detecting missing locations in routes, (b) using a maximum likelihood estimator to accurately model the visiting time duration of a place and transportation time between places, and (c) exploiting collaborative filtering approaches to take advantage of user and location similarities.

REFERENCES

Yuki Arase, Xing Xie, Takahiro Hara, and Shojiro Nishio. 2010. Mining people's trips from large scale geo-tagged photos. In *Proceedings of ACM International Conference on Multimedia (MM'10)*. 133–142.

Jie Bao, Yu Zheng, and Mohamed F. Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL-GIS)*. 199–208.

Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. 2011. Discovering popular routes from trajectories. In *Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE'11)*. 900–911.

Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, and Xing Xie. 2011. Searching trajectories by locations: An efficiency study. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (SIGMOD'11)*. 255–266.

An-Jung Cheng, Yan-Ying Chen, Yen-Ta Huang, Winston H. Hsu, and Hong-Yuan Mark Liao. 2011. Personalized travel recommendation by mining people attributes from community-contributed photos. In *Proceedings of the 19th ACM International Conference on Multimedia (MM)*. 83–92.

Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*. 1082–1090.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms* (3rd ed.), MIT Press.

Yong Ge, Qi Liu, Hui Xiong, Alexander Tuzhilin, and Jian Chen. 2011. Cost-aware travel tour recommendation. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*. 983–991.

Hsun-Ping Hsieh, Cheng-Te Li, and Shou-De Lin. 2012. Recommending time-sensitive routes by exploiting large-scale check-in data. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing (Urbcomp'12)*. 55–62.

Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. 2010. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (*CIKM'10*). 579–588.

Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. 2011. Personalized travel package recommendation. In *Proceedings of the IEEE 11th International Conference on Data Mining (ICDM'11)*. 407–416.

Xin Lu, Changhu Wang, Jiang-Ming Yang, Yanwei Pang, and Lei Zhang. 2010. Photo2trip: Generating travel routes from geo-tagged photos for trip planning. In *Proceedings of the ACM International Conference on Multimedia (MM'10)*. 143–152.

Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo. 2010. Socio-spatial properties of online location-based social networks. In *Proceedings of the AAAI International Conference on Weblog and Social Media (ICWSM'10)*.

Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo. 2011. Exploiting place features in link prediction on location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*. 1046–1054.

Lu-An Tang, Yu Zheng, Jing Yuan, Jiawei Han, Alice Leung, Wen-Chih Peng, and Thomas La Porta. 2012. A framework of traveling companion discovery on trajectory data streams. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

Lu-An Tang, Yu Zheng, Xing Xie, Jing Yuan, Xiao Yu, and Jiawei Han. 2011. Retrieving k-nearest neighboring trajectories by a set of point locations. In *Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases (SSTD'11)*. 223–241.

Lu-An Tang, Yu Zheng, Jing Yuan, Jiawei Han, Alice Leung, Chih-Chieh Hung, and Wen-Chih Peng. 2012. On discovery of traveling companions from streaming trajectories. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering (ICDE'12)*. 186–197.

Ling-Yin Wei, Wen-Chih Peng, Bo-Chong Chen, and Ting-Wei Lin. 2010. PATS: A framework of pattern-aware trajectory search. In *Proceedings of the 11th IEEE International Conference on Mobile Data Management (MDM'10)*. 372–377.

Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*. 195–203.

Hyoseok Yoon, Yu Zheng, Xing Xie, and Woontack Woo. 2011. Social itinerary recommendation from user-generated digital trails. In *Personal and Ubiquitous Computing* 16, 5, 469–484.

Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*. 316–324.

Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xi, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL-GIS)*. 99–108.

Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th ACM International Conference on World Wide Web (WWW'09)*. 791–800.

Yu Zheng and Xing Xie. 2011. Learning travel recommendations from user-generated GPS traces. In *ACM Transactions on Intelligent Systems and Technology* 2, 1, Article 2.

Zhijun Yin, Liangliang Cao, Jiawei Han, Jiebo Luo, and Thomas Huang. 2011. Diversified trajectory pattern ranking in geo-tagged social media. In *Proceedings of the 11th SIAM International Conference on Data Mining (SDM'11).*

Received October 2012; revised September 2013; accepted November 2013