Maximizing the Probability of Arriving on Time: A Practical Q-Learning Method

Zhiguang Cao[1], **Hongliang Guo**[2], **Jie Zhang**[3], **Frans Oliehoek**[4,5], **and Ulrich Fastenrath**[6]
[1]Energy Research Institute @NTU, Nanyang Technological University, Singapore
[2]School of Automation Engineering, University of Electronic Science and Technology of China, China
[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[4]Department of Computer Science, University of Liverpool, UK
[5]Informatics Institute, University of Amsterdam, Netherlands
[6]Department of Traffic Information Management and Routing Optimization, BMW Group, Germany
caoz0005@ntu.edu.sg, guohl1983@uestc.edu.cn, zhangj@ntu.edu.sg, frans.oliehoek@liverpool.ac.uk, Ulrich.Fastenrath@bmw.de

Abstract

The stochastic shortest path problem is of crucial importance for the development of sustainable transportation systems. Existing methods based on the probability tail model seek for the path that maximizes the probability of arriving at the destination before a deadline. However, they suffer from low accuracy and/or high computational cost. We design a novel Q-learning method where the converged Q-values have the practical meaning as the actual probabilities of arriving on time so as to improve accuracy. By further adopting dynamic neural networks to learn the value function, our method can scale well to large road networks with arbitrary deadlines. Experimental results on real road networks demonstrate the significant advantages of our method over other counterparts.

Introduction

The sustainability of future intelligent transport systems (ITSs) critically depends on two aspects. First, ITSs need to be efficient, in the sense that they enable as many users as possible to minimize their expected travel time (Wu, Sheldon, and Zilberstein 2016; Cao et al. 2016d). Second, ITSs need to enable the users to reliably predict their travel time in order to meet their appointments and deadlines (Cao et al. 2016a). Due to its ability to model uncertainties in travel time, the stochastic shortest path (SSP) problem has emerged as the dominant theoretical framework for dealing with these aspects (Lim et al. 2013; Anantharam et al. 2016). The problem is well-understood and efficiently solved in risk neutral settings, which corresponds to the first aspect: a path is considered as optimal if it guarantees the *least expected travel time* (LET) (Miller-Hooks and Mahmassani 2000; Ben-Elia et al. 2013). However, in many cases, such as planning a route to attend a business meeting, people are not risk neutral, and the second aspect is crucial: the LET path may fail to meet driver's expectation if there is a large variance (i.e., risk) of travel time. To overcome this problem, the *mean-risk model* is developed for finding a path that minimizes the sum of a linear combination of mean and variance regarding the travel time (Lim et al. 2013; Nikolova and Stier-Moses 2014; Lianeas, Nikolova, and Stier-Moses 2016). Although risk is alleviated by this model, in many other cases of route planning, driver utilities are also governed by deadlines (Fan, Kalaba, and Moore II 2005; Samaranayake, Blandin, and Bayen 2012), such as catching flight, fire rescue and organ delivery, all of which need arriving on time with the maximum chance. These critical cases render the LET and mean-risk model based SSP impractical in real applications. Therefore, the *probability tail model* is proposed to find a path that maximizes the probability of reaching destination before a deadline (Nie and Wu 2009; Lim and Rus 2012; Cao et al. 2016b). This model is promising in that it integrates travel time, risk and deadline.

Unfortunately, the theory and methods for the LET and mean-risk settings cannot be directly extended to the probability tail model. In particular, the principle of optimality breaks down, precluding dynamic programming approaches, which is NP-hard in nature (Nikolova et al. 2006). Although a number of algorithms to address this probability tail model based SSP problem have been proposed (Lim and Rus 2012; Lim et al. 2013), they are limited by strong assumptions, such as Gaussian distribution of travel time, independence among travel time on different road links, and relatively large deadlines, which may not hold in real situations. A data-driven solution was proposed (Cao et al. 2016b) by formulating the path finding as a cardinality minimization problem, which is in turn approximated as a mixed integer linear programming (MILP) problem. This method circumvents the above strong assumptions, but in the meanwhile significantly decreases computation efficiency. Additionally, its approximate solution leaves room for further improvement in terms of accuracy in finding the real optimal path.

In this paper, we propose a novel method without strong assumptions. It is based on the key insight that the problem can be reformulated in such a way that dynamic programming approaches do apply. A disadvantage of this reformulation is that it may blow up the state space by taking the cartesian product of the set of nodes (intersections of the road network) and the set of possible amounts of time-to-deadline (i.e., time left before deadline). This is especially problematic in the typical setting where the latter set is continuous. To overcome this problem, we propose to generalize over the continuous space of remaining time by using neural networks to approximate the value function. To find a solution for the reformulated problem, we consider the Q-learning method (Watkins and Dayan 1992; Wiering 2000). It is designed to maximize the probability of arriving on time, which is approximated as the ratio between the number of times in which the vehicle reaches destination before deadline, i.e., success, and the total number of travels.

Overall, the main benefit of the proposed method is its practicality, in that it can be readily applied in the real world, specifically: 1) it is computationally feasible even for large road networks, 2) it is interpretable by the users: the converged Q-values have the practical meaning as the actual probabilities of reaching destination before deadline, 3) it enables the capability of providing time-dependent path recommendations, 4) it directly utilizes available travel time data and does not require any strong assumptions. We conduct extensive experiments on both artificial and real road networks, and the results justify the significant advantages of our method over others. Thus, our solution has high potential to be deployed in real vehicles for field test.

Background

The probability tail model based SSP problem can be mathematically described as follows: in a directed graph $G = (\mathcal{V}, \mathcal{L})$, $\mathcal{V}$ is a set of nodes representing road intersections, $\mathcal{L}$ is a set of edges representing road links, and o,$d \in \mathcal{V}$ represent the origin and destination, respectively. The objective is to maximize $Prob(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} \leq \tau)$, where $\vec{\mathbf{w}}$ is a random vector containing the travel time for each road link, τ is the time-to-deadline (i.e., the deadline is defined by user), and $\vec{\mathbf{x}}$ is a set of road links in which an element is "1" if the road link is on the corresponding path (Cao et al. 2016b). This problem is difficult to solve efficiently as it does not follow any typical optimization forms, e.g., convex optimization.

Rather than treating the problem as a monolithic non-convex optimization problem, it can also be treated as a sequential decision making problem. In particular, in the next section we will describe how the problem can be modeled as a *Markov decision process (MDP)* (Puterman 1994). An MDP is defined by a five-tuple $(S, A, P_{s,a}^{s'}, R(s), \gamma \in [0, 1])$, where S represents the state space ($s, s' \in S$), A represents the set of actions, $P_{s,a}^{s'}$ represents the transition distribution from s to s' by action a ($a \in A$), $R(s)$ represents the reward at s, and γ is the discount factor. The MDP aims to find an optimal policy π^* which maps states to actions in such a way that the expected cumulative discounted reward is maximized. In cases where the MDP is not known in advance, *reinforcement learning* (RL) methods can be used to learn the optimal policy (Sutton and Barto 1998). For instance, Q-learning (Watkins and Dayan 1992) repeatedly applies the following equation to sampled transitions (s, a, s', r):

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s') + \gamma \max_{a'} Q(s', a') - Q(s,a)], \qquad (1)$$

with learning rate $\alpha \in (0, 1]$. This can be shown to converge to an optimal Q-function: the converged Q-values represent the expected summation of discounted future reward, i.e., $Q^*(s, a) = E\left(\sum_{k=0}^{\infty} \gamma^k R_k\right)$. From Q^*, the optimal policy can be extracted by taking the greedy action in each state. (Please refer to (Sutton and Barto 1998) for more details.)

Basic Q-Learning for Discrete Deadlines

Before dealing with the more complex and realistic case of continuous time-to-deadline, we treat the case where each time-to-deadline takes a value from a discrete finite set. This allows for a relatively straightforward, but effective formulation of the problem as a discrete MDP, and thus the application of canonical RL algorithms, e.g., Q-learning.

MDP Formulation for Probability Tail Model

To solve the probability tail model based SSP problem by the Q-learning method, each success of reaching destination before deadline is considered as a reward, each pair of intersection and time-to-deadline are considered as a state, and the driving direction at each intersection is considered as an action. Naturally, $s = \langle v, \tau \rangle$, $s' = \langle v', \tau' \rangle$, $v, v' \in \mathcal{V}$, and v' be the succeeding intersection of v. Γ ($\tau, \tau' \in \Gamma$) is the set of time-to-deadlines, and we have $\tau - t_{v,v'} = \tau'$, where $t_{v,v'}$ is the random travel time on road link $l_{v,v'}$ ($l_{v,v'} \in \mathcal{L}$), and τ' is the remaining time-to-deadline at v'. Note that, τ at origin o is determined by the user, and the remaining time-to-deadline τ' at intermediate intersection v' between origin o and destination d is determined by τ and travel time cost $t_{v,v'}$ on the associated road links. A represents driving directions. $P_{s,a}^{s'} = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the distribution of $t_{v,v'}$ by action a such that the vehicle will move from intersection v with time-to-deadline τ at step t, to intersection v' with time-to-deadline τ' at step $t + 1$. $R(s')$ is the immediate reward received after transiting to intersection v' with time-to-deadline τ' from intersection v with time-to-deadline τ. The reward depends on whether the vehicle arrives on time.

Q-Value Representation and Path Planning

The just defined MDP has a particularly nice property: we can define the reward and discount factor in such a way that the converged Q-value represents the probability of arriving on time. Specifically, we denote the immediate reward as $R(s') \in \{0, 1\}$ with $R(s') = 1$ if and only if $v' = d$ and $\tau - t_{v,v'} \geq 0$. Thus, the immediate reward is 1 only at the intersection node preceding d if the vehicle can arrive at the destination before the deadline; otherwise, $R(s') = 0$. We set the discount factor γ to 1. (General Q-learning with $\gamma = 1$ does not necessarily ensure convergence, but for the SSP problem, convergence can be guaranteed with $\gamma = 1$ (Yu and Bertsekas 2013)).

In this case, after an o-d pair is determined, we update Q-value using Eq. (1) for each intersection, starting with v (i.e., $v = o$) and the user-defined time-to-deadline τ in each episode. To transit to next intersection v', we employ the action selection policy, i.e., Softmax strategy (Sutton and Barto 1998). This strategy balances between exploration and exploitation for the candidate succeeding intersections. Then, we use $P_{s,a}^{s'}$ to sample $t_{v,v'}$ to decide τ' at v'. These steps are repeated until all Q-values converge.

The converged Q-values in our problem can be shown to be the probabilities of arriving on time in the following. From Eq. (1), Q-value converges when $Q_{k+1}(s, a) = Q_k(s, a)$, $\forall s$ and $\forall a$. After the convergence, we have the Q-value defined as $Q^*(s, a)$ given by $Q^*(s, a) = Q^*(s, a) + \alpha_k[R(s') + \gamma \max_{a'} Q^*(s', a') - Q^*(s, a)]$. After a simple combination and elimination, we have

$$Q^*(s,a) = R(s') + \gamma \max_{a'} Q^*(s', a'). \qquad (2)$$

As this equation will be executed for a large number of times and it will converge to $Q^*(s,a) = E(R(s')) + \gamma \max_{a'} Q^*(s',a')$, which is exactly the probability of arriving on time.

When Q-learning has converged, we can obtain an optimal path by first stopping the Softmax strategy and then determining the best action a^* at a specified intersection with a certain deadline as $a^*(\langle v,\tau\rangle) = \arg\max_a Q(\langle v,\tau\rangle, a)$. We first compute $a^*(\langle o,\tau\rangle)$ to determine the optimal driving direction at origin o, i.e., the current state, with a user-defined deadline. The direction will determine the next intersection and the remaining time-to-deadline, i.e., next state. Then, we input the second intersection with the remaining time-to-deadline, to find the next intersection. The same process is repeated until the destination is reached. Thus, the complete optimal path can be found.

Q-Learning for Continuous Deadlines

The basic version of our Q-learning method introduced in the previous section will become prohibitively time-consuming in the case of continuous deadlines (i.e., the time-to-deadline is also continuous) and large-scale road networks due to the huge size of state space. To address those challenges, we develop a practical value function approximation method based on the dynamic neural network to directly learn the optimal Q-values, which approximates the probability of arriving at the destination on time. In the following, we will introduce the value function update scheme, function approximation method and the deployment steps.

Value Function Update Scheme

In order to deal with continuous deadlines and large scale networks, we use approximator $f(\cdot)$ to represent the value function $V(s)$, where $V(s) = \max_a Q(s,a)$. We propose to exploit the special structure of our problem: even for large networks with thousands of nodes, the locations can still be easily enumerated. We therefore apply the value function fitting operator by enumerating all the locations, but sampling from the travel time distributions. This means that all locations get the same amount of coverage, leading to a good approximation in entire network[1]. The value function with any time-to-deadline for any intersection is represented as:

$$V_{k+1}(s) = \max_a \left\{ \sum P_{s,a}^{s'} [V_k(s') + R(s')] \right\}, \tag{3}$$

where $0 \le V(s') \le 1$, $V(s') = 0$ if $v' = d$, $s = \langle v,\tau\rangle$, and $s' = \langle v',\tau'\rangle$. Eq. (3) uses the expected function value at s' (for the succeeding location) in the k^{th} iteration to compute the function value at s in the $(k+1)^{th}$ iteration. In particular, $V_k(s')$ on the right hand side of Eq. (3) is computed through the approximator $f_k(\cdot)$ in that iteration.

Function Approximation through Neural Network

To approximate $V_k(\cdot)$, we adopt a two-layer dynamic neural network[2] to learn function $f_k(\cdot)$ in each iteration as the probability of arriving on time for a given intersection with a given deadline. Specifically, $f_k(\cdot)$ is expressed as follows:

$$f_k(\vec{\mathbf{z}}) = g_2(g_1(\vec{\mathbf{z}} \cdot \vec{\omega}_1 + u_1) \cdot \vec{\omega}_2 + u_2), \tag{4}$$

where $\vec{\mathbf{z}}$ is the feature vector; $\vec{\omega}_1$, u_1, $\vec{\omega}_2$, and u_2 are parameters of the neural network; $g_1(\cdot)$ and $g_2(\cdot)$ are activation functions.

The feature vector $\vec{\mathbf{z}}$ used to represent s is defined as follows: $\vec{\mathbf{z}}(s) = \langle \tau, \vec{\mu}(v), \vec{\sigma}(v)\rangle$, where τ and v are the time-to-deadline and location in s; $\vec{\mu}$ and $\vec{\sigma}$ are the means and standard deviations of travel time for the K-shortest paths from current location to destination. Note that, while we enumerate locations to guarantee a sufficient number of backup operations for each location, the neural network featurizes the locations in order to generalize its prediction over locations. The rationale for $\vec{\mu}(v)$ and $\vec{\sigma}(v)$ is that the probability of arriving on time is highly associated with the mean and standard deviation of travel time. $\vec{\mu}(v)$ and $\vec{\sigma}(v)$ can be easily computed based on the travel time data on each road link. Consequently, the feature of a training sample for s can be expressed as $\vec{\mathbf{z}} = [\tau, \vec{\mu}(v), \vec{\sigma}(v)]^\top$, where the length of $\vec{\mu}(v)$ and $\vec{\sigma}(v)$ is K. Since $f(\vec{\mathbf{z}})$ will be applied for continuous deadlines, τ in training samples should cover extensive values. Therefore, we randomly select $\mathcal{N}$ deadlines, each of which is denoted by τ_i, and we have $\tau_i = \beta \cdot T_e$, where β is the deadline parameter, and $\beta \in [0,2]$. T_e is the least expected travel time from v to d. Note that larger β implies loose deadlines, and vice versa. Thus, the entire feature vector of an individual intersection v is expressed as:

$$\vec{\mathbf{Z}}_s = [\tau_1, \vec{\mu}(v), \vec{\sigma}(v); \cdots; \tau_{\mathcal{N}}, \vec{\mu}(v), \vec{\sigma}(v)]^\top. \tag{5}$$

To dynamically train the neural network at the $(k{+}1)^{th}$ iteration, we can set $V_k(s)$ on the right-hand side of Eq. (3) by the learned $f_k(\vec{\mathbf{z}})$, and $V_{k+1}(s)$ on the left-hand side can then be updated accordingly. This value will be adopted as the new label to train $f_{k+1}(\vec{\mathbf{z}})$ in the next iteration through back propagation. This process is repeated until convergence.

Core Deployment

Considering both the value function update scheme and function approximation method, we deploy the core part (i.e., training value function) of our method as shown in Alg. 1. Specifically, Lines 1-2 initialize parameters and prepare feature values. Lines 3-20 dynamically learn value functions for each intersection using the neural network. In particular, for each intersection, the trained neural network from the preceding iteration computes the value functions of the succeeding intersections, which are then used to update the value function of the current intersection (Lines 5-14). In Lines 15-16, convergence errors of all intersections are accumulated. In Lines 17-18, the average convergence error is updated to determine the loop termination. In Lines 19-20, the neural network is trained using feature values and updated labels.

To find the optimal path before departure (which is preferred in a real application), the best action at a certain intersection with a given deadline is determined by:

$$a^*(s) = \arg\max_a \left\{ \sum P_{s,a}^{s'} [V^*(s') + R(s')] \right\}, \tag{6}$$

[1]To save computation time, utilizing a subset of locations is also feasible, which is shown in the experiments, i.e., Fig. 4(a).

[2]Other supervised machine learning methods, such as support vector regression, can also be employed for function approximation. We use the dynamic neural network here because it can naturally output values between zero and one (due to its sigmoid function) for representing the probabilities of arriving on time.

```
input : $G = (\mathcal{V}, \mathcal{E})$, road network;
        $\Omega_{v,v'}$, set of travel time data on $l_{v,v'}$;
        $N$, size of travel time data on each road link;
        $\mathbb{N}$, the configured neural network; $o$-$d$ pair;
        $\epsilon_1$, the convergence error; $k$, the iteration number.
1  Initialize $V_0(s)$ via warm start; and set $\epsilon_1 = 1$; $\epsilon_2 = 0$;
   $k = 0$; $N = 1000$; $\zeta = 0.025$;
2  Import feature data $\vec{\mathbf{Z}}_s$ for each $s$ by Eq. (5);
3  while $\epsilon_1 > 0.01$ do
4      foreach $v \in \mathcal{V}$ do
5          foreach $v'$ that succeeds $v$ do
6              Sample $N$ travel time $t_{v,v'}$ out of $\Omega_{v,v'}$;
7              if $v' \neq d$ then
8                  if $k \geq 1$ then
9                      Calculate $V_k(s')$ in Eq. (3) through
                       the trained $f_k(\vec{\mathbf{z}})$ in Eq. (4);
10                 else
11                     $V_k(s')$ = initial values $V_0(s')$;
12             else
13                 Compute $R_k(s')$ in Eq. (3);
14         Update $V_{k+1}(s)$ in Eq. (3);
15         if $k \geq 1$ then
16             Update $\epsilon_2 = \epsilon_2 + |V_{k+1}(s) - V_k(s)|$;
17     if $k \geq 1$ then
18         Update $\epsilon_1 = \epsilon_2 / |\mathcal{V}|$; Reset $\epsilon_2 = 0$;
19     Learn $f_{k+1}(\vec{\mathbf{z}})$ by incorporating feature $\vec{\mathbf{Z}}_s$ and new
       label $V_{k+1}(\langle v, \tau \rangle)$ for each $v$ into $\mathbb{N}$;
20     Update iteration number: $k = k + 1$;
output: Converged $V^*(s)$ for each $\langle v, \tau \rangle$.
```

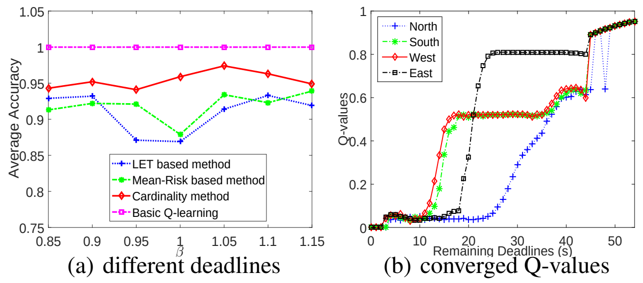
Algorithm 1: Value Function Training

where $P^{s'}_{s,a}$ is represented by travel time data on corresponding road link. The next state s' (which consists of v' and τ') can be decided as follows: v' is the succeeding intersection, and $\tau' = \tau - E(t_{v,v'})$, where $E(t_{v,v'})$ is the expected travel time over the previously chosen road link. Then we iteratively use the same equation to compute the next action until the complete path is achieved.

Other Practical Considerations

Here we discuss other practical features considered in our method, including time-dependent path recommendation, simultaneous training for multiple destinations and accelerating the training process.

If we collect the data $P^{s'}_{s,a}$ according to a specified time period, such as peak hour of 7am-9am, our Q-learning method is able to provide time-dependent path recommendation. Also note that our Q-learning method adopts Alg. 1 to dynamically learn the value function for a specified destination from all nodes. If the destination is changed, Alg. 1 needs to run again to learn a new value function. To better generalize our Q-learning method, we randomly select multiple destinations on the road network, and concatenate their feature values to the same matrix (i.e., $\vec{\mathbf{Z}}_s$ in Eq. (5)). Then we only need to run Alg. 1 once, and the learned value function will be applied to any other destinations on



(a) different deadlines (b) converged Q-values

Figure 1: Artificial Network with Discrete Deadlines

the same road network. When the destination is changed, we just change the input features (i.e., $\vec{\mu}$ and $\vec{\sigma}$ of the K-shortest paths to the new destination) to the trained neural network. In this way, our Q-learning method is able to calculate the probability of arriving on time from any origin(s) to any destination(s).

Traditionally, value function $V_0(\langle v, \tau \rangle)$ is initialized as 0, which may slow down the convergence speed (Carden 2014). Therefore, we propose a *warm start* strategy[3] that approximates the probability of arriving on time for vehicles at intersection v with time-to-deadline τ as follows: $V_0(\langle v, \tau \rangle) = 1/(1 + e^{-\zeta(\tau - T_e)})$, where ζ is the coefficient. The rationale is that we need to increase the probability from 0 to 1 (that is why sigmoid function is used) as τ increases.

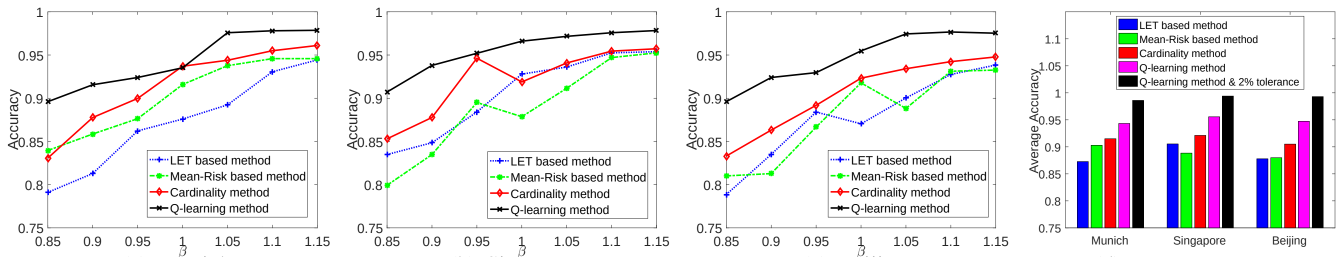
Experimentation

We conduct extensive experiments to evaluate our proposed Q-learning method and compare with other methods on various road networks. All experiments are performed on a typical PC with Intel Core i7-3540M processor and 16GB RAM.

Artificial Network with Discrete Deadlines

We first test the basic version of our Q-learning method for discrete deadlines on an artificial road network – a grid with 20×20 intersections. We use $m_1 = 15$ as mean and $\sigma_1 = 3$ as standard deviation to randomly generate the mean travel time m_2 for each individual road link. We then adopt m_2 and $\sigma_2 = 0.3m_2$ to generate 200 instances of travel time data for corresponding links to represent $P^{s'}_{s,a}$. All travel time is rounded up as positive integers in seconds. At each intersection, there are four travel directions (north, south, west and east). We compare with three methods: (1) LET based method computes a path of the least expected travel time based on the travel time data; (2) Mean-Risk based method computes a path with minimal value of $\mu_p + \lambda\nu_p$, where μ_p is the expected travel time, ν_p is the variance, and λ is the coefficient (Nikolova and Stier-Moses 2014); and (3) Cardinality method approximates and computes a path of the probability tail model by formulating it as the MILP problem which is further solved using a partial Lagrange multiplier method (Cao et al. 2016c). We randomly select 100 o-d pairs and

[3]This warm start strategy is also applicable to our basic version of the Q-learning method for discrete deadlines as long as the Q_0 is calculated in the same way for V_0.



(a) Munich (b) Singapore (c) Beijing (d) average accuracy

Figure 2: Accuracy Results on the Three Real Road Networks with Continuous Deadlines

use deadline parameters $\beta = 0.85, 0.90, \ldots, 1.10, 1.15$. For each o-d pair and specified deadline, we define the ground-truth path (achieved through enumerating all the paths) as the one having the maximal number of times of not being late, given the 200 instances of data on each link.

The accuracy of finding the ground-truth path is plotted in Fig. 1(a) (**all figures are best viewed in color**). For all deadlines, our Q-learning method always achieves average accuracy of 100%, confirming that the converged Q-values are able to represent probabilities of arriving on time. The cardinality method obtains the average accuracy of about 95% as it adopts ℓ_1-norm to approximately minimize the frequency of not being late. The LET and Mean-Risk based methods obtain the average accuracy of around 88%. The high accuracy of our method also shows that the recommended path guarantees arriving on time more often than others. This advantage comes from the fact that our method incorporates the event of arriving on time in the reward and objective.

The accuracy of our method can also be demonstrated by the plot of converged Q-values for an intersection which is two grids to the north of the destination as shown in Fig. 1(b). The converged Q-values in [0, 1] denote the probabilities of arriving on time from the current location by taking a corresponding action. Meanwhile, the optimal action usually changes with deadlines, e.g., traveling west is optimal when the time-to-deadline is between 10 and 20 and traveling east is optimal when the time-to-deadline is larger than 20. The Q-value for traveling south is not higher than that of traveling west and east although the current location is north to the destination. This is because the cost of traveling south is always large according to the generated traffic data. So, traveling south yields a lower chance of arriving on time. We also see that the Q-values for all actions are 0 when time-to-deadline is less than 4, which is too tight. Thus, we conclude that an optimal path depends strongly on deadline even though an origin is fixed. This factor is not incorporated in LET or Mean-Risk based methods, which is why their average accuracy fluctuates when deadline is varied.

Real Networks with Continuous Deadlines

To verify our Q-learning method for the probability tail model with continuous deadlines, we perform experiments on three large road networks extracted from the city maps of Munich, Singapore, and Beijing, as summarized in Table 1. We randomly select 200 o-d pairs on each network, and the average minimal number of road links between each origin and destination is given in the third row, which generally reflects the minimum quantity of decision-making to find an optimal path. Then, we prepare 1,000 instances of travel time data for each road link: (1) On Munich network, we use actual length of each road link to divide the collected real travel speed of vehicles from July 2014 to March 2015[4]; (2) On Singapore network, we use actual length of each road link as the mean to randomly generate travel time data, and the standard deviation is 0.3 times of the length[5]; (3) On Beijing network, we directly use the travel time data collected from real travel trajectories of taxi from September to October 2013 (Wang, Zheng, and Xue 2014). For our method, we randomly select 100 destinations on a network and concatenate their feature values to a same matrix (i.e., $\vec{\mathbf{Z}}_s$ in Eq. (5)). Note that we do not explicitly explore the dependence or correlation of travel time on different road links, because it will be naturally included in the data set if there is any, and the travel time data is the direct input to our method.

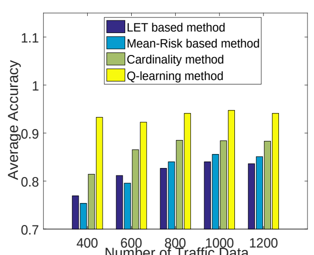
Table 1: Settings of the Three Real Road Networks

	Munich	**Singapore**	**Beijing**
#Nodes	51,517	6,476	129,607
#Links	115,651	10,225	294,868
#Links between o-d	221	86	358

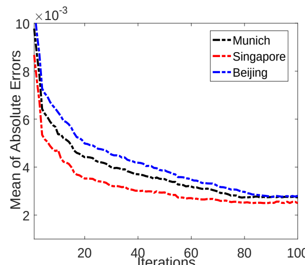
Accuracy The accuracy of each method on the three networks are plotted in Figs. 2(a-c). Our Q-learning method achieves around 95% accuracy, higher than all other methods for each deadline. Compared with the results in Fig. 2(a), our method achieves slightly lower accuracy, because, to handle continuous deadlines, a neural network is used to learn value functions in which learning error may exist. Besides, the three real networks are considerably larger than the grid. The accuracy of the cardinality method is higher than the LET and Mean-Risk based methods. Moreover, the accuracy of all methods becomes lower as deadline becomes tighter. Our method is less affected because it dynamically updates Q-values according to each specified deadline.

[4]This data set is provided by the BMW Group, Germany.

[5]All relevant information is obtained from OpenStreetMap.

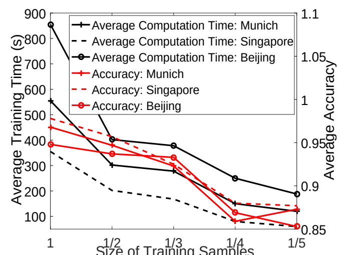


(a) different data size

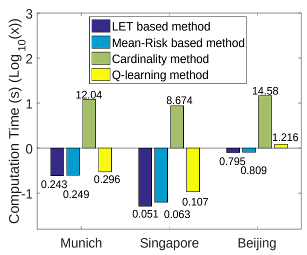


(b) mean absolute error

Figure 3: Accuracy Changes with Data Size and Iterations



(a) different training sample size



(b) average computation time

Figure 4: Computation Time

We also plot in Fig. 2(d) the overall accuracy of all methods as well as the 2%-tolerance accuracy[6] of our Q-learning method. The overall accuracy on Munich and Beijing is slightly lower than that on Singapore for most of the methods. This is because the sizes of Munich and Beijing networks are much larger, so as the number of possible paths between an o-d pair, thus making it more difficult to achieve the ground-truth path. In particular, our method will determine a road link to traverse according to Eq. (6) at each intersection. Even if only one road link is not on the ground-truth path, the returned path is not counted as accurate. In the case with at least 358 intersections on average (Table 1) for each path finding on Beijing network, the overall accuracy of around 95% is sufficiently high. Moreover, the 2%-tolerance accuracy of our method is nearly 100%, indicating that the paths found by our method is of high quality even if they might not be the ground-truth path.

We also test the accuracy against the quantity of travel time data on each road link by taking Beijing network as an example since its size is largest. As shown in Fig. 3(a), our method is almost not affected by the quantity of travel time data. By contrast, the accuracy of other methods is reduced when data size is small. Moreover, in each iteration of the value function learning, we record the mean absolute error (i.e., the difference between the neural network output and the target value) on the three road networks, shown in Fig. 3(b). The average absolute error becomes smaller as the iteration increases. This clearly justifies the efficacy of the dynamic neural network to learn accurate value functions.

Table 2: Accuracy of Time-Dependent Q-learning (%)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
week	94.8	93.7	95.1	93.3	94.5	92.9	90.8	91.6
day	96.1	95.0	94.2	95.5	95.8	96.3	94.1	93.2
hour	97.4	97.7	97.0	96.2	97.3	98.2	96.5	96.1

Our Q-learning method is data-driven and easy to execute in a time-dependent manner, which further improves the performance of routing service, i.e., avoiding frequent value function learning while maintaining an acceptable level of accuracy. To justify, we explore the travel time data of eight continuous weeks (July-August 2014) on Munich network and further extract eight segments according to different time units: (1) **week**, we divide the data into eight weeks, and each segment represents a week, (2) **day**, we extract the data of eight Mondays separately, and each segment represents a Monday, (3) **hour**, we extract the data of 7am-9am (i.e., peak hours) on each Monday separately, and each segment represents a span of peak hour. For all of the three units, we apply our method only on the first segment of the traffic data and then use the learned value functions to perform routing service based on all of the eight segments accordingly. The accuracy results are recorded in Table 2.

[6]If the difference between the probability of arriving on time for the path returned by our method and the probability for the ground-truth path is less than 2%, then the returned path is considered the same as the ground-truth path as the difference is very marginal.

We can observe that, segment 1 usually achieves higher accuracy than that of the other seven segments. This is because the training data and testing data are the same for segment 1. Generally, as the segment index increases, the accuracy will decrease, but the deterioration is very slight, especially for the **hour**, which achieves the accuracy higher than 96% even for week 8. Another remarkable observation is that the higher the time resolution, the better the accuracy. This happens because the high time resolution usually captures the traffic characteristics better, i.e., the peak hour pattern captured by the **hour** unit. Thus, we can conclude that the time-dependent Q-learning method helps to avoid frequent value function learning while achieving good accuracy, especially for the time unit **hour**.

Computation Time Normally, the computation time in each iteration of the dynamic neural network is associated with the size of training samples. Previously, we use training samples of all nodes to learn the value function. To improve computation efficiency, we may only use samples from some nodes, while maintaining an acceptable level of accuracy. Thus, we evaluate the average computation time of our method in each iteration and its accuracy with respect to different sizes of training samples, as plotted in Fig. 4(a). As expected, the computation time and the accuracy decrease as the training sample size reduces. On Beijing network, the deterioration of accuracy is only around 1% while the reduction of computation time is about 360 seconds (i.e., almost half) as we use no less than 1/3 of the training samples. Note that we did not count the computation time of the K-shortest paths in our method, because normally those paths are static, which can be stored in a look-up table.

We also record the average computation time of each routing for all methods in Fig. 4(b). We only count the time of path finding for the Q-learning method since learning value function can be done offline. The LET and Mean-Risk based methods have the shortest average computation time, which slightly increases as network size increases. Caused by solving the MILP problem, the cardinality method needs much longer time (around 15 seconds) to compute a path on Beijing network. By contrast, our Q-learning method takes slightly longer than the LET and Mean-Risk based methods. It takes only 1.216 seconds to obtain an optimal path on Beijing network, which is highly efficient.

Conclusion and Future Work

In this paper, we designed a practical Q-learning method to efficiently and accurately solve the probability tail model based SSP problem. Compared with other baselines, our method offers several important practical features: 1) It considers travel time, risk and deadlines simultaneously when computing an optimal path; 2) It provides probability of arriving on time from any origin(s) to any destination(s) once the value function is learned; 3) It can flexibly provide time-dependent path recommendation and avoid frequent training even if travel time data changes. For future work, we plan to consider other practical factors (e.g., weather condition) in each state and deploy our method in vehicles for field test.

Acknowledgment

This work is supported by the BMW Group, Germany, and also partially supported by the MOE AcRF Tier 1 funding (M4011261.020) awarded to Dr. Jie Zhang, and NWO Innovational Research Incentives Scheme Veni (639.021.336) awarded to Dr. Frans Oliehoek.

References

Anantharam, P.; Thirunarayan, K.; Marupudi, S.; Sheth, A.; and Banerjee, T. 2016. Understanding city traffic dynamics utilizing sensor and textual observations. In *Proceedings of The Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, 3793–3799.

Ben-Elia, E.; Di Pace, R.; Bifulco, G. N.; and Shiftan, Y. 2013. The impact of travel informations accuracy on route-choice. *Transportation Research Part C: Emerging Technologies* 26:146–159.

Cao, Z.; Guo, H.; Zhang, J.; and Fastenrath, U. 2016a. Multiagent-based route guidance for increasing the chance of arrival on time. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, 1180–1187.

Cao, Z.; Guo, H.; Zhang, J.; Niyato, D.; and Fastenrath, U. 2016b. Finding the shortest path in stochastic vehicle routing: a cardinality minimization approach. *IEEE Transactions on Intelligent Transportation Systems* 17(6):1688–1702.

Cao, Z.; Guo, H.; Zhang, J.; Niyato, D.; and Fastenrath, U. 2016c. Improving the efficiency of stochastic vehicle routing: A partial lagrange multiplier method. *IEEE Transactions on Vehicular Technology* 65(6):3993–4005.

Cao, Z.; Jiang, S.; Zhang, J.; and Guo, H. 2016d. A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion. *IEEE Transactions on Intelligent Transportation Systems* PP(99):1–16.

Carden, S. 2014. Convergence of a q-learning variant for continuous states and actions. *Journal of Artificial Intelligence Research* 49:705–731.

Fan, Y.; Kalaba, R.; and Moore II, J. 2005. Arriving on time. *Journal of Optimization Theory and Applications* 127(3):497–513.

Lianeas, T.; Nikolova, E.; and Stier-Moses, N. E. 2016. Asymptotically tight bounds for inefficiency in risk-averse selfish routing. In *Proceedings of the Twenty-Fifth international joint conference on Artificial Intelligence (IJCAI)*, 338–344.

Lim, S., and Rus, D. 2012. Stochastic motion planning with path constraints and application to optimal agent, resource, and route planning. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 4814–4821.

Lim, S.; Sommer, C.; Nikolova, E.; and Rus, D. 2013. Practical route planning under delay uncertainty: Stochastic shortest path queries. In *Robotics: Science and Systems*, volume 8, 249–256.

Miller-Hooks, E. D., and Mahmassani, H. S. 2000. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science* 34(2):198–215.

Nie, Y. M., and Wu, X. 2009. Shortest path problem considering on-time arrival probability. *Transportation Research Part B: Methodological* 43(6):597–613.

Nikolova, E., and Stier-Moses, N. E. 2014. A mean-risk model for the traffic assignment problem with stochastic travel times. *Operations Research* 62(2):366–382.

Nikolova, E.; Kelner, J. A.; Brand, M.; and Mitzenmacher, M. 2006. Stochastic shortest paths via quasi-convex maximization. In *Proceedings of European Symposium of Algorithms*, 552–563.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

Samaranayake, S.; Blandin, S.; and Bayen, A. 2012. A tractable class of algorithms for reliable routing in stochastic networks. *Transportation Research Part C: Emerging Technologies* 20(1):199–217.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Wang, Y.; Zheng, Y.; and Xue, Y. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 25–34.

Watkins, C. J. C. H., and Dayan, P. 1992. Technical note: Q-learning. *Machine Learning* 8(3-4):279–292.

Wiering, M. 2000. Multi-agent reinforcement learning for traffic light control. In *Proceedings of the 17th International Conference Machine Learning (ICML)*, 1151–1158.

Wu, X.; Sheldon, D.; and Zilberstein, S. 2016. Optimizing resilience in large scale networks. In *Proceedings of The Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, 3922–3928.

Yu, H., and Bertsekas, D. P. 2013. On boundedness of q-learning iterates for stochastic shortest path problems. *Mathematics of Operations Research* 38(2):209–227.