The following publication W. Li, J. Guan, X. Lian, S. Zhou and J. Cao, "Probabilistic Time-Constrained Paths Search over Uncertain Road Networks," in IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 399-414, 1 March-April 2018 is available at https://doi.org/10.1109/TSC.2016.2582692

Probabilistic Time-Constrained Paths Search over Uncertain Road Networks

Wengen Li, Jihong Guan, Xiang Lian, Shuigeng Zhou, *Member*, *IEEE* and Jiannong Cao, *Fellow*, *IEEE*

Abstract—With the proliferation of positioning technologies and GPS-enabled mobile devices, it has become very important to search for optimal paths to cover required *points of interest* (POIs, e.g., banks and restaurants) over road networks in many location-based services. In practice, however, traffic conditions are inherently uncertain and dynamically changing over time, which makes it rather challenging to provide accurate results for path queries based on travelling time. Inspired by this, we consider the practical settings of road networks and model them by *uncertain road networks (URNs)* on which the travelling time of each road is uncertain and captured by a set of travelling time samples. Then, we formalize the *probabilistic time-constrained path* (*PTP*) query over uncertain road networks to retrieve those paths that not only cover required POIs with constrained service time but also have the minimum travelling times in high confidence. We prove that *PTP* query problem is NP-hard. In order to answer *PTP* queries efficiently, we propose an efficient *PTP* query approach with effective pruning strategies regarding the time constraints on POIs and the probabilistic/rank requirements of queries. Extensive experiments on real road networks validate the efficiency and effectiveness of our *PTP* query approach.

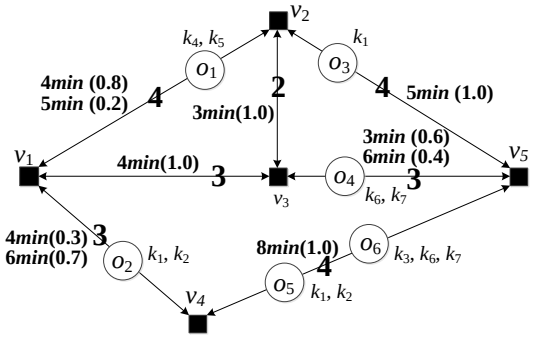
Index Terms—Location-based service, uncertain road network, service time-constrained POI, path search

1 Introduction

The path computation has always been an important part in service-oriented computing such as conventional map and navigation services. Particularly, with the increasing popularity of GPS-enabled mobile devices, computing a path over road networks to cover required services becomes a fundamental operation in many location-based applications and location-aware recommendation systems [1].

Figure 1 shows an example of road network on which edges and vertices represent road segments and intersection points of road segments, respectively. In addition, a set of *points of interest* (POIs), $o_1 \sim o_6$, such as restaurants, banks, and supermarkets (represented by white circles) reside over road segments and provide certain services described by keywords (k_i). A particular path query is to find paths between given source and destination locations that pass through a number of user-specified POIs on the road network with some criteria (e.g., small travelling distance or time). Assuming that v_4 is the current location of a user who has a destination of v_5, one example of path query is: "*Give me a driving path from v_4 to v_5 that sequentially passes a bank offering exchange (k_1, k_2) service and a supermarket (k_6) with the minimum travel time*".

While many existing works [2], [3], [4], [5] on path query assumed that the traffic on road networks was deterministic, however, this is not the real scenario. In practice, vehicle speeds collected on roads are uncertain and imprecise since different vehicles may have various possible speeds on the same road. Although there are some existing works studying the path query on uncertain road networks [6], [7], [8], [9], [10], however, they only compute probabilistic paths with the optimal travelling time between two locations and did not consider retrieving paths to cover POIs.



q=(v_4(*s*), v_5(*e*), 12:00, {<k_1, k_2; 30min>, <k_6; 20min>}, 1, 0.5)

Fig. 1. An example of uncertain road network, where circles are POIs with keyword description. The length and travel time of each edge are labeled beside. For example, the length of edge $e_{1,2} = (v_1, v_2)$ is 4 and the travelling time is 4 minutes in a probability of 0.8 and 5 minutes in a probability of 0.2. In addition, A *PTP* query q=$(v_4, v_5$, 12:00, $\{\langle k_1, k_2 : 30min\rangle, \langle k_6 : 20min\rangle\}, 1, 0.5)$ to find the optimal paths from v_4 to v_5 to cover keywords "k_1, k_2" and "k_6" sequentially is also illustrated.

In addition, it is often necessary to consider the time constraints, i.e., the service time, of POIs. For example, in Figure 1, one may want to find a path from v_4 to v_5, and drop by a bank (k_1) for some exchange (k_2) on the path. Starting from v_4, we consider two paths P_1=$(v_4, o_2, v_1, v_3, v_5)$ and P_2=(v_4, o_5, v_5) that cover o_2 and o_5, respectively. Assume that the departure time is 12:00 and the business hours of o_2 and o_5 are $\{$9:00-13:00, 14:00-18:00$\}$ and $\{$8:00-12:00, 13:00-17:00$\}$, respectively. Without considering

Wengen Li is jointly with the Department of Computer Science and Technology, Tongji University, Shanghai, China, and the Department of Computing, Hong Kong Polytechnic University, Hong Kong, China. E-mail: 8lwengen@tongji.edu.cn
Jihong Guan is the correspondence author and with the Department of Computer Science and Technology, Tongji University, Shanghai, China. E-mail: jhguan@tongji.edu.cn
Xiang Lian is with the Department of Computer Science, Kent State University, Ohio, USA. E-mail: xiang.lian@utrgv.edu
Shuigeng Zhou is with Shanghai key Lab of Intelligent Information Processing, and the School of Computer Science, Fudan University, Shanghai, China. E-mail: sgzhou@fudan.edu.cn
Jiannong Cao is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, China. E-mail: csjcao@comp.polyu.edu.hk

the service time constraints of banks, o_5 is better than o_2 since P_2 is much shorter than P_1. However, one will find that o_5 is closed (the earliest arrival time is 12:02 which will be discussed in Example 4) if s/he chooses P_2. In this case, bank o_2 is the better choice. Therefore, we can see that it is important to consider service time constraints of POIs for path query.

Inspired by the practical requirements of path query over road networks above, in this paper, we model road networks as uncertain road networks (*URNs*) over which the travelling time of each edge is captured by a set of time samples. Moreover, based on *URNs*, we propose a *probabilistic time-constrained path* (*PTP*) query to find the best paths that sequentially pass through a number of POIs containing user-specified query keywords (i.e., types of services provided by POIs), satisfy the service time constraints of POIs, and have the smallest travel times with high confidence.

PTP query is particularly useful in location-based services. When touring a city, a tourist may be unfamiliar with the service hours of POIs (the exhaustive manual checking is boring) and the traffic condition of the road network there. In this case, the *PTP* query can help the tourist find the paths that satisfy both keyword and service time constraints of POIs, and have the optimal travel times with high confidence.

However, efficient answering of *PTP* queries is rather challenging. Actually, as discussed in Section 3.3, *PTP* query is NP-hard.

In order to tackle the *PTP* problem, we design effective pruning strategies to filter out false alarms and obtain a small set of candidate paths. After that, a refinement step based on sampling is conducted to calculate the final query results. To summarize, in this paper, we make the following contributions.

- We model real road networks by uncertain road networks over which we formulate the *PTP* query (Section 3).
- We build efficient indices for both POIs and uncertain road networks, and design three effective pruning strategies, i.e., *time constraint pruning*, *probabilistic pruning*, and *travel time pruning*, w.r.t. POI time constraint, probability threshold and travel time requirement to reduce the search space of the *PTP* problem. In addition, we propose an efficient *PTP* query processing algorithm that utilizes the built indices and the proposed pruning strategies (Section 4).
- We demonstrate, through extensive experiments on different real data sets, the efficiency and effectiveness of our *PTP* approach (Section 5).

In addition, Section 2 reviews related work and Section 6 concludes the paper.

2 Related Work

In this section, we review the literature on *path queries to cover POIs over deterministic road networks* and *path queries on uncertain road networks*.

2.1 Path queries to cover POIs on deterministic road networks

Some works have been conducted to compute the optimal paths to cover required POIs on deterministic road networks. Sharifzadeh et al. [3] proposed the *optimal sequence route* query to cover specified types of POIs sequentially, i.e., a total order is imposed on the POIs to be visited. Levin et al. [5] proposed a similar query but considered partial order of POIs. In addition, some prior works also use keywords to describe required POIs instead of using the types of POIs. For example, Yao et al. [4] proposed to find the shortest path between two locations to cover some POIs that had all specified query keywords. In addition to the keywords requirement, Cao et al. [2] imposed a budget constraint over the path and used an objective score to optimize the final path answers. Compared with type description, keyword description is more flexible because it allows users to specify more detailed requirements for POIs.

The techniques in above work are designed for the deterministic road networks without any time constraints of POIs. Therefore, we cannot directly apply them to our *PTP* query which considers both of the uncertainty of road networks and the time constraints of POIs.

In addition, though the service time of POIs is considered in the nearest neighbor query in [11], the uncertainty of road networks is out of their consideration, which makes it different from our work.

2.2 Path queries on uncertain road networks

In the literature, the uncertainty of travel times over road networks are mainly represented with two methods, i.e., discrete probability distribution functions [6], [7], [9], [12], [13] and sample-based representation [8], [10], [12], [14], [15], [16]. With these representations, we have different definitions of the optimality of a path over uncertain road networks. Roughly, these existing works can be put into following two categories.

- Least expected travel time path query [6], [7], [9], [10], [12], [14]
- Reliable path query [8], [13], [17], [18], [19], [20], [21]

We detail these existing works as below.

Least expected travel time path query: Least expected travel path query retrieves the optimal path that has the minimum expected travel time. Both Hall [6] and Miller-hooks [7] used discrete probability distribution to represent the travel time of roads and compute the optimal path between two locations based on the expected travel time. [9] used a set of random travel times to capture the uncertainty of the travelling time of roads and proposed an efficient multicriteria A* algorithm to exactly determine the least expected time path in uncertain time-dependent networks. Yang et al. [10] leveraged a sampled-based representation scheme to construct an integer programming model for finding the a priori least expected time path. In their work, some reformulations are proposed to establish linear inequalities that can be easily dualized by a Lagrangian relaxation. A moment-based characterization for continuous link travel times through variance is presented in [22] which computes the expected travel time for a given departure time with an ensemble mean travel time over a number of days. Chen et al. [12] proposed three three definitions of optimality for finding the optimal path under an uncertain environment, i.e., expected value model, dependent-chance model, and chance-constrained model. They designed a simulation-based generic algorithm to compute the final result. Huang et al. [14] assumed that link travel times were uncertain and correlated over time and space. They defined a disutility function of travel time to evaluate the paths and returned the path with the minimum expected disutility as the optimal paths. Different from these works, our *PTP* query computes the probability that one path has the optimal travelling time directly instead of using the expected value.

Reliable path query: Reliable path query aims at finding these paths that can be travelled within certain time in high confidence. Hua et al. [8] applied random variables approximated by a set of samples to capture the uncertainty of road networks and proposed three types of probabilistic queries, i.e., a probabilistic path query, a weighted-threshold top-k path query, and a probability-threshold top-k path query. These queries can be computed with joint probability distribution directly due to the absence of POI constraints. Wu et al. [13] studied the problem of finding a priori optimal path to guarantee a given likelihood of arriving on-time over uncertain networks. Xing et al. [15] used Lagrangian substitution approach to estimate the lower bound of the most reliable path solution through solving a sequence of standard shortest path problems. Zhou et al. [16] discussed two models to evaluate the travel time robustness, i.e., absolute and α-percentile robust shortest path problems. They proposed a Lagrangian relaxation approach to deal with the problem.

All these works described above considered computing the optimal paths between two locations without the constraints of POIs, which thus cannot be directly applied to our *PTP* problem. Lian et al. [23] proposed the *trip planner query* to retrieve a route to traverse a set of points over probabilistic time-dependent road networks. This work, however, also did not consider the service time constraint of POIs. Thus the proposed techniques cannot be applied to tackle our *PTP* query directly.

3 Problem Statement

3.1 Data Models

3.1.1 Uncertain Road Networks

Uncertain road network model captures the inherent uncertainty of the travel times over road networks and is formally defined as below.

Definition 1. (*Uncertain Road Networks, URNs*) An uncertain road network is denoted by $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$, where:

- $V(\mathcal{G})$ is a set of vertices and each vertex $v_i \in V(\mathcal{G})$ represents an intersection of roads or a road terminal;
- $E(\mathcal{G})$ is a set of directed edges $e_{i,j}$ (i.e., the road segment from v_i to v_j) with length $|e_{i,j}|$, each associated with a random variable $\mathbf{t}(e_{i,j})$ representing the uncertain times for vehicles to travel from v_i to v_j,

where the distribution $\mathbf{t}(e_{i,j})$ is captured by T discrete random samples collected on edge $e_{i,j}$. □

Example 1. Figure 1 illustrates an example of uncertain road network with 5 vertices and 7 edges, near which lengths and travel times are marked beside. In particular, the travel time on edge $e_{1,2}$ is 4 minutes with probability 0.8 and 5 minutes with probability 0.2, which means 80% of T samples are 4 minutes while 20% are 5 minutes. For simplicity, in this example, only three edges are uncertain and the others are assumed to be deterministic.

In this paper, we assume that edges $e_{i,j}$ and $e_{j,i}$ have the same travel time, i.e., $\mathbf{t}(e_{i,j})$=$\mathbf{t}(e_{j,i})$. Nonetheless, our proposed approaches can be also applied to roads with asymmetric travel time. In addition, the travelling time samples for adjacent edges are collected independently in the same time period as discussed in Section 4.6. Thus, in some ways, the correlation between adjacent edges has been implicitly considered with respect to the temporal relationship.

TABLE 1
The possible worlds of URN G in Figure 1.

possible world	the travel time of edges	$Pr(w_i)$	$\mathcal{P}^1_{w_i}$
$w_1(\mathcal{G})$	$\mathbf{t}(e_{1,2})=4, \mathbf{t}(e_{1,4})=4, \mathbf{t}(e_{3,5})=3$	0.144	P_1
$w_2(\mathcal{G})$	$\mathbf{t}(e_{1,2})=4, \mathbf{t}(e_{1,4})=4, \mathbf{t}(e_{3,5})=6$	0.096	P_2
$w_3(\mathcal{G})$	$\mathbf{t}(e_{1,2})=4, \mathbf{t}(e_{1,4})=6, \mathbf{t}(e_{3,5})=3$	0.336	P_1
$w_4(\mathcal{G})$	$\mathbf{t}(e_{1,2})=4, \mathbf{t}(e_{1,4})=6, \mathbf{t}(e_{3,5})=6$	0.224	P_2
$w_5(\mathcal{G})$	$\mathbf{t}(e_{1,2})=5, \mathbf{t}(e_{1,4})=4, \mathbf{t}(e_{3,5})=3$	0.036	P_1
$w_6(\mathcal{G})$	$\mathbf{t}(e_{1,2})=5, \mathbf{t}(e_{1,4})=4, \mathbf{t}(e_{3,5})=6$	0.024	P_2
$w_7(\mathcal{G})$	$\mathbf{t}(e_{1,2})=5, \mathbf{t}(e_{1,4})=6, \mathbf{t}(e_{3,5})=3$	0.084	P_1
$w_8(\mathcal{G})$	$\mathbf{t}(e_{1,2})=5, \mathbf{t}(e_{1,4})=6, \mathbf{t}(e_{3,5})=6$	0.056	P_2

Obviously, the travel time $\mathbf{t}(e_{i,j})$ on edge $e_{i,j}$ is bounded by $[t^-(e_{i,j}), t^+(e_{i,j})]$, where $t^-(e_{i,j})$=$\min_{t \in \mathbf{t}(e_{i,j})} t$ and $t^+(e_{i,j})$=$\max_{t \in \mathbf{t}(e_{i,j})} t$. For a path $P = (v_1, v_2, \ldots, v_l)$, its travel time over uncertain road network $\mathcal{G}$ is $\mathbf{t}(P) = \sum_{i=1}^{l-1} \mathbf{t}(e_{i,i+1})$. The minimum and maximum travel time of P are $t^-(P) = \sum_{i=1}^{l-1} t^-(e_{i,i+1})$ and $t^+(P) = \sum_{i=1}^{l-1} t^+(e_{i,i+1})$, respectively.

3.1.2 Possible Worlds of URNs

Possible worlds [24] semantics are widely used in probabilistic databases, where each possible world is a materialized instance of the database that can appear in the real world. Similarly, the possible worlds of an uncertain road network $\mathcal{G}$ are defined below.

Definition 2. (*Possible Worlds of* $\mathcal{G}$) A *possible world*, $w(\mathcal{G})$, of uncertain road network $\mathcal{G}$ is a deterministic graph on which the random variable $\mathbf{t}(e_{i,j})$ of each edge $e_{i,j}$ takes a certain value $x(e_{i,j})$ and its appearance probability, $Pr\{w(\mathcal{G})\}$ is:

$$Pr\{w(\mathcal{G})\} = Pr\left\{\bigwedge_{\forall e_{i,j} \in E(\mathcal{G})} \mathbf{t}(e_{i,j}) = x(e_{i,j})\right\}. \quad (1)$$

All possible worlds of $\mathcal{G}$ are recorded as $\mathcal{W}(\mathcal{G})$. □

In Definition 2, the appearance probability of possible world, $Pr\{w(\mathcal{G})\}$, is the joint probability that each variable $\mathbf{t}(e_{i,j})$ takes value $x(e_{i,j})$. Considering the assumption that random variables $\mathbf{t}(e_{i,j})$ are independent, we have

$$Pr\{w(\mathcal{G})\} = \prod_{\forall e_{i,j} \in E(\mathcal{G})} Pr\{\mathbf{t}(e_{i,j}) = x(e_{i,j})\}. \quad (2)$$

Therefore, the number of possible worlds of $\mathcal{G}$ is exponential with respect to the number of edges of $\mathcal{G}$, i.e., $\prod_{e_{i,j} \in E} |\mathbf{t}(e_{i,j})|$ (= $T^{|E|}$ in the worst case), where $|\mathbf{t}(e_{i,j})|$ is the number of different values in $\mathbf{t}(e_{i,j})$.

Example 2. Table 1 presents 8 possible worlds, $w_1(\mathcal{G}) \sim w_8(\mathcal{G})$, of the uncertain road network in Figure 1. For example, possible world $w_1(\mathcal{G})$ takes $\mathbf{t}(e_{1,2})$=4, $\mathbf{t}(e_{1,4})$=4, and $\mathbf{t}(e_{3,5})$=3 (the travel times of other edges are deterministic and we can regard that the collected samples have the same value). From Figure 1, we have $Pr\{w_1(\mathcal{G})\}$=$Pr\{\mathbf{t}(e_{1,2})$=4$\}$ × $Pr\{\mathbf{t}(e_{1,4})$=4$\}$ × $Pr\{\mathbf{t}(e_{3,5}) = 3\} = 0.8 \times 0.3 \times 0.6 = 0.144$.

3.1.3 Service Time Constrained POI

On an uncertain road network $\mathcal{G}$, there exist many service time constrained points of interest (POI), which represent facility and service providers such as banks and hotels.

Definition 3. (*Service Time Constrained Points of Interest*) Each service time constrained *point of interest* (*POI*) is in the form of $o_i = (e, d, K, I)$, where

- e is the edge on which POI o_i resides;

- d is the offset distance of object o_i on edge e with regard to the start vertex of e;
- K is a set of keywords describing the properties of o_i or the services that o_i provides; and
- I is a set of time intervals $\{I_1, I_2, \dots\}$ in which o_i is valid for providing services.

The whole collection of POIs is denoted as $\mathcal{O}$. □

In Definition 3, the interval set $o_i.I = \{I_1, I_2, \dots\}$ specifies the service time constraints of o_i (e.g., the duration of service time) on a 24-hour cycle. For simplicity, in this paper, we use POIs and time constrained POIs interchangeably when the context is clear.

Example 3. Table 2 lists the details of the 6 POIs, $o_1 \sim o_6$, in Figure 1. For example, POI o_2 is on edge $o_2.e$=$e_{1,4}$, has offset distance $o_2.d$=$|o_2, v_1|$=1.5, contains 2 keywords $o_2.K$="k_1, k_2", and is associated with 2 time intervals $o_2.I$={9:00-13:00, 14:00-18:00}, i.e., valid time intervals of services.

TABLE 2
The POIs in Figure 1.

POIs	$o_i.e$	$o_i.d$	$o_i.K$	$o_i.I$
o_1	$e_{1,2}$	$\|o_1, v_1\|$ =3.0	k_4, k_5	{11:00-13:00, 17:00-21:00}
o_2	$e_{1,4}$	$\|o_2, v_1\|$ =1.5	k_1, k_2	{09:00-13:00, 14:00-18:00}
o_3	$e_{2,5}$	$\|o_3, v_2\|$ =1.0	k_1	{09:00-11:00, 13:00-16:00}
o_4	$e_{3,5}$	$\|o_4, v_3\|$ =1.0	k_6, k_7	{08:00-20:00}
o_5	$e_{4,5}$	$\|o_5, v_4\|$ =1.0	k_1, k_2	{08:00-12:00, 13:00-17:00}
o_6	$e_{4,5}$	$\|o_6, v_4\|$ =2.0	k_3, k_6, k_7	{08:00-21:00}

3.2 Probabilistic Time-constrained Path Query

A *probabilistic time-constrained path* (*PTP*) query is denoted by q=$(s, e, t_d, \mathcal{K}, h, \tau)$, where s and e are the start and end locations, respectively; t_d is the departure time; $\mathcal{K}$ specifies the query keywords along with the staying times at the corresponding POIs containing query keywords; h is an integer parameter w.r.t. the number of the returned paths; and $\tau \in (0, 1]$ is a probability threshold to ensure minimum travel time in high confidence. Particularly, $\mathcal{K}$={$\langle K_1, t_1\rangle$, ..., $\langle K_\psi, t_\psi\rangle$}, where K_i and t_i ($i = 1, \dots, \psi$) are keywords and the expected staying time at the i-th POI, respectively.

For simplicity, we use $P(s, e) = \langle s, o_{x_1}, \dots, o_{x_\psi}, e\rangle$ to represent any path from s to e that sequentially covers ψ POIs with the ψ sets of query keywords, i.e., $K_i \subseteq o_{x_i}.K$. All these paths are recorded as $\mathcal{P}(s, e)$. Now we have the definition of *PTP* query as below.

Definition 4. (Probabilistic Time-Constrained Path, *PTP*, Query) Given an uncertain road network $\mathcal{G}$, the objective of a *PTP* query q is to retrieve a subset, $\mathcal{P}_h(s, e)$, of paths from $\mathcal{P}(s, e)$, i.e.,

$$\mathcal{P}_h(s, e) = \{P | P \in \mathcal{P}(s, e) \wedge Pr_h\{P\} \geq \tau\} \quad (3)$$

where

$$Pr_h\{P\} = \left\{ \sum_{\forall w(\mathcal{G}), P \in \mathcal{P}_w^h(s,e)} Pr\{w(\mathcal{G})\} \right\} \quad (4)$$

and $\mathcal{P}_w^h(s, e)$ are the top-h optimal paths in possible world $w(\mathcal{G})$ such that for $\forall P \in \mathcal{P}_w^h(s, e)$

$$K_i \subseteq o_{x_i}.K, i = 1, 2, \dots, \psi \quad (5)$$

$$[a(o_{x_i}), a(o_{x_i}) + q.t_i] \subseteq o_{x_i}.I \quad (6)$$

$$\forall P' \in \mathcal{P}(s, e) \backslash \mathcal{P}_w^h(s, e), t_w(P) \leq t_w(P') \quad (7)$$

where $t_w(\cdot)$ is the travel time of a path in possible world $w(\mathcal{G})$. □

In Definition 4, $Pr_h\{P\}$ is the probability that path P is among the top-h optimal paths in all possible worlds. Therefore, *PTP* query retrieves those paths, $\mathcal{P}_h(s, e)$, that are top-h time-constrained paths in all possible worlds $\mathcal{W}(\mathcal{G})$ with probability not less than threshold τ. In other words, the returned *PTP* paths satisfy keyword/time constraints on POIs, and have top-h smallest travel times with high confidence.

Example 4. Figure 1 shows an example of a *PTP* query q, where s=v_4, e=v_5, t_d=12:00, $\mathcal{K}$={$\langle$"k_1, k_2", $30min\rangle$, $\langle$"k_6", $20min\rangle$}, h=1, and τ=0.5. There are two POIs (o_2 and o_5) containing "k_1, k_2" and two POIs (o_4 and o_6) containing "k_6", which produce four candidate paths, i.e., P_1=(v_4, o_2, o_4, v_5), P_2=(v_4, o_2, o_6, v_5), P_3=(v_4, o_5, o_4, v_5), and P_4=(v_4, o_5, o_6, v_5). The arrival time at o_5 is 12:00+$\mathbf{t}(e_{4,5}) \cdot \frac{o_5.d}{|e_{4,5}|}$=12:02, which is out of the time intervals of o_5. Therefore, both P_3 and P_4 cannot satisfy the service time constraints of o_5 and are pruned.

Considering P_1 in possible world $w_1(\mathcal{G})$, the arrival time at o_2 is 12:00+$\mathbf{t}(e_{4,1}) \cdot \frac{|e_{4,1}| - o_2.d}{|e_{4,1}|}$=12:02 which is within the time interval of o_2, i.e., $o_2.I$={9:00-13:00, 14:00-18:00}. After staying at o_2 for 30 minutes, the arrival time at o_4 is 12:39 which satisfies the time constraint of o_4, i.e., {8:00-20:00}. Finally, we have $\mathbf{t}(P_1)$=11. Similarly, we have $\mathbf{t}(P_2)$=12 in $w_1(\mathcal{G})$. Therefore, we obtain $\mathcal{P}_{w_1}^1(s, e) = \{P_1\}$.

The top-1 time-constrained paths, $\mathcal{P}_w^1(s, e)$, in other possible worlds can be computed in the same way and the results are listed in Table 1. Then, we have $Pr_h\{P_1\}$= 0.144+0.336+0.036+0.084=0.6>τ (=0.5) and $Pr_h\{P_2\}$= 0.096+0.224+0.024+0.056=0.4 <τ (=0.5). Therefore, P_1 is the only *PTP* query answer. □

3.3 Problem Complexity

The objective of *PTP* query is to find a subset of paths $\mathcal{P}_h(s, e) \subseteq \mathcal{P}(s, e)$ such that these path are among the top-h optimal paths in high confidence. Without loss of generality, we assume a specific case of *PTP* query where $q.h = 1$ and the appearance probabilities of all possible worlds are equal.

We can prove that *PTP* query problem is NP-hard by a reduction from partial covering problem [25] which has set-cover problem as one of its specific cases. Given the global set $U = \{e_1, e_2, \cdots, e_m\}$ and a set of subsets $S = \{U_1, U_2, \cdots, U_n\}$ where $U_i \subseteq U$, a partial covering problem find whether there exist k subsets $S' \subseteq S$ such that S' covers at lest x elements of U.

First, we define the decision problem of *PTP* query problem as below.

Definition 5 (Decision-PTP). A Decision-PTP problem is to determine whether there exists k paths in $\mathcal{P}(s, e)$ such that these k paths contain the top-1 path in at least x possible worlds.

Then, we have the following theorem.

Theorem 1. PTP query problem is NP-hard.

Proof: Assume $\mathcal{P}(s, e) = \{P_1, P_2, \cdots, P_n\}$ and $\mathcal{W}(\mathcal{G}) = \{w_1, w_2, \cdots, w_m\}$. Now we convert a partial covering instance to an instance of PTP-Decision problem. We regard each subset $U_i \in S$ as a path P_i and each element $e_j \in U$ as a possible world w_j. If $e_j \in U_i$, we have the travelling time of path P_i in possible world w_j, $t_{w_j}(P_i) = 1$, otherwise $t_{w_j}(P_i) = +\infty$. Thus, we can find k subsets S' from S to cover at least x elements of U if and only

TABLE 3
Frequently used notations and their meanings.

notation	meaning
$\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$	uncertain road network
v_i	a vertex
$e_{i,j}$ ($\|e_{i,j}\|$)	an edge (and its length) from v_i to v_j
$\mathbf{t}(e_{i,j})$	the traveling time of $e_{i,j}$
$t^-(e_{i,j})$ ($t^+(e_{i,j})$)	the minimum (maximum) traveling time of $e_{i,j}$
$\mathbf{t}(P)$	the traveling time of a path P
$t^-(P)$ ($t^+(P)$)	the minimum (maximum) travelling time of path P
$w(\mathcal{G})$	a possible world of $\mathcal{G}$
$\mathcal{W}(\mathcal{G})$	all possible worlds of $\mathcal{G}$
$o_i = \{e, d, K, I\}$	a POI
$\mathbf{a}(o_i)$	the arrival time at POI o_i
$\mathbf{a}_c(o_i)$	the constrained arrival time at POI o_i
$P(s, e)$	a path from s to e covering required POIs
$Q = (s, e, t_d, \mathcal{K}, h, \tau)$	a *PTP* query
$\mathcal{P}_w^h(s, e)$	top-h time-constrained paths in $w(\mathcal{G})$

if we can find k paths that are the top-1 optimal path in at least x possible worlds. We can prove this from the following two sides.

- Assume that we find k subsets S' from S such that S' covers at least x elements of U. Therefore, in each of the corresponding x possible worlds, the corresponding k paths must contain the top-1 optimal paths considering that $t_{w_j}(P_i) = 1$ if $e_j \in U_i$, otherwise $t_{w_j}(P_i) = +\infty$. Thus, we have at least x possible worlds of $\mathcal{W}(\mathcal{G})$ in which the corresponding k paths have the top-1 path.
- Assume that we find k paths such that they contain the top-1 path in at least x possible worlds. In each of these possible worlds, we must have that the corresponding element is covered by the corresponding k subsets since we can always find a path with travelling time of 1, i.e., the top-1 path. Thus, there are at least x elements are covered.

Now we can conclude that PTP-Decision problem is NP-hard. □

Therefore, the efficient answering of *PTP* queries is challenging due to an exponential number of possible worlds. One straightforward method is to enumerate all possible worlds $\mathcal{W}(\mathcal{G})$, obtain $\mathcal{P}_w^h(s, e)$ under each $w(\mathcal{G})$, and aggregate all obtained results to generate the *PTP* query result. However, this method is rather inefficient due to exponential number of possible worlds that are computationally expensive to materialize. Therefore, in this paper, we propose to process *PTP* query in a two-phase fashion. First, we propose effective pruning strategies to filter out false alarms without enumerating all possible worlds and produce a small set of candidate paths. Second, a refinement step based on Monte Carlo theory [26] is conducted to compute the final result.

Table 3 summarizes the frequently used notations and their meanings in this paper.

4 Solutions

4.1 Overview of the solution

Figure 2 illustrates the flow chart of our proposed solution. In a nutshell, the solution covers the following three tasks.

- First, we generate the path search space according to the *PTP* query, POIs and uncertain road network (*URN*).
- Second, efficient pruning strategies are designed to filter out infeasible paths to obtain a small set of candidate paths.
- Third, we conduct a refinement step to refine the candidate routes and get the final results.

The details of these tasks are elaborated in the following sections. In section 4.2, we first build indices for POIs and road network, respectively, to accelerate query processing. Section 4.3 talks about the generation of search space. In section 4.4, the details of three pruning strategies are presented. Section 4.5 talks about how to compute candidate routes with the proposed pruning strategies and how to refine the obtained candidate routes to generate the final results as well. In addition, we also discuss about the updates of travelling time samples in uncertain road network in Section 4.6.

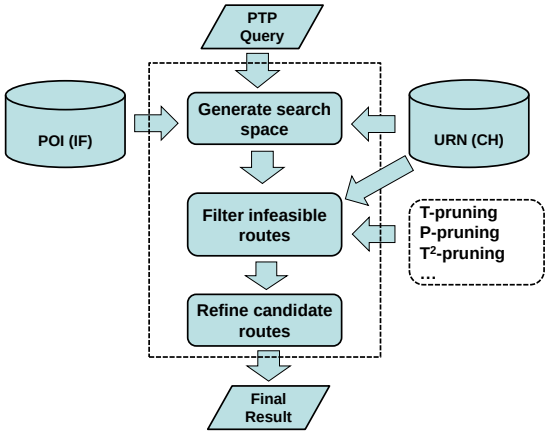


Fig. 2. The flow chart of the proposed solution.

4.2 Indexing POIs and Road Networks

In order to efficiently retrieve those POIs that satisfy the query keywords during the *PTP* query answering, we utilize an inverted file [27] to index keywords as well as the relevant POIs. We denote $Inv(k)$ as the set of POIs containing k, with cardinality $|Inv(k)|$. Given a set of query keywords K_i=$\{k_1, \ldots, k_l\}$, the set $Inv(k_1) \cap \cdots \cap Inv(k_l)$ contains POIs that have all query keywords in K_i. Note that, to further improve the query efficiency of accessing query keywords, we also index keyword entries in the inverted file with a B^+-tree, where the keys in the B^+-tree are unique values transformed from keywords.

In addition, in order to efficiently compute the shortest path between two POIs, we index the road networks with Contraction Hierarchy(CH) [28], which imposes a total order on the vertices V according to their importance[1], and pre-computes the distances among vertices based on this order. Initially, the least important vertex, say v_i, is removed and its adjacent vertices are checked. If the shortest path of a pair of adjacent vertices, say v_j and v_k, passes through v_i, a virtual edge with the weight of the shortest distance between v_j and v_k is introduced. For example, in Figure 1, v_4 (which has the smallest degree) is first removed and no virtual edges are introduced because the shortest path (v_1, v_3, v_5) between v_1 and v_5 does not pass v_4. Then, the second least important vertex is removed and processed in the same way. By doing this repeatedly until all vertices are processed or the entire road network are reduced to an appropriate scale, we can build a hierarchical index for the road network. During the shortest distance/path computation, bidirectional Dijkstra's algorithm with some modifications, i.e., only those unvisited vertices (adjacent to current vertex) ranking higher (lower) than the current expanding vertex are considered during the forward (backward) traversal, is employed. Thus, numerous vertices are avoided, which greatly accelerates the query processing. Note that, CH index is adopted

1. We use the degrees of vertices in this paper because vertices with large degrees usually correspond to busy traffic points on road networks.

because we only consider the shortest distance path between each pair of POIs and the objective of *PTP* query is to find a sequence of required POIs with the least travelling time in high confidence.

4.3 Generating search space

Given a *PTP* query q=$(s, e, t_d, \mathcal{K}, h, \tau)$, those relevant POIs with respect to the query keywords are retrieved. For $K_i \in \mathcal{K}$ $(i = 1, \ldots, \psi)$, all POIs containing K_i are computed by using the inverted index as discussed in Section 4.2. If no ambiguity, we also record those POIs containing all keywords in K_i as $Inv(K_i)$ and use $|Inv(K_i)|$ to represent its cardinality. Then we have the total number of combinations $\prod_{i=1}^{\psi} |Inv(K_i)|$ which increases exponentially with respect to the number of required POIs. Therefore, to enumerate all these combinations will be time-consuming. Our objective is to greatly reduce the size of these combinations with novel pruning strategies as detailed in Section 4.4.

4.4 Pruning Mechanisms

In this section, we will design effective pruning methods to reduce the *PTP* search space, which can produce a small set of candidate paths for further refinement. Concretely, we first present *time constraint pruning* (**T-pruning**) to prune those paths that violate the time constraints of POIs. Then, *probabilistic pruning* (**P-pruning**) is proposed to enable the pruning by utilizing the probability threshold τ. Furthermore, since only top-h time-constrained paths are requested under possible worlds, we also propose the *travel time pruning* (**T^2-pruning**) that directly filters out false alarms that cannot be top-h time constrained paths.

4.4.1 Time Constraint Pruning

Recall that $P(s, e)$ represents any path that sequentially passes POIs o_{x_1}, o_{x_2}, ..., and o_{x_ψ}, as illustrated in Figure 3, where o_{x_i} $(i = 1, \ldots, \psi)$ represents any POI containing query keywords K_i. Thus, if $P(s, e)$ does not satisfy the time constraints, $o_{x_i}.I$, of some POI o_{x_i} for any possible world, then $P(s, e)$ cannot be a *PTP* answer (i.e., $Pr_h\{P(s, e)\} = 0$), and thus should be pruned.

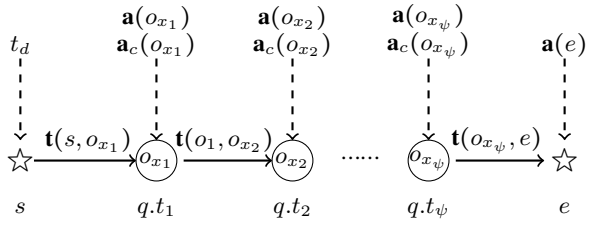


Fig. 3. The arrival times at different POIs along path $P(s, e)$.

Specifically, let $\mathbf{a}(o_{x_i})$ be the arrival time at the i-th POI o_{x_i}, and we have:

$$\mathbf{a}(o_{x_i}) = t_d + \mathbf{t}(P(s, o_{x_i})) + \sum_{j=1}^{i-1} q.t_j \tag{8}$$

where $P(s, o_{x_i})$ is the subpath of $P(s, e)$ that from s to o_{x_i}, $\mathbf{t}(P(s, o_{x_i}))$ is the travel time of $P(s, o_{x_i})$ and $q.t_j$ $(j$=$1, \ldots, i-1)$ is the staying time at POI o_{x_j}. Due to the uncertainty of $\mathbf{t}(P(s, o_{x_i}))$, $\mathbf{a}(o_{x_i})$ is uncertain and falls into the time interval $[a^-(o_{x_i}), a^+(o_{x_i})]$, where $a^-(o_{x_i})$ and $a^+(o_{x_i})$ are the earliest and latest arrival times at o_{x_i}, respectively. Then, we denote the *constrained arrival time*, $\mathbf{a}_c(o_{x_i})$, as the set of possible arrival times within the service time of o_{x_i}, i.e.,

$$\mathbf{a}_c(o_{x_i}) = \{x | x \in \mathbf{a}(o_{x_i}) \wedge [x, x + q.t_i] \subseteq o_{x_i}.I\} \tag{9}$$

Intuitively, constrained arrival time $\mathbf{a}_c(o_{x_i})$ computes arrival times such that they and their staying times are within the time intervals of o_{x_i}. Assuming that the interval of $\mathbf{a}_c(o_{x_i})$ is $[a_c^-(o_{x_i}), a_c^+(o_{x_i})]$, we have $[a_c^-(o_{x_i}), a_c^+(o_{x_i})] \subseteq [a^-(o_{x_i}), a^+(o_{x_i})]$. Then, we have the following lemma about **T-Pruning**:

Lemma 1. (Time Constraint Pruning, **T-Pruning**) Given an *URN* $\mathcal{G}$ and a *PTP* query q, if a path, $P(s, e)$, passes some POI $o_{x_i}(1 \leq i \leq \psi)$ in $\mathcal{G}$, and $|\mathbf{a}_c(o_{x_i})| = 0$, then $P(s, e)$ can be safely pruned.

Proof: According to the lemma assumption that $|\mathbf{a}_c(o_{x_i})|$=0, $P(s, e)$ cannot be among the top-h time-constrained paths $\mathcal{P}_w^h(s, e)$ under any possible world $w(\mathcal{G})$ (due to the violation of POI time constraints). Thus, the *PTP* probability, $Pr_h\{P(s, e)\}$ is equal to 0, which is smaller than the nonzero τ. Therefore, $P(s, e)$ cannot be a *PTP* query answer and can be pruned. □

Note that, both arrival and departure times at POIs o_{x_i} must fall into some single time interval $o_{x_i}.I$ so that the staying time $q.t_i$ can be satisfied without an interruption.

To conduct T-pruning, we need to compute $\mathbf{a}_c(o_{x_i})$ efficiently. With Eqs. (8) and (9), the arrival time and constrained arrival time at each POI can be computed iteratively as follows.

$$\mathbf{a}(o_{x_i}) = \begin{cases} t_d + \mathbf{t}(s, o_{x_i}), & i = 1 \\ \mathbf{a}_c(o_{x_{i-1}}) + q.t_{i-1} + \mathbf{t}(o_{x_{i-1}}, o_{x_i}), & i > 1 \end{cases} \tag{10}$$

where $\mathbf{t}(x, y)$ is the travel time of the shortest path from x to y. In addition, $\mathbf{a}(q.e) = \mathbf{a}_c(o_{x_\psi}) + q.t_\psi + \mathbf{t}(o_{x_\psi}, q.e)$.

In general, however, it is not trivial to enumerate all values of $\mathbf{a}(o_{x_i})$ and $\mathbf{a}_c(o_{x_i})$ due to their numerous instances. Instead, to avoid costly computations, we compute lower/upper bounds for $\mathbf{a}(o_{x_i})$ and $\mathbf{a}_c(o_{x_i})$. To obtain lower/upper bounds, $a^-(o_{x_i})$ and $a^+(o_{x_i})$, of the arrival time $\mathbf{a}(o_{x_i})$ for POI o_{x_i}, we aim to obtain lower/upper bounds of $\mathbf{t}(o_{x_{i-1}}, o_{x_i})$. Denote the bounds of travel time $\mathbf{t}(x, y)$ as $[t^-(x, y), t^+(x, y)]$. Initially, for the subpath from s to o_{x_1} on path $P(s, e)$, we have $a^-(o_{x_1}) = t_d + t^-(s, o_{x_1})$ and $a^+(o_{x_1}) = t_d + t^+(s, o_{x_1})$ Accordingly, the constrained arrival time interval at o_{x_1} is computed as:

$$[a_c^-(o_{x_1}), a_c^+(o_{x_1})] = [a^-(o_{x_1}), a^+(o_{x_1})] \cap o_{x_1}.I', \tag{11}$$

where $o_{x_1}.I'$ is computed by subtracting $q.t_1$ from the upper value of each sub-interval of $o_{x_1}.I$ so that the required staying time $q.t_1$ can be satisfied. For example, if $o_{x_1}.I$={8-12,13-16} and $q.t_1$=1, we have $o_{x_1}.I'$={8-11,13-15}.

Similarly, for the i-th POI o_{x_i} on $P(s, e)$ $(2 \leq i \leq \psi)$, we have:

$$[a_c^-(o_{x_i}), a_c^+(o_{x_i})] = [a^-(o_{x_i}), a^+(o_{x_i})] \cap o_{x_i}.I'$$

where

$$a^-(o_{x_i}) = a_c^-(o_{x_{i-1}}) + q.t_{i-1} + t^-(o_{x_{i-1}}, o_{x_i}),$$
$$a^+(o_{x_i}) = a_c^+(o_{x_{i-1}}) + q.t_{i-1} + t^+(o_{x_{i-1}}, o_{x_i}).$$

With the bounds of constrained arrival times at POIs, a path $P(s, e)$ can be pruned if there exists some POI o_{x_i} such that $[a^-(o_{x_i}), a^+(o_{x_i})] \cap o_{x_i}.I' = \emptyset$, i.e., $|\mathbf{a}_c(o_{x_i})| = 0$. Thus, with **T-pruning**, those paths that violate the time constraints of POIs can be pruned to reduce the search space.

4.4.2 Probabilistic Pruning

The basic idea of probabilistic pruning is as follows. Intuitively, if an upper bound of the *PTP* probability, $Pr_h\{P(s,e)\}$ for $P(s,e)$ is smaller than the specified probability threshold τ, then $P(s,e)$ can be pruned.

Lemma 2. (Probabilistic Pruning, **P-Pruning**) Denote the upper bound of the *PTP* probability, $Pr_h\{P(s,e)\}$, as $Pr_h^+\{P(s,e)\}$. If $Pr_h^+\{P(s,e)\}<\tau$ holds, then $P(s,e)$ can be safely pruned.

Proof: It can be easily pruned by the inequality transition of $Pr_h\{P(s,e)\} \le Pr_h^+\{P(s,e)\}$ and $Pr_h^+\{P(s,e)\} < \tau$. □

Now, the task turns out to be the derivation of a probability upper bound, $Pr_h^+\{P(s,e)\}$. According to the definition of the *PTP* query, a *PTP* path should satisfy the time constraints of POIs and have the top-h smallest travel times with a probability above τ. Therefore, for path $P(s,e)$, we need to consider two factors, i.e., the time constraints of POIs and ranking probability (having the top-h smallest travel time). Note that, for path $P(s,e)$, satisfying the time constraints of POIs is the precondition of considering its ranking probability. Therefore, we will first use the constraint-based probability upper bound of $P(s,e)$ to prune this path. If $P(s,e)$ survives, its ranking probability will be further considered to compute another rank-based probability upper bound to prune this path.

A. Constraint-based probability upper bound

According to Eq. (8), the arrival time at each POI o_{x_i}, $\mathbf{a}(o_{x_i})$, is a set of discrete values within $[a^-(o_{x_i}), a^+(o_{x_i})]$. In general, we have $[a_c^-(o_{x_i}), a_c^+(o_{x_i})] \subseteq [a^-(o_{x_i}), a^+(o_{x_i})]$, i.e., only a subset of values in $\mathbf{a}(o_{x_i})$ satisfy the time constraints of o_{x_i}. We assume that $E(o_{x_i})$ is the event that $p(s,e)$ satisfies the time constraints of o_{x_i}, i.e.,

$$Pr\{E(o_{x_i})\} = Pr\{a_c^-(o_{x_i}) \le \mathbf{a}(o_{x_i}) \le a_c^+(o_{x_i})\}$$

Hence, we have the constraint-based probability, $Pr_c\{P(s,e)\}$, i.e., the probability that $P(s,e)$ satisfies the time constraints at all ψ POIs, as below.

$$Pr_c\{P(s,e)\} = Pr\left\{\bigwedge_{i=1}^{\psi} E(o_{x_i})\right\}. \tag{12}$$

If $Pr_c\{P(s,e)\} \ge \tau$, we call $P(s,e)$ an *effective path*. Then, we have the following lemma.

Lemma 3. Given a subpath $P(s,o_{x_i})$ of $P(s,e)$, we have:

$$\begin{aligned} Pr_c\{P(s,e)\} &\le Pr_c\{P(s,o_{x_i})\} \\ &\le \min_{j=1,\ldots,i} Pr\{a_c^-(o_{x_j}) \le \mathbf{a}(o_{x_j}) \le a_c^+(o_{x_j})\} \end{aligned}$$

Proof: we have

$$Pr_c\{P(s,o_{x_i})\} = Pr\left\{\bigwedge_{j=1}^{i} E(o_{x_j})\right\}. \tag{13}$$

Given $i \le \psi$, $Pr\left\{\bigwedge_{j=1}^{\psi} E(o_{x_j})\right\} \le Pr\left\{\bigwedge_{j=1}^{i} E(o_{x_j})\right\}$ holds because $\bigwedge_{j=1}^{\psi} E(o_{x_j})$ has more events (i.e., more constraints) than $\bigwedge_{j=1}^{i} E(o_{x_j})$. Therefore, by combining Eqs. (12) and (13), we have $Pr_c\{P(s,e)\} \le Pr_c\{P(s,o_{x_i})\}$. Furthermore, we have:

$$\begin{aligned} Pr_c\{P(s,e)\} &\le Pr_c\{P(s,o_{x_i})\} = Pr\{\bigwedge_{j=1}^{i} E(o_{x_j})\} \\ &= Pr\{\bigwedge_{j=1}^{i} a_c^-(o_{x_j}) \le \mathbf{a}(o_{x_j}) \le a_c^+(o_{x_j})\} \\ &\le \min_{j=1,\ldots,i} Pr\{a_c^-(o_{x_j}) \le \mathbf{a}(o_{x_j}) \le a_c^+(o_{x_j})\}. \end{aligned}$$

Hence, Lemma 3 holds. □

Lemma 3 indicates that a longer path has a lower constraint-based probability and an upper bound of the *PTP* probability $Pr_h\{P(s,e)\}$ can be computed by $\min_{j=1,\ldots,i} Pr\{a_c^-(o_{x_j}) \le \mathbf{a}(o_{x_j}) \le a_c^+(o_{x_j})\}$. To compute this upper bound, we need to calculate $Pr\{a_c^-(o_{x_j}) \le \mathbf{a}(o_{x_j}) \le a_c^+(o_{x_j})\}$ for each POI o_{x_j}. A straightforward method is to compute the probabilistic distribution of $\mathbf{a}(o_{x_j})$, and aggregate these samples that are within $[a_c^-(o_{x_j}), a_c^+(o_{x_j})]$ to obtain the constraint-based probability. However, this incurs a very high computation cost due to exponential number of possible instances. Instead of computing $\mathbf{a}(o_{x_j})$ directly, we compute an upper bound for $Pr\{a_c^-(o_{x_j}) \le \mathbf{a}(o_{x_j}) \le a_c^+(o_{x_j})\}$ without computing the actual probability distribution.

We denote the *cumulative distribution function* (CDF) of $\mathbf{t}(p(s,o_{x_i}))$ as $F_P(z)$=$Pr\{\mathbf{t}(P(s,o_{x_i})) \le z\}$ whose lower and upper bounds are accordingly assumed to be $F_P^-(z)$ and $F_P^+(z)$, respectively. Then, we have:

$$Pr\{a_c^-(o_{x_j}) \le \mathbf{a}(o_{x_j}) \le a_c^+(o_{x_j})\} \le F_P^+(b) - F_P^-(a) \tag{14}$$

where a=$a_c^-(o_{x_j}) - t_d - \sum_{l=1}^{j-1} q.t_l$, b=$a_c^+(o_{x_j}) - t_d - \sum_{l=1}^{j-1} q.t_l$, and $q.t_l$ is the staying time at POI o_l. Eq. (14) can be proved by the following derivation:

$$\begin{aligned} Pr\{a_c^-(o_{x_j}) \le \mathbf{a}(o_{x_j}) \le a_c^+(o_{x_j})\} &= Pr\{a \le \mathbf{a}(o_{x_j}) - t_d - \sum_{l=1}^{j-1} q.t_l \le b\} \\ &= Pr\{a \le \mathbf{t}(P(s,o_{x_j})) \le b\} \\ &\le F_P^+(b) - F_P^-(a). \end{aligned}$$

Next, we discuss how to compute the bounds of $F_P(z)$, i.e., $F_P^-(z)$ and $F_P^+(z)$. Assume the CDF of $\mathbf{t}(e)$ is $F_e(x)$=$Pr\{\mathbf{t}(e) \le x\}$. Generally, $F_e(x)$ can be computed easily, since the distribution of $\mathbf{t}(e)$ is known to be represented by samples. However, $F_P(z)$ is difficult to compute due to the unknown distribution of $\mathbf{t}(p(s,o_{x_i}))$. Therefore, our basic idea is to generate bounds of $F_P(z)$ based on $F_e(x)$ instead of the detailed distribution of $\mathbf{t}(P(s,o_{x_i}))$. To this end, we set a certain threshold for $\mathbf{t}(e)$ ($e \in P(s,o_{x_i})$) according to z and $\mathbf{t}(e)$ such that the summation of these thresholds is less than z. Further, these thresholds are used to compute the corresponding $F_e(x)$ and $F_P(z)$ as discussed below.

We assume $p(s,o_{x_i}) = e_1 \to e_2 \cdots \to e_l$ and specify each $e \in P(s,o_{x_i})$ a threshold:

$$t_\triangle(e) = t^-(e) + \frac{(t^+(e) - t^-(e)) \cdot (z - \sum_{j=1}^{l} t^-(e_j))}{\sum_{j=1}^{l}(t^+(e_j) - t^-(e_j))} \tag{15}$$

Then, we have the following lemma to compute bounds of $F_p(z)$:

Lemma 4. Given $t_\triangle(e_j)$ (j=1, …, l) in Eq. (15), we have three cases as follows:

- Case 1: If $\sum_{j=1}^{l} t^-(e_j) > z$, we have $F_P(z) = 0$.
- Case 2: If $\sum_{j=1}^{l} t^+(e_j) < z$, we have $F_P(z) = 1$.
- Case 3: If $\sum_{j=1}^{l} t^-(e_j) \le z \le \sum_{j=1}^{l} t^+(e_j)$, we have

$$\prod_{j=1}^{l} F_{e_j}(t_\triangle(e_j)) \le F_P(z) \le 1 - \prod_{j=1}^{l}(1 - F_{e_j}(t_\triangle(e_j))) \tag{16}$$

Proof: According to the definition of $\mathbf{t}(P)$, we have

$$\sum_{j=1}^{l} t^-(e_j) \le \mathbf{t}(P) \le \sum_{j=1}^{l} t^+(e_j),$$

thus, both Cases 1 and 2 hold. As for Case 3, we first prove the lower bound of $F_P(z)$ as follows.

$$\begin{aligned} F_P(z) &= Pr\{\mathbf{t}(s,o_{x_i}) \le z\} \\ &= \sum_{y_1+\cdots+y_l\le z} Pr\{\mathbf{t}(e_1)=y_1 \wedge \cdots \wedge \mathbf{t}(e_l)=y_l\} \\ &\ge \sum_{y_1+\cdots+y_l\le z \wedge y_j \le t_\triangle(e_j)} Pr\{\mathbf{t}(e_1)=y_1 \wedge \cdots \wedge \mathbf{t}(e_l)=y_l\} \end{aligned} \tag{17}$$

According to the definition of $t_\triangle(e_i)$, if $y_i \le t_\triangle(e_i)$ holds for $i=1,\ldots,l$, we have $\sum_{i=1}^{l} x_i \le \sum_{i=1}^{l} t_\triangle(e_i)$ and further:

$$\begin{aligned} \sum_{i=1}^{l} t_\triangle(e_i) &= \sum_{i=1}^{l}(t^-(e_i) + \frac{(t^+(e_i)-t^-(e_i))\cdot(z-\sum_{j=1}^{l} t^-(e_j))}{\sum_{j=1}^{l}(t^+(e_j)-t^-(e_j))}) \\ &= \sum_{i=1}^{l} t^-(e_i) + \frac{(z-\sum_{j=1}^{l} t^-(e_j))\cdot\sum_{i=1}^{l}(t^+(e_i)-t^-(e_i))}{\sum_{j=1}^{l}(t^+(e_j)-t^-(e_j))} \\ &= \sum_{i=1}^{l} t^-(e_i) + z - \sum_{j=1}^{l} t^-(e_j) \\ &= z \end{aligned}$$

Therefore, inequality $\sum_{i=1}^{l} x_i \le z$ always holds when $y_i \le t_\triangle(e_i)(1 \le i \le l)$. Moreover, according to Eq. (17), we have:

$$\begin{aligned} F_P(z) &\ge \sum_{y_1+\cdots+y_l\le z \wedge y_j \le t_\triangle(e_j)} Pr\{\mathbf{t}(e_1)=y_1 \wedge \cdots \wedge \mathbf{t}(e_l)=y_l\} \\ &= Pr\{\mathbf{t}(e_1) \le t_\triangle(e_1) \wedge \cdots \wedge Pr(\mathbf{t}(e_l) \le t_\triangle(e_l)\} \\ &= \prod_{j=1}^{l} Pr\{\mathbf{t}(e_j) \le t_\triangle(e_j)\} = \prod_{j=1}^{l} F_{e_j}(t_\triangle(e_j)) \end{aligned}$$

the left hand side of Eq. (11) holds.

To prove the right hand side of Eq. (11), we need to prove that $1 - F_P(z) \ge \prod_{i=1}^{l}(1 - F_{e_i}(t_\triangle(e_i)))$, i.e.,

$$Pr\{\mathbf{t}(P(s,o_{x_i})) \ge z\} \ge \prod_{i=1}^{l}(1 - F_{e_i}(t_\triangle(e_i))). \tag{18}$$

Similarly, we have:

$$\begin{aligned} Pr\{\mathbf{t}(s,o_{x_i}) \ge z\} &= \sum_{y_1+\cdots+y_l\ge z} Pr\{\mathbf{t}(e_1)=y_1 \wedge \cdots \wedge \mathbf{t}(e_l)=y_l\} \\ &\ge \sum_{y_1+\cdots+y_l\ge z \wedge y_j \ge t_\triangle(e_j)} Pr\{\mathbf{t}(e_1)=y_1 \wedge \cdots \wedge \mathbf{t}(e_l)=y_l\} \\ &= Pr\{\mathbf{t}(e_1) \ge t_\triangle(e_1), \ldots, Pr(\mathbf{t}(e_l) \ge t_\triangle(e_l)\} \\ &= \prod_{i=1}^{l}(1 - F_{e_i}(t_\triangle(e_i))), \end{aligned}$$

and the right hand side of Eq. (11) also holds. □

From Lemma 4, we obtain bounds for $F_P(z)$, i.e., in Cases 1 and 2, we have $F_P^-(z)$=$F_P^+(z)$=0 (or 1); in Case 3, we have $F_P^-(z)$=$\prod_{i=1}^{l} F_{e_i}(t_\triangle(e_i))$ and $F_P^+(z) = 1 - \prod_{i=1}^{l}(1 - F_{e_i}(t_\triangle(e_i)))$. Further, since it holds that $Pr\{a \le \mathbf{t}(P(s,o_{x_i})) \le b\} = F_P(b) - F_P(a)$, we can obtain bounds for $Pr\{a \le \mathbf{t}(P(s,o_{x_i})) \le b\}$ below:

$$Pr\{a \le \mathbf{t}(P(s,o_{x_i})) \le b\} \ge F_P^-(b) - F_P^+(a), \text{ and} \tag{19}$$

$$Pr\{a \le \mathbf{t}(P(s,o_{x_i})) \le b\} \le F_P^+(b) - F_P^-(a). \tag{20}$$

Then, with Eq. (14), $P(s,e)$ can be pruned if there exists a subpath $P(s,o_{x_j})$ (j=1,…,i) such that $F_P^+(b) - F_P^-(a) \le \tau$.

B. Rank-based probability upper bound

The constraint-based upper bound only considers the time constraints of POIs. In this subsection, we will derive a rank-based probability upper bound for filtering out false alarms further.

Assume that we have obtained a set of effective paths (as discussed in Section 4.4.3), U=$\{P_1,\cdots,P_n\}$, sorted by $t^-(\cdot)$ where $Pr_c\{P_i\} \ge \tau(1 \le i \le n)$. We record U_i=$\{P_1,\cdots,P_i\}(i \le n)$. For path P_i, if there are at least h paths in U_{i-1} having travel time less than P_i, then P_i has a rank larger than h. However, a strict ranking among these paths cannot be implemented because the travel time of each path is a random variable bounded by an interval. Our idea is to compute the probability that there are at least h paths in U_{i-1} having travel time less than P_i, i.e.,

$$\sum_{A\subseteq U_{i-1}\wedge|A|\ge h} Pr\{\bigwedge_{P\in A}\mathbf{t}(P) \le \mathbf{t}(P_i)\}\cdot Pr\{\bigwedge_{P\in U_{i-1}\setminus A}\mathbf{t}(P) > \mathbf{t}(P_i)\}$$

Generally, it is not trivial to enumerate all such subsets to compute the exact probability. Here, we reduce to consider the first h paths and have the following lemma.

Lemma 5. Given a set of constraint-based paths U=$\{P_1,\cdots,P_n\}$ sorted by $t^-(.)$, for path $P_i(i > h)$, if $1-\prod_{P\in U_h} LB\_F_P(t^-(P_i))<\tau$, then paths in $U \setminus U_{i-1}$ can be pruned.

Proof: First, we have

$$\begin{aligned} &\sum_{A\subseteq U_{i-1}\wedge|A|\ge h} Pr\{\bigwedge_{P\in A}\mathbf{t}(P) \le \mathbf{t}(P_i)\}\cdot Pr\{\bigwedge_{P\in U_{i-1}\setminus A}\mathbf{t}(P) > \mathbf{t}(P_i)\} \\ &\ge Pr\{\bigwedge_{P\in U_h}\mathbf{t}(P) \le \mathbf{t}(P_i)\} \ge \prod_{P\in U_h} Pr\{\mathbf{t}(P) \le \mathbf{t}(P_i)\} \\ &\ge \prod_{P\in U_h} Pr\{\mathbf{t}(P) \le t^-(P_i)\} \ge \prod_{P\in U_h} F_P^-(t^-(P_i)) \end{aligned}$$

Therefore, we further have $Pr_h\{P_i\} \le 1 - \prod_{P\in U_h} F_P^-(t^-(P_i))$. If $1 - \prod_{P\in U_h} F_P^-(t^-(P_i)) < \tau$, by the inequality transition, we have $Pr_h\{P_i\} < \tau$, thus P_i can be safely pruned. Meanwhile, for each path P_t in $U \setminus U_i = \{P_{i+1},\cdots,P_n\}$, we have

$$Pr_h\{P_t\} \le 1 - \prod_{P\in U_h} F_P^-(t^-(P_t)) \le 1 - \prod_{P\in U_h} F_P^-(t^-(P_i)) < \tau$$

Therefore, $U \setminus U_{i-1}$ can be pruned if $1 - \prod_{P\in U_h} F_P^-(t^-(P_i)) < \tau$. □

Thus, a candidate path that cannot be pruned by the constraint-based probability upper bound might be filtered out by using this rank-based bound. In addition, a sub-path can also be pruned by using this bound as described below.

Lemma 6. Given a set of constrained paths U=$\{P_1,\ldots,P_n\}$ sorted by $t^-(\cdot)$ and a sub-path $P(s,o_{x_i})$, $P(s,o_{x_i})$ can be pruned if $1 - \prod_{P\in U_h} F_P^-(t^-(P(s,o_{x_i}))) < \tau$.

Proof: A complete path generated from $P(s,o_{x_i})$ has a larger $t^-(\cdot)$ value than that of $P(s,o_{x_i})$, which reduces its corresponding *PTP* probability, hence the proof. □

Thus, $1 - \prod_{P\in U_h} F_P^-(t^-(P))$ can be used as the rank-based probability upper bound to prune candidate paths.

C. Discussion on two probability upper bounds

During the query processing, both constraint-based and rank-based upper bounds of the *PTP* probability are employed to prune paths. The constraint-based probability upper bound can prune those paths that have constraint-based probabilities less than τ, which mainly considers the keyword/time constraints of POIs. For the rank-based probability upper bound, it requires that there exist h candidate paths whose constraint-based probabilities are above τ. Thus, we can also use it to prune those candidate paths that have *PTP* probabilities less than τ. Therefore, the rank-based probability works as a complement to the constraint-based one to collaboratively reduce the number of candidate paths.

4.4.3 Travel Time Pruning

In this subsection, we further describe the *travel time pruning*, denoted as **T^2-Pruning**. According to the definition of the *PTP* query, we aim to compute the probabilities that paths are among the top-h time-constrained paths which satisfy both keyword and time constraints and have the top-h smallest travel times. Thus, in **T^2-Pruning**, our basic idea is to use a travel time threshold to directly prune those paths with larger travel times than h constraint-based paths we have seen so far without computing probability upper bounds. Concretely, we have the **T^2-Pruning** in the following lemma.

Lemma 7. (Travel Time Pruning, **T^2-Pruning**) Assume that we have obtained h effective paths, which satisfy time constraints at POIs on these paths with probability above τ. Let ρ be the largest travel time upper bound among these h paths. Then, a path $P(s,e)$ can be pruned if its lower bound of travel time, $t^-(P(s,e))$, is larger than ρ.

Proof: From the lemma assumption, if $t^-(P(s,e)) > \rho$ holds, $P(s,e)$ has larger travel time than at least h paths in all possible worlds (since we have obtained h paths that satisfy keyword/time constraints at POIs). Hence, path $P(s,e)$ cannot be a top-h time-constrained path and should be pruned. □

We consider two cases of obtaining h effective paths used in Lemma 7. First, for $P(s,e)$, if the arrival time at every POI on $P(s,e)$ is within the corresponding service time, i.e., $[a_c^-(o_{x_i}), a_c^+(o_{x_i})]$=$[a^-(o_{x_i}), a^+(o_{x_i})]$, then path $P(s,e)$ is an effective path. The second case is that, there exist some POIs o_{x_i} on $P(s,\ e)$ whose time intervals do not fully cover the whole arrival time, i.e., $[a_c^-(o_{x_i}), a_c^+(o_{x_i})] \neq [a^-(o_{x_i}), a^+(o_{x_i})]$. In this case, our rationale is to derive an lower bound, $Pr_c^-\{P(s,e)\}$, for $Pr_c\{P(s,e)\}$. If this lower bound is above threshold τ, then path $P(s,e)$ is an effective path. According to the definition of $Pr_c\{P(s,e)\}$, we have

$$\begin{aligned} Pr_c\{P(s,e) &\geq Pr_c\{P(s,o_{x_1})\} \cdot \prod_{i=2}^{\psi} Pr_c\{P(o_{x_{i-1}}, o_{x_i})\} \\ &\geq Pr_c^-\{P(s,o_{x_1})\} \cdot \prod_{i=2}^{\psi} Pr_c^-\{P(o_{x_{i-1}}, o_{x_t})\} \\ &\equiv Pr_c^-\{P(s,e)\} \end{aligned} \quad (21)$$

where $Pr_c^-\{P(x,y)\}$ is a lower bound for $Pr_c\{P(x,y)\}$.

Now our goal is to compute the lower bounds of $Pr_c\{P(s, o_{x_1})\}$ and $Pr_c\{P(o_{x_{i-1}}, o_{x_i})\}$ (for $2 \leq i \leq \psi$). Obviously, if there exists a travel time interval for $\mathbf{t}(P(x,y))$ in which $P(x,y)$ always satisfies the keyword/time constraints of POIs, we can compute a lower bound for the probability that the travel time $\mathbf{t}(x,y)$ falls into this interval by using Eq. (19). Based on the constrained arrival time at $o_{x_{i-1}}$ and o_{x_i}, the constraint-based interval of $\mathbf{t}(P(o_{x_{i-1}}, o_{x_i}))$, i.e., $[t_c^-(P(o_{x_{i-1}}, o_{x_i})),\ t_c^-(P(o_{x_{i-1}}, o_{x_i}))]$, can be computed by solving the following inequalities:

$$\begin{cases} t^-(P(o_{x_{i-1}}, o_{x_i})) \leq x \leq t^+(P(o_{i-1}, o_{x_i})), \\ a_c^-(o_{x_i}) \leq a_c^-(o_{x_{i-1}}) + x + q.t_{i-1} \leq a_c^+(o_{x_i}) \\ a_c^-(o_{x_i}) \leq a_c^+(o_{x_{i-1}}) + x + q.t_{i-1} \leq a_c^+(o_{x_i}). \end{cases} \quad (22)$$

Then, by using Eq. (19), we can calculate a lower bound of $Pr_c\{P(o_{x_{i-1}}, o_{o_{x_i}})\}$ below:

$$\begin{aligned} & Pr_c\{P(o_{i-1}, o_{x_i})\} \\ \geq\ & Pr\{t_c^-(P(o_{x_{i-1}}, o_{x_i})) \leq \mathbf{t}(P(o_{x_i}, o_{x_{i+1}})) \leq t_c^+(P(o_{i-1}, o_i))\} \\ \geq\ & F^-(t_c^+(P(o_{i-1}, o_{x_i}))) - F^+(t_c^-(P(o_{x_{i-1}}, o_{x_i}))). \\ \equiv\ & Pr_c^-\{P(o_{i-1}, o_{x_i})\}. \end{aligned} \quad (23)$$

With $Pr_c^-\{P(s, o_{x_1})\}$ and $Pr_c^-\{P(o_{x_{i-1}}, o_{x_i})\}$, we can compute the lower bound of the constraint-based probability in Eq. (21). If this constraint-based probability is above τ, then we can use $P(s,e)$ as an effective path to filter out other false alarms as described in Lemma 7.

4.5 PTP Processing Algorithm

To efficiently answer *PTP* queries, we first generate a small set of candidate paths by using our pruning strategies. After that, a refinement step based on sampling is conducted to obtain the final *PTP* query results.

4.5.1 Candidate path generation

We propose a expansion-based heuristic algorithm which has the flavor similar to A* algorithm. We expand from POI to POI and the shortest path between each pair of POIs is quickly computed with CH index. During the expansion, **T-pruning**, **P-pruning** and **T^2-pruning** are employed together to filter out infeasible paths.

Algorithm 1: GenerateCandidatePath

Input: a *PTP* query q=$(s, e, t_d, \mathcal{K}, h, \tau)$
Output: Candidate paths set C

```
1  Initialize C, Q, U ←new priorityQueue()
2  ρ ← +∞
3  P ← ⟨s⟩
4  Q.enqueue(P,t_est(P))
5  while Q is not empty do
6      P ← Q.dequeue()
7      if t_est(P) ≥ ρ then
8          break
9      i ←POICount(P)
10     o_curr ←currPOI(P)
11     if i = ψ then
12         P_new ← P ⊕ P(o_curr, e)
13         if t⁻(P_new) < ρ then
14             C.add(P_new, t⁻(P_new))
15         if Pr_c{P_new} ≥ τ then
16             U.add(P_new, t⁻(P_new))
17             P-Pruning(C)
18             update(ρ)
19     else
20         P_new ← P.expand()
21         if not(t⁻(P_new) ≥ ρ and T-Pruning(P_new) and
           P-Pruning(P_new)) then
22             Q.enqueue(P_new, t_est(P_new));
23         update P
24         Q.enqueue(P,t_est(P))
25 return C
```

The *PTP* query processing is elaborated in Algorithm 1 which accepts as input a *PTP* query q and produces as output a small set of candidate paths C. Initially, three priority queues C, Q, and U are created to store candidate paths, expanded paths which have covered partial POIs, and constraint-based paths, respectively. All these paths are sorted by the lower bound, $t^-(\cdot)$, of the travel time. In addition, ρ records the threshold of the current top-h constraint-based paths (line 2).

First, the path containing only the start vertex, s, is added to Q (lines 3-4) and its weight is an estimated value of the travel time, $t_{est}(P)$. Assume that the current POI of P is o_{curr} (initially, s)

and the next POI (if exists) is o_{next}. Meanwhile, $Euc(x, y)$ is the Euclidean distance between x and y, and $vel^+(\mathcal{G})$ is the maximum velocity over the whole road network. Then, we compute $t_{est}(P)$ as follows:

- Case 1: P has covered all required POIs.

$$t_{est}(P) = t^-(P) + \frac{Euc(o_{curr}, e)}{vel^+(\mathcal{G})}$$

- Case 2: P only covers partial POIs.

$$t_{est}(P) = t^-(P) + \frac{Euc(o_{curr}, o_{next}) + Euc(o_{next}, e)}{vel^+(\mathcal{G})}$$

In each loop, while Q is not empty (line 5), the path P with the minimum $t_{est}(P)$ is dequeued from Q (line 6). If $t_{est}(P) \geq \rho$ (line 7), algorithm terminates, otherwise we have two cases as described below:

- **Case 1**: P has covered all required POIs (line 11). In this case, we expand from P to compute the shortest path to e and generate a complete path, P_{new}, from s to e. If P_{new} survives in **T^2-pruning**, it is returned as a candidate path (line 14). Additionally, if $Pr_c\{P_{new}\} \geq \tau$, P_{new} is added to U (line 16), each path in C is checked by P-pruning (line 17), and ρ is updated (line 18).
- **Case 2**: P only covers partial POIs (line 19). In this case, we expand from P to cover the $(i+1)$-th POI, where i is the number of POIs that P has already covered, and a new path, P_{new}, is generated. Then, three pruning methods are employed. If P_{new} survives, we add it to Q with its estimate travel time. Meanwhile, P is updated (next POI, o_{next}) and reinserted into Q.

The loops terminate when the dequeued path has a $t_{est}(\cdot)$ value larger than ρ, or Q is empty. Finally, a set C of candidates is returned.

4.5.2 Refinement

Generally, the number of candidate paths is small but some of them may be not PTP answers. Hence, a refinement step is required to compute the real results. In order to efficiently obtain the final result, we use a sampling algorithm, as described in Algorithm 2, to calculate the probabilities of candidate paths that belong to top-h time-constrained paths. Particularly, in line 4, function *sample*($\mathcal{G}$) samples the travel time of each edge, according to its probability distribution. Here, only the edges covered by the paths in U are sampled, since other edges of $\mathcal{G}$ have no effect on the final result. According to the Monte Carlo theory [26], we have:

$$Pr\{|Pr_h(P) - \hat{Pr}(P)| \leq \eta\} > 1 - \xi \quad (24)$$

where $\hat{Pr}(P)$=N_0/N is computed according to Algorithm 2, and both parameters ξ and η are set to 0.1 by default [26] (line 2). Finally, those paths with N_0/N larger than τ are returned as the final query result.

Algorithm 2: Calculate $\hat{Pr}(P)$

Input: $P \in C$
Output: $\hat{Pr}(P)$

1 $N_0 \leftarrow 0$
2 $N \leftarrow \frac{4(|C|-1)\ln(2/\xi)}{\eta}$
3 **for** $i \leftarrow 1$ *to* N **do**
4 $\quad w_i \leftarrow$ sample($\mathcal{G}$)
5 $\quad$ compute $\mathcal{P}^h_{w_i}(s, e)$ among U
6 $\quad$ **if** $P \in \mathcal{P}^h_{w_i}(s, e)$ **then**
7 $\quad\quad N_0 \leftarrow N_0$+1
8 **return** N_0/N

4.6 Updates of URNs

In order to capture the dynamic uncertain travel times on roads of an uncertain road network $\mathcal{G}$, we maintain a queue for each edge $e_{i,j}$, which contains T samples, s_r $(t_0 - T + 1 \leq r \leq t_0)$, for the last T timestamps within $[t_0 - T + 1, t_0]$, where t_0 is the current timestamp. Each sample is represented by a triple $(e_{i,j}, s_r, r)$, where $e_{i,j}$ is an edge, s_r is a possible travel time value on edge $e_{i,j}$, and r is the timestamp that the sample is collected.

In order to facilitate *PTP* query processing, we need to maintain some pre-computed data for the *URN* $\mathcal{G}$, including the minimum/maximum travel time, $t^-(e_{i,j})$ and $t^+(e_{i,j})$, on each edge $e_{i,j}$; and the minimum/maximum velocities, denoted as $vel^-(\mathcal{G})$ and $vel^+(\mathcal{G})$, on the entire road network $\mathcal{G}$. At a new timestamp $(t_0 + 1)$, a new sample, $s_{(t_0+1)}$, of the travel time on edge $e_{i,j}$ arrives and is added to the queue. Meanwhile, the old one, $s_{(t_0-T+1)}$, is expired and removed from the queue, and the pre-computed data is updated as described below.

- **Update $t^-(e_{i,j})$ and $t^+(e_{i,j})$**: For new sample $s_{(t_0+1)}$, if $s_{(t_0+1)}$ is within interval $[t^-(e_{i,j}), t^+(e_{i,j})]$, we do not need to update the time interval. Otherwise, if $s_{(t_0+1)}$<$t^-(e_{i,j})$ holds, we update $t^-(e_{i,j})$ with the new sample $s_{(t_0+1)}$; if $s_{(t_0+1)} > t^+(e_{i,j})$ holds, we update $t^+(e_{i,j})$ with sample $s_{(t_0+1)}$. For the expired sample $s_{(t_0-T+1)}$, if $s_{(t_0-T+1)} \in [t^-(e_{i,j}), t^+(e_{i,j})]$, we do not need to update the time interval. When the expired sample $s_{(t_0-T+1)}$ equals to either $t^-(e_{i,j})$ or $t^+(e_{i,j})$, we need to scan all samples within the last T timestamps, and re-calculate the interval $[t^-(e_{i,j}), t^+(e_{i,j})]$.
- **Update $vel^-(\mathcal{G})$ and $vel^+(\mathcal{G})$**: If the travel time interval $[t^-(e_{i,j}), t^+(e_{i,j})]$ above remains the same, we do not need to update $vel^-(\mathcal{G})$ and $vel^+(\mathcal{G})$. Otherwise, we calculate the minimum and maximum velocities on edge $e_{i,j}$, i.e., $vel^-(e_{i,j})$ and $vel^+(e_{i,j})$, respectively. That is, if $t^+(e_{i,j})$ has changed, we have $vel^-_{new}(e_{i,j})$=$|(e_{i,j}|/t^+(e_{i,j})$; if $t^-(e_{i,j})$ is updated, we have $vel^+_{new}(e_{i,j})$=$|(e_{i,j}|/t^-(e_{i,j})$. As a result, if old velocity bounds $[vel^-(e_{i,j}), vel^+(e_{i,j})]$ is within $[vel^-(\mathcal{G}), vel^+(\mathcal{G})]$, we update $vel^+(\mathcal{G})$ with $vel^+(e_{i,j})$ (when $vel^+(e_{i,j})$>$vel^+(\mathcal{G})$), and update $vel^-(\mathcal{G})$ with $vel^-(e_{i,j})$ (when $vel^-(e_{i,j})$<$vel^-(\mathcal{G})$). On the other hand, if $vel^-(e_{i,j})$=$vel^-(\mathcal{G})$ and $vel^-(e_{i,j})$ increases to $vel^-_{new}(e_{i,j})$, then we re-compute the velocity lower bound for all edges. Similarly, if $vel^+(e_{i,j}) = vel^+(\mathcal{G})$ and $vel^+(e_{i,j})$ decreases to $vel^+_{new}(e_{i,j})$, we also re-calculate the velocity upper bound for all edges.

5 Experimental Evaluation

5.1 Data sets & Setups

In this section, we conduct extensive experiments on three real data sets (i.e., California, Beijing and London) to evaluate the performance of our proposed solution to the *PTP* problem.

Specifically, California data set (including both road network and POIs) is from *Real Datasets for Spatial Databases*[2]; Beijing data set has road network from *OpenStreetMap*[3] (OSM) and POIs from *Datatang*[4]; London data set extracts both road network and its POIs from OSM. The statistical information of three data sets, including index construction times and index sizes for both inverted file (IF) and Contraction Hierarchy (CH), is detailed in Table 4.

TABLE 4
Statistics of data sets.

Attributes	California	Beijing	London
number of vertices	21,048	46,029	209,045
number of edges	21,693	62,778	282,267
number of POIs	104,470	252,125	34,341
Index size (IF) (MB)	0.5	1.2	0.3
Index construction time (IF) (sec)	0.11	0.595	0.043
Index size (CH) (MB)	44	152	341
Index construction time (CH) (sec)	2.2	28.7	47.9

TABLE 5
Parameters in experiments.

Parameters	Values
$\|\mathcal{K}\|$	1,**2**,3,4,5
h	1, **3**, 5, 7, 9
τ	0.1, 0.3, **0.5**, 0.7, 0.9

Keywords of POIs: The POIs of California has 62 categories and that of Beijing has 533 categories. In our experiment, each category is regarded as the keyword description of POIs, i.e., $o_i.K$, in this category. Each POI in London is associated with a set of keywords, extracted from OSM, to describe its properties and services.

Time intervals of POIs: We define three time interval patterns, i.e., $\{$7:00-12:00,13:00-17:00$\}$ (such as banks), $\{$8:00-22:00$\}$ (such as restaurants), and $\{$0:00-24:00$\}$ (such as 24-hour stores). First, one of the three patterns is selected for each POI. After that, to increase the diversity of the time intervals of POIs, for each POI, we generate sub-intervals of the selected time interval pattern such that they have an overlap of more than 70%. For example, assuming that a time interval in the pattern is $[a, b]$, we randomly generate a sub-interval $[x, y]$ within $[a, b]$ such that $\frac{y-x}{b-a} \geq 0.7$.

Travel time of roads: First, for each road, we randomly select a speed sub-interval within [20km/h, 80km/h] which is the driving speed interval for most cities. Then, a set, $\mathcal{V}$, of T(=50) speed samples is randomly generated within the sub-interval according to ***Uniform distribution*** and ***Truncated Gaussian distribution***, respectively. In addition, we also extract ***Real*** speed samples from the GPS trajectories of over 10, 000 taxis in Beijing data set[5]. Then, with the length of each road e, each sample of $\mathbf{t}(e)$ is obtained by $\frac{|e|}{v}$ ($v \in V$).

Queries: For each data set, we evaluate 50 *PTP* queries whose start and end locations are generated randomly within the road network and their departure times are randomly selected between 6:00 and 22:00. Additionally, the query keywords are also randomly selected from the corresponding vocabulary of each data set. The staying time at each POI is randomly selected between 600 and 1,200 seconds.

Table 5 illustrates experimental settings, where default values of parameters are in bold font. All experiments are run on a PC with a 3.1GHz Intel processor and a 8GB RAM.

2. http://www.cs.utah.edu/ lifeifei/SpatialDataset.htm
3. http://www.openstreetmap.org
4. http://www.datatang.com/
5. In the case that there are no enough samples for some roads, we add the average values of existing samples to ensure T samples.

5.2 Experimental Results

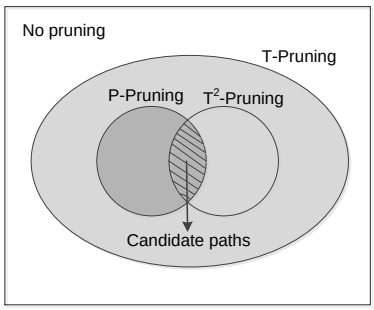


Fig. 4. The relationship between three pruning methods

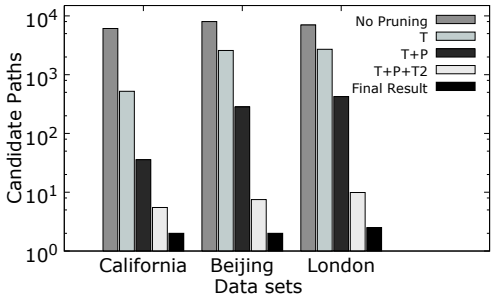


Fig. 5. Pruning ability test on different data sets

Method comparison: We compare the performance of our solution with that of the solution by enumerating all possible worlds [24] (EPW for short). As illustrated in Figure 6, the response time of EPW is much larger than that of our solution on all three datasets. Actually, when the number of query keywords increases, it is computationally prohibitive to process the PTP query with EPW solution. On the contrary, our solution can always compute the PTP query efficiently.

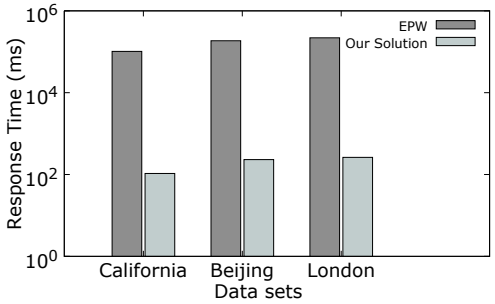
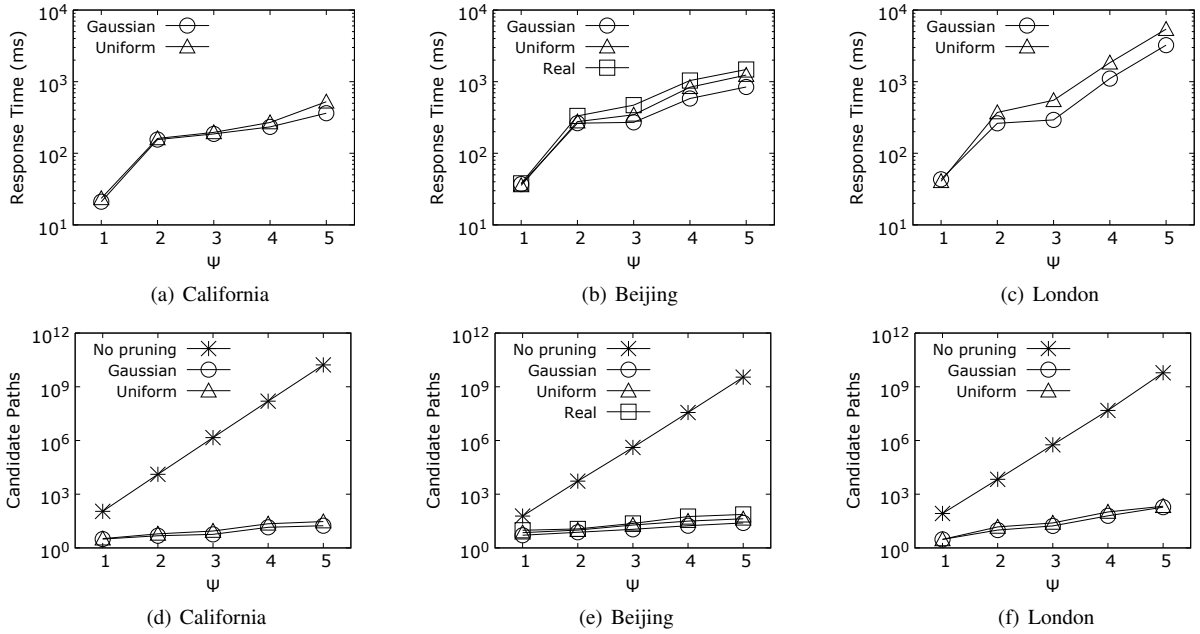


Fig. 6. The response time comparison between EPW and our solution

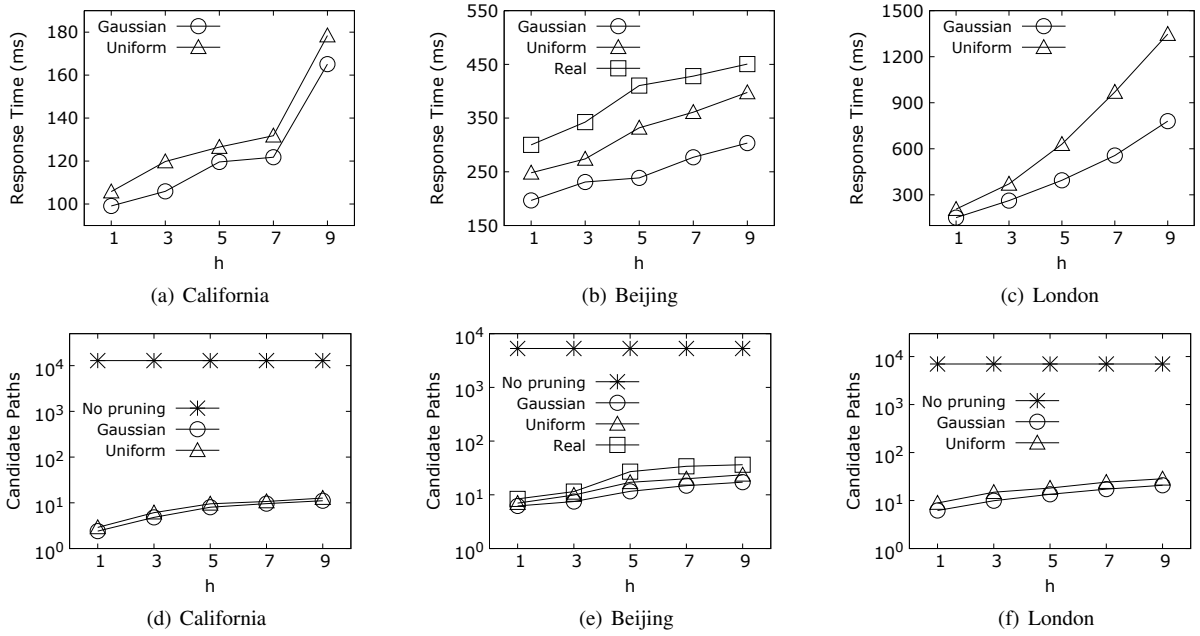
Pruning power vs. data sets: Figure 4 illustrates the relationship of candidate path sets after applying pruning strategies, i.e., **T-pruning** (T), **P-pruning** (P), and **T^2-pruning** (T^2). We can see that T-pruning results in a superset of candidate sets from the other two pruning strategies. Moreover, the intersection of two candidate path sets by applying P-pruning and T^2-pruning is our final candidate set that needs further refinement. Figure 5 illustrates the performance of three pruning strategies over three data sets with Gaussian vehicle speeds (the case of data sets with Uniform vehicle speeds is similar and thus omitted). As presented, a considerable portion of infeasible paths are pruned when only T-pruning is employed. Further, compared with T-pruning, the number of candidate paths after T+P pruning is greatly reduced. Finally, we can get a small set of candidate paths with all the three pruning strategies, i.e., T+P+T^2 pruning. Furthermore, we can see



(a) California (b) Beijing (c) London

(d) California (e) Beijing (f) London

Fig. 7. The performance vs. the number of keyword sets, $|q.\mathcal{K}|$



(a) California (b) Beijing (c) London

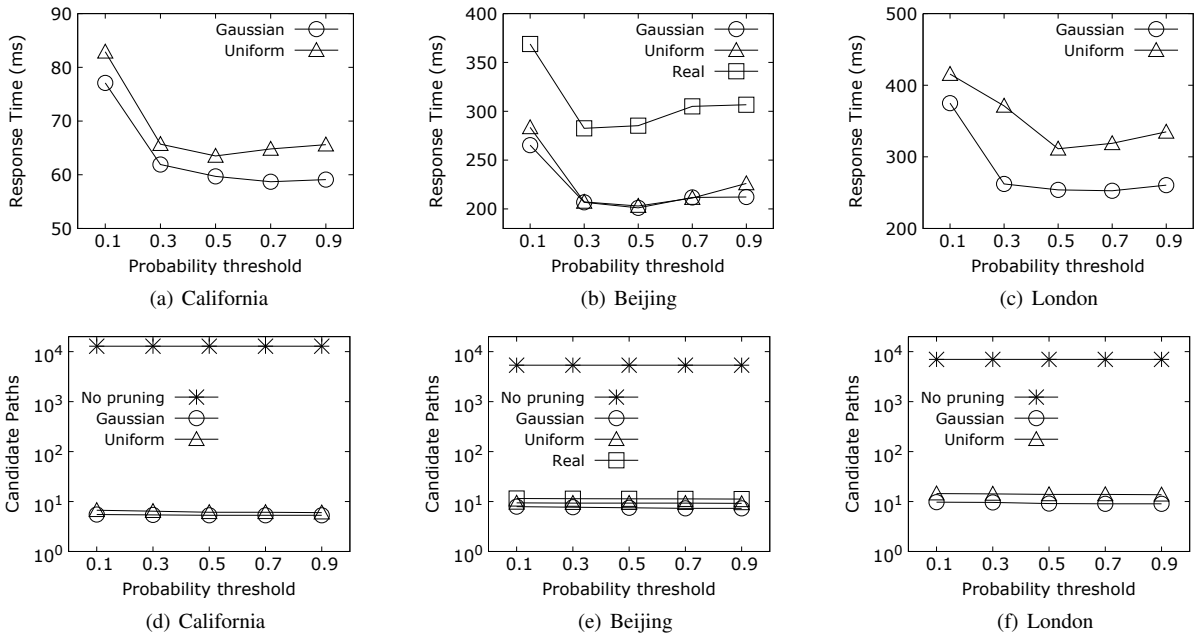
(d) California (e) Beijing (f) London

Fig. 8. The performance vs. parameter, h

from Figure 5 that the number of final candidate paths (after all pruning) on each data set has almost the same scale as that of final results (obtained after the verification step given in Algorithm 2), which confirms the effectiveness of our pruning strategies.

Performance vs. the number of keyword sets $|\mathcal{K}|$: Figure 7 illustrates the experimental results by varying the number of query keyword sets, i.e., $|\mathcal{K}|$, on different data sets with Gaussian, Uniform, and Real distribution of the travel time on each edge. With the increase of $|\mathcal{K}|$, the response time increases as more paths should be evaluated. Nonetheless, with $|\mathcal{K}|$ less than 4 (the general case in reality), we can efficiently get final results with around one second. With the increase of $|\mathcal{K}|$, the number of complete paths increases exponentially without applying any pruning methods. By using our proposed pruning strategies, however, the number of candidate paths slightly increases, and remains small (i.e., less than 20). For example, the number of complete paths without pruning is in the scale of millions when $|\mathcal{K}|$=3, and the number of candidate paths after T+P+T^2 pruning is only in the scale of tens. Meanwhile, with Uniform distribution of $\mathbf{t}(e)$, there are more candidate paths than that of truncated Gaussian distribution, which accordingly takes higher response time. This is because, Gaussian distribution concentrates around the mean value, which makes the P-pruning more effective. The real distribution takes even more time and generates more candidate paths because its speed span is larger than that of other two distribution.

Performance vs. parameter h**:** Figure 8 presents the response time and the number of candidate paths by varying h, where other parameters are set to their default values. We can see that the number of candidate paths has a smooth increase, since larger $q.h$ means more paths need to be retrieved and checked. This further incurs an increase in the query response time for large h, as shown in Figure 8.



(a) California (b) Beijing (c) London

(d) California (e) Beijing (f) London

Fig. 9. The performance vs. probability threshold, τ

Performance vs. probability threshold τ: Figure 9 reports the experimental results with different probability thresholds from 0.1 to 0.9. From Figure 9, the curves for response time are almost concave. The reason is as follows. On one hand, a small τ covers more candidate paths, but it generates a good pruning threshold ρ in Algorithm 1. On the other hand, a larger τ can prune more expanding paths during the query processing, but impede the generation of the pruning threshold ρ. Therefore, with a small τ (e.g., 0.1), large number of evaluated paths incurs a high response time. With the increase of τ, the number of evaluated paths decreases, which results in the decrease of the response time. However, when τ becomes higher, Algorithm 1 will postpone to terminate because the pruning threshold ρ is difficult to generate (since fewer paths satisfy the constraint of the probability threshold τ). Thus, the response time first decreases, and then increases again for large τ. With the increase of τ, the number of candidate paths slightly decreases because fewer complete paths satisfy the probabilistic requirement.

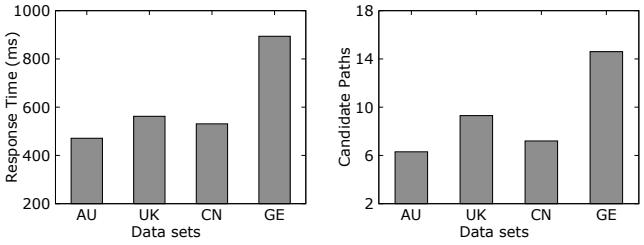
Scalability: To evaluate the scalability of the proposed solution, we test the performance of our approach against the size of road networks. As presented in Table 6, we use four datasets, i.e., the road network and POIs of Australia, United Kingdom, China, and Germany[6]. We run 100 queries with default parameter in Table 5 and compute the average response time and candidate paths. As illustrated in Figure 10, we can conclude that our proposed solution also achieve a good performance over larger road networks with millions of roads and POIs.

TABLE 6
Data sets for scalability test.

Attributes	Australia	United Kingdom	China	Germany
number of vertices	202, 666	389, 503	914, 297	1,029, 674
number of edges	272, 558	487, 160	1, 240, 893	1, 369, 473
number of POIs	130, 644	539, 912	198, 219	1, 333, 775

In addition, we also validate the good performance of our solution while varying the staying time, departure time, etc. For the limitation of space, we omit the detailed results here.

6. All these datasets are extracted from OSM and we make them public at http://pan.baidu.com/s/1boVgJkr



(a) Response time (b) #Candidate paths

Fig. 10. The scalability test over large road networks

6 Conclusion

In this paper, we formalize a probabilistic time-constrained *PTP* query, which models road networks by uncertain graphs, and retrieves paths that satisfy both keyword and time constraints at POIs, and are in the set of top-h optimal paths with high probabilities. In order to efficiently tackle the *PTP* problem, we propose three effective pruning strategies to filter out false alarms, and integrate them into an efficient *PTP* search algorithm. Through extensive experiments, we demonstrate the efficiency and effectiveness of our proposed *PTP* query answering approach.

Acknowledgments

Wengen Li and Jihong Guan were supported by the National Natural Science Foundation of China under grant No. 61373036 and the Program of Shanghai Subject Chief Scientist under grant No. 15XD1503600. Shuigeng Zhou was supported by the Key Projects of Fundamental Research Program of Shanghai Municipal Commission of Science and Technology under grant No. 14JC1400300. Wengen Li and Jiannong Cao were supported by the Project of Strategic Importance of The Hong Kong Polytechnic University under grant No. 1-ZE26, the NSFC/RGC Joint Research Scheme under grant No. N_PolyU519/12, and the NSFC key project under grant No. 61332004.

References

[1] J. Zhang, C. Chow, and Y. Li, "igeorec: A personalized and efficient geographical location recommendation framework," *IEEE T. Services Computing*, vol. 8, no. 5, pp. 701–714, 2015.

[2] X. Cao, L. Chen, G. Cong, and X. Xiao, "Keyword-aware optimal route search," *PVLDB*, vol. 5, no. 11, pp. 1136–1147, 2012.

[3] M. Sharifzadeh, M. R. Kolahdouzan, and C. Shahabi, "The optimal sequenced route query," *VLDB J.*, vol. 17, no. 4, pp. 765–787, 2008.

[4] B. Yao, M. Tang, and F. Li, "Multi-approximate-keyword routing in gis data," in *GIS*, 2011, pp. 201–210.

[5] R. Levin, Y. Kanza, E. Safra, and Y. Sagiv, "Interactive route search in the presence of order constraints," *PVLDB*, vol. 3, no. 1, pp. 117–128, 2010.

[6] R. W. Hall, "The fastest path through a network with random time-dependent travel times," *Transportation Science*, vol. 20, no. 3, pp. 182–188, 1986.

[7] E. Miller-hooks, "Adaptive least-expected time paths in stochastic, time-varying transportation and data networks," *Networks*, vol. 37, no. 1, pp. 35–52, 2001.

[8] M. Hua and J. Pei, "Probabilistic path queries in road networks: traffic uncertainty aware path selection," in *EDBT*, 2010, pp. 347–358.

[9] B. Y. Chen, W. H. K. Lam, Q. Li, A. Sumalee, and K. Yan, "Shortest path finding problem in stochastic time-dependent road networks with stochastic first-in-first-out property," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1907–1917, 2013.

[10] L. Yang and X. Zhou, "Constraint reformulation and a lagrangian relaxation-based solution algorithm for a least expected time path problem," *Transportation Research Part B: Methodological*, vol. 59, no. C, pp. 22–44, 2014.

[11] C. F. Costa, M. A. Nascimento, J. A. F. de Macêdo, and J. C. Machado, "Nearest neighbor queries with service time constraints in time-dependent road networks," in *Proceedings of the Second ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS 2013, November 5, 2013, Orlando, Florida, USA*, 2013, pp. 22–29.

[12] A. Chen and Z. Ji, "Path finding under uncertainty," *Journal of Advanced Transportation*, vol. 39, no. 1, pp. 19–37, 2005.

[13] Y. Nie and X. Wu, "Shortest path problem considering On-Time arrival probability," *Transportation Research Part B*, vol. 43, no. 6, pp. 597–613, 2009.

[14] H. Huang and S. Gao, "Optimal paths in dynamic networks with dependent random link travel times," *Transportation Research Part B: Methodological*, vol. 46, no. 5, pp. 579–598, 2012.

[15] T. Xing and X. Zhou, "Finding the most reliable path with and without link travel time correlation: A lagrangian substitution based approach," *Transportation Research Part B: Methodological*, vol. 45, no. 10, pp. 1660–1679, 2011.

[16] X. Zhou and T. Xing, "Reformulation and solution algorithms for absolute and percentile robust shortest path problems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 943–954, 2013.

[17] Y. Nie and X. Wu, "Shortest path problem considering On-Time arrival probability," *Transportation Research Part B*, vol. 43, no. 6, pp. 597–613, 2009.

[18] Y. M. Nie and X. Wu, "Reliable a priori shortest path problem with limited spatial and temporal dependencies," in *Proceedings of the 18 th International Symposium on Transportation and Traffic Theory*, 2009, pp. 69–195.

[19] S. Samaranayake, S. Blandin, and A. Bayen, "A tractable class of algorithms for reliable routing in stochastic networks," in *Proceedings of the 19th International Symposium on Transportation and Traffic Theory*, 2011.

[20] M. G. Bell, "Hyperstar: A multi-path astar algorithm for risk averse vehicle navigation," *Transportation Research Part B: Methodological*, vol. 43, no. 1, pp. 97 – 107, 2009.

[21] J.-D. Schmöcker, H. Shimamoto, and F. Kurauchi, "Generation and calibration of transit hyperpaths," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 406 – 418, 2013.

[22] L. Sun, W. Gu, and H. Mahmassani, "Estimation of expected travel time using the method of moment," *Canadian Journal of Civil Engineering*, vol. 38, no. 2, pp. 154–165, 2011.

[23] X. Lian and L. Chen, "Trip planner over probabilistic time-dependent road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 2058–2071, 2014.

[24] S. Abiteboul, P. C. Kanellakis, and G. Grahne, "On the representation and querying of sets of possible worlds," in *SIGMOD Conference*, 1987, pp. 34–48.

[25] R. Gandhi, S. Khuller, and A. Srinivasan, "Approximation algorithms for partial covering problems," *J. Algorithms*, vol. 53, no. 1, pp. 55–84, 2004.

[26] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge Univ. Press, 2005.

[27] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Comput. Surv.*, vol. 38, no. 2, Jul. 2006.

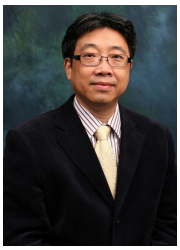
[28] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *WEA*, 2008, pp. 319–333.

Wengen Li received his BS degree from the Department of Computer Science and Technology, Tongji University, in 2011. He is currently a joint PhD candidate at Department of Computer Science and Technology, Tongji University and the Department of Computing, Hong Kong Polytechnic University. His research interests include spatial databases and information retrieval.

Jihong Guan received the bachelor's degree from Huazhong Normal University in 1991, the master's degree from Wuhan Technical University of Surveying and Mapping (merged into Wuhan University since 2000) in 1991, and the PhD degree from Wuhan University in 2002. She is currently a professor in the Department of Computer Science and Technology, Tongji University, Shanghai, China. Before joining Tongji University, she served in the Department of Computer, Wuhan Technical University of Surveying and Mapping from 1991 to 1997, as an assistant professor and an associate professor (since August 2000), respectively. She was an associate professor (2000-2003) and a professor (Since 2003) in the School of Computer, Wuhan University. Her research interests include databases, data mining, distributed computing, bioinformatics, and geographic information systems (GIS).

Xiang Lian received the BS degree from the Department of Computer Science and Technology, Nanjing University, in 2003, and the PhD degree in computer science from the Hong Kong University of Science and Technology, Hong Kong. He is now an assistant professor in the Department of Computer Science at the University of Texas Rio Grande Valley. His research interests include probabilistic/uncertain data management, probabilistic RDF graphs, inconsistent probabilistic databases, and streaming time series.

Shuigeng Zhou received the bachelor's degree from Huazhong University of Science and Technology (HUST) in 1988, the master's degree from the University of Electronic Science and Technology of China (UESTC) in 1991, and the PhD degree in computer science from Fudan University, Shanghai, China, in 2000. He is currently a professor in the School of Computer Science, Fudan University. He served in Shanghai Academy of Spaceflight Technology from 1991 to 1997, as an engineer and a senior engineer (since 1995), respectively. He was a postdoctoral researcher in the State Key Lab of Software Engineering, Wuhan University from 2000 to 2002. His research interests include data management, data mining and bioinformatics.



Jiannong Cao received the BSc degree from Nanjing University, China, in 1982, and the MSc and PhD degrees from Washington State University, USA, in 1986 and 1990, all in computer science. He is currently a chair professor and the head of the Department of Computing at Hong Kong Polytechnic University. His research interests include parallel and distributed computing, computer networks, mobile and pervasive computing, fault tolerance, and middle-ware.