3-2020

Using reinforcement learning to minimize the probability of delay occurrence in transportation

Zhiguang CAO
Singapore Management University, zgcao@smu.edu.sg

Hongliang Guo

Wen Song

Kaizhou Gao

Zhengghua Chen

See next page for additional authors

Citation

CAO, Zhiguang; Guo, Hongliang; Song, Wen; Gao, Kaizhou; Chen, Zhengghua; Zhang, Le; and Zhang, Xuexi. Using reinforcement learning to minimize the probability of delay occurrence in transportation. (2020). *IEEE Transactions on Vehicular Technology*. 69, (3), 2424-2436.
Available at: https://ink.library.smu.edu.sg/sis_research/8158

Using Reinforcement Learning to Minimize the Probability of Delay Occurrence in Transportation

Zhiguang Cao, Hongliang Guo*, Wen Song, Kaizhou Gao, Zhenghua Chen, Le Zhang, Xuexi Zhang

***Abstract*—Reducing traffic delay is of crucial importance for the development of sustainable transportation systems, which is a challenging task in the studies of stochastic shortest path (SSP) problem. Existing methods based on the probability tail model to solve the SSP problem, seek for the path that minimizes the probability of delay occurrence, which is equal to maximizing the probability of reaching the destination before a deadline (i.e., arriving on time). However, they suffer from low accuracy or high computational cost. Therefore, we design a novel and practical Q-learning approach where the converged Q-values have the practical meaning as the actual probabilities of arriving on time so as to improve the accuracy of finding the real optimal path. By further adopting dynamic neural networks to learn the value function, our approach can scale well to large road networks with arbitrary deadlines. Moreover, our approach is flexible to implement in a time dependent manner, which further improves the performance of returned path. Experimental results on some road networks with real mobility data, such as Beijing, Munich and Singapore, demonstrate the significant advantages of the proposed approach over other methods.**

I. Introduction

Transportation and mobility are crucial to the sustainable development of any city. With the ever-increasing demands for mobility and modern logistics, the vehicle population has been steadily growing over the past several years from 920 million units in 2006 to 1.3 billion units in 2015 [1]. One of the adverse consequences of the vehicle population growth is the occurrence of traffic delay, which usually causes huge economic loss and inestimable environmental damage [2], [3]. Addressing the issue of traffic delay attracts broad attention from government, industry and research community due to its high relevance to people's daily life. Among all solutions, studying the stochastic shortest path (SSP) problem is considered as an efficient way to mitigate the traffic delay and enhance the driver's mobility experience, which may also help to alleviate traffic congestion [4], [5]. On the other hand, as one of the most attractive emerging technologies, reinforcement learning has been widely and successfully adopted in various scenarios, e.g., Alphago [6], learning actions for autonomous vehicles [7], and solving combinatorial optimization problem [8]. In this paper, we focus on nourishing the solutions to the SSP problems by exploring the approach of reinforcement learning, which in particular, aims to minimize the probability of delay occurrence in transportation.

A. SSP Problem and Arriving on Time

Different from deterministic shortest path problem which seeks static routes (e.g., the shortest path), the stochastic shortest path problem takes into account various uncertainties in real-world traffic and yields route recommendations dynamically, leading to a better solution to the development of sustainable transport systems [9], [10]. The least expected time (LET) path was first proposed to address the stochastic shortest path problem, where a path is optimal if it guarantees least expected travel time [11]. Miller-Hooks and Mahmassani [12] solved the time dependent LET path problem with and without waiting policy. Then, Waller and Ziliaskopoulos [13] solved the LET path problem considering correlations between travel time of road links. The promising aspect about the LET path finding is that it can be converted into a deterministic problem which can be efficiently solved by traditional Dijkstra or A* algorithm. However, the LET path may fail to meet driver's expectation if there is a large variance of travel time. Therefore, Bell and Cassir [14] proposed and solved the problem of Mean-Risk path, which aims to find a path that minimizes the sum of linear combination of mean and variance regarding the travel time. Nikolova and Stier-Moses, and Lianeas et al. [15], [16] improved the computational efficiency of the Mean-Risk path by developing a quasi-convex optimization based solution. However, real-world transport scenarios often involve an important requirement of specific deadlines [17]. For example, in the cases of organ delivery, fire rescue, and catching up flights, drivers must reach destination before a specific deadline in which the LET or Mean-Risk path may fail to meet this requirement.

With the deadlines, Fan et al. [18] proposed the *probability tail model* where an optimal policy is defined as finding the next node to visit, so as to minimize the probability of delay occurrence for the whole path, which is also equal to maximizing the probability of arriving at destination before the deadline (i.e., arriving on time). This model has two attractive properties: 1) it takes the specific demand of deadline into account, giving drivers an extra dimension of settings and 2) it

* Corresponding Author: Hongliang Guo.

Zhiguang Cao is with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore (email: zhiguangcao@outlook.com).

Hongliang Guo is with the School of Automation Engineering, University of Electronic Science and Technology of China (email: guohl1983@uestc.edu.cn).

Wen Song is with the Institute of Marine Science and Technology, Shandong University, China (email: wensong@email.sdu.edu.cn).

Kaizhou Gao is with the Macau Institute of Systems Engineering, Macau University of Science and Technology, Macau, China (email: kzgao@must.edu.mo).

Zhenghua Chen and Le Zhang are with the Institute for Infocomm Research (I2R), Singapore(email: chen0832@e.ntu.edu.sg, zhangleuestc@gmail.com).

Xuexi Zhang is with the School of Automation, Guangdong University of Technology, China (email:zxxnet@gdut.edu.cn).

uses a probabilistic instead of deterministic metric to evaluate stochastic traffic situations, which is more realistic especially with uncertainties [18], [19]. Christman and Cassamano [20] presented a dynamic programming based approach to solve this problem. Particularly, given a source, destination, and time budget, the goal is to adaptively choose the next node to visit to maximize the probability of arriving to the destination on time. However, the computation efficiency is not justified in this work. To improve the computational efficiency of solving the probability tail model for the stochastic shortest path problem, several quasi-convex optimization based solutions were developed [19], [21], [22]. However, these solutions are limited by strong assumptions, such as Gaussian distribution of travel time, independence among travel time on different road links, and relatively large deadlines. These assumptions are imposed because the analytic formula of the model does not follow any canonical optimization forms, thus preventing its wide applications in real transport systems. To overcome these limitations, a data-driven solution was proposed [23], [24], [25], which formulates path finding as a cardinality minimization problem. The cardinality minimization is then approximately solved by reformulating it as a mixed integer linear programming (MILP) problem based on the travel time data of each road link. This data-driven approach circumvents strong assumptions, but significantly decreases computation efficiency. Additionally, its approximation solution leaves room for further improvement in accuracy for this arriving on time problem.

B. Adaptive and Heuristic Solutions to SSP Problem

The solutions in the preceding subsection always explicitly formulate the SSP problem as an optimization format, such as integer linear programming or mixed integer programming. Although those solutions can express various and complex constraints and objective functions conveniently, they suffer from prohibitive computation inefficiency. On the other hand, adaptive and heuristic solutions are attractive, because they can guarantee tractable computation for SSP problems of large scale, and less depend on optimization model. Eiger et al. [26] introduced a preference structure of paths over probabilistic networks, and proved that stochastic optimal path (SOP) under linear and exponential utility functions can be solved efficiently by the heuristic solutions with satisfactory accuracy, such as Dijkstra-like methods. Current et al. [27] proposed a solution to the stochastic shortest covering path problem, and solved it by a heuristic based on Lagrangian relaxation technique. They further developed an exact algorithm evolving from a branch and bound method, which can efficiently solve the problem. Bander and White [28] applied AO* algorithm to non-stationary stochastic shortest path problem, by using heuristic function as a lower bound of cost. Thus, a portion of the unnecessary states were eliminated, which significantly saved computation time in comparison to the dynamic programming approach. Rambha et al. [29] formulated an adaptive transit routing problem as finite horizon Markov decision process and solved it via a dynamic programming approach, before which, an elimination of the redundant individual states was adopted to exclude some unnecessary solution candidates. To circumvent the disadvantages (i.e., strong assumption and curse of dimensionality) of the parametric model based methods for adaptive routing, Mao and Shen [30] applied reinforcement learning as a non-parametric model-free method to minimize the average travel time in stochastic time-dependent network. The Q-learning method for both discrete and continuous state space demonstrated encouraging evidence for solving the real world routing problem. More recently, there is a trend to apply deep neural network or deep reinforcement learning for solving the variants of SSP problem, such as traveling salesman problem (TSP) or vehicle routing problem (VRP). In particular, those methods mainly adopt Pointer Network, self-attention, Transformer, reinforcement learning or their combinations to perform decision making for sequentially selecting nodes to visit next [31], [32], [33], [34].

Although the adaptive and heuristic solutions have achieved big success in the general SSP problem or its variants, none of them have been applied to solve the arriving on time based SSP problem. Therefore, it is significant to leverage the advantages of the probability tail model in SSP problem, and the model-free nature as well as the tractable computation of the adaptive and heuristic solutions.

C. Our Contribution

In this paper, we consider an adaptive solution, i.e., Q-learning approach, to solve the arriving on time based SSP problem. Q-learning is generally model-free, and it does not need to explore the traffic engineering knowledge to model an exponential-like function of local traffic condition. In contrast, Q-learning takes advantage of the Temporal-Difference (TD) prediction with a data-driven nature, which directly takes the real traffic data as input. Thus, it does not require the problem to follow any optimization forms, and inherently gets rid of those strong assumptions in conventional methods. On the other hand, relying only on trial data, Q-learning has been used in many traffic related problems such as traffic signal control [35], cooperative route planning [36], traffic flow control [37], and adaptive routing [30], but not for the arriving on time based stochastic shortest path problems. To apply Q-learning in our work, we need to address two additional challenges: 1) how to design the algorithm so as to achieve optimal or close-to-optimal solutions and 2) how to deal with the extremely large state space caused by infinite number of possible (continuous) deadlines and large-scale road networks.

Specifically, our proposed Q-learning approach 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 the destination before deadline, i.e., success, and the total number of travels. Each success event is considered as a *reward*. Moreover, road intersections (i.e., the locations of vehicle) and time deadlines in the probability tail model are naturally represented as *states*. In so doing, the converged Q-values have the practical meaning as the actual probabilities of arriving on time; hence exactly achieving the goal. Moreover, by employing a dynamic neural network to learn the value function, our approach scales well to large road networks and can support both discrete and continuous values of the deadlines. Moreover, our approach is flexible in

incorporating more relevant traffic factors and provide time dependent routing service. We conduct extensive experiments and the results on both artificial and real road networks justify the significant advantages of our approach over other methods.

We wish to note that this paper is an extension to our prior work [38]. The major extensions include: the proof of convergence for $\gamma = 1$ in our stochastic shortest path problem; the warm start strategy for both discrete deadlines and continuous deadlines, and their improvements by experimentation; one more basic Q-learning method as benchmark; and the enrichment of introduction and related work.

II. Q-learning for Discrete Deadlines

We first present the mathematical formulation of the probability tail model in the SSP problem, and then describe this arriving on time based SSP problem as a Markov Decision Process (MDP). Afterwards, we present the Q-learning approach to solve the MDP. Generally, Q-learning is tablewise [39]. Therefore, we start with a simpler case in this section in which each deadline takes a value from a discrete finite set (i.e., discrete time budget). The purpose is to clearly showcase the special design of our approach in which the converged Q-values have the practical meaning as the actual probabilities of arriving on time, achieving an optimal solution. We will then show in the next section on how to extend the basic model of the Q-learning approach to address continuous deadlines (i.e., continuous time budget) and large road networks by adopting dynamic neural networks.

A. Analytic Formula of Probability Tail Model

We start with a directed graph $G = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of nodes, representing road intersections, and $\mathcal{L}$ is the set of arcs, representing road links. Besides, we have o,$d \in \mathcal{V}$, representing the origin and destination, respectively. Thus, the objective considered in the probability tail model can be formulated as follows [24], [20]:

$$\max_{\vec{\mathbf{x}}} Prob(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} \leq \tau) \quad \Big| \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}};\ \vec{\mathbf{x}} \in \{0,1\}^{|\mathcal{L}|}, \qquad (1)$$

where $\vec{\mathbf{w}}$ is the vector containing the random travel times for the road links; $\mathbf{M}$ is the node-arc incidence matrix of G with size of $|\mathcal{V}| \times |\mathcal{L}|$, and each column of $\mathbf{M}$ corresponds to an arc, with the two ends (nodes) being 1 and -1, respectively, and the remaining elements being 0; τ is the time budget with respects to the user-defined deadline; $\vec{\mathbf{b}}$ is the o-d (i.e., origin-destination) with size of $|\mathcal{V}| \times 1$, all elements of which are zeros except for the origin o ("1") and the destination d ("-1"); and $\vec{\mathbf{x}}$ is the set of road links with size of $|\mathcal{L}| \times |1$, where an element is "1" if the road link is on the corresponding path. The equality constraint guarantees that $\vec{\mathbf{x}}$ is a connected path from the origin to the destination. Eq. (1) aims to maximize the probability of reaching destination before the deadline, which is equal to minimizing the probability of delay occurrence.

B. MDP Expression and the Q-learning Approach

The probability tail model in Eq. (1) is difficult to solve efficiently because it does not follow typical optimization forms, e.g., convex or quasi-convex optimization. However, it can be viewed from a data-driven perspective. Specifically, let a vehicle traverse on a path for N times. For each time, if the vehicle reaches the destination within the time budget τ, the counter is increased by one. Then, the probability of arriving on time is approximated as the ratio between the counter and N. Note that, in this paper, we adopt this ratio to represent the probability, thus the arriving on time based SSP problem can be naturally modeled as an MDP, which aims to find a path with the maximum probability of arriving on time, given any o-d pair and any deadline. In light of this, the Q-learning approach could be explored to solve this MDP [39].

Particularly, with respects to the arriving on time based SSP problem, the MDP is defined by a five-tuple (S, A, $P_{s,a}^{s'}$, $R(s)$, $\gamma \in [0,1]$), where,

- $S = \{\langle v_1, \tau_1\rangle, \langle v_2, \tau_2\rangle, \ldots\}$ represents the state space. The states $s, s' \in S$, $s = \langle v, \tau\rangle$, $s' = \langle v', \tau'\rangle$, represent the current state and the next state it transited to. $v, v' \in \mathcal{V}$, and v' is the succeeding intersection of v. Γ ($\tau, \tau' \in \Gamma$) is the set of time budgets (or remaining time budgets) with respects to the deadline, and we have τ-$t_{v,v'} = \tau'$, where $t_{v,v'}$ is the sampled travel time on road link $l_{v,v'}$ ($l_{v,v'} \in \mathcal{L}$). τ' is the remaining time budget at v'. We would like to note that the time budget at the origin o is determined by user. For the arriving on time based SSP problem, we have to consider both road intersection and time budget in each state (i.e., state space is $|\mathcal{V}| \times |\Gamma|$).[1]
- A is the finite set of actions, and $a \in A$, i.e., driving directions[2].
- $P_{s,a}^{s'} = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the distribution of travel time cost by action a such that the vehicle will move from intersection v with time budget τ at step t to intersection v' with remaining time budget τ' at step $t+1$.
- $\gamma \in [0,1]$ is the discount factor.
- $R(s')$ is the immediate reward received after transiting to intersection v' with time budget τ' from intersection v with time budget τ. The reward depends on whether the vehicle arrives on time or not. The details are defined in Section II-C.

Consequently, the generalized update of Q-values within an episode is formulated as follows:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha[R(s') + \gamma \max_a Q_t(s',a) - Q_t(s,a)], \qquad (2)$$

$$s \leftarrow s', \qquad (3)$$

where $Q_t(s,a)$ is the value of state-action pair at the t-th step, and $\alpha \in (0,1]$ is the learning rate. To balance *exploitation* and *exploration*, during train, we adopt the classic ϵ-greedy strategy to select actions. We note that our Q-learning approach is flexible to consider other practical factor(s) in the states, e.g., weather condition, as long as we integrate it into $s = \langle v, \tau\rangle$. Moreover, time dependent routing service by our Q-learning approach is applicable if $P_{s,a}^{s'}$ is obtained in a time

[1]This is precisely why it is challenging to deal with the continuous time budget, which will be addressed in the next section.

[2]Suppose at the intersection of a crossroad, the direction can be going straight, turning left, turning right, or turning back.

dependent manner, which is easy to achieve due to the data-driven nature of our approach. Since the basic Q-learning can only cope with small state space, we only consider small graph of map, and finite time budget and travel time in this section. To achieve the latter, intuitively, we may apply discretization by rounding up the time budget and travel time into integers.

C. Q-Value Representation and Path Planning

The converged Q-values in standard Q-learning represent the expected summation of discounted future reward, i.e., $Q^*(s,a) = E\left(\sum_{t=0}^{\infty} \gamma^t R_t\right)$, where $\gamma \in (0,1]$ is the discount factor, and R_t is the immediate reward at the t-th step. We define the reward and discount factor in a way that the converged Q-value represents the probability of arriving on time. Specifically, we describe the immediate reward as $R(s') \in \{0,1\}$, and $R(s') = 1$ 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 be 1. In this case, after an o-d pair is determined, we update the Q-value using Eq. (2) for each intersection, which starts with v (i.e., $v = o$) in each episode. To transit to next intersection v', we employ the action selection policy, i.e., Softmax strategy [39]. 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'. We repeat these steps until all Q-values converge.

The converged Q-values can be shown to be the probabilities of arriving on time in the following. From Eq. (2), the Q-value converges when $Q_{t+1}(s,a) = Q_t(s,a)$, $\forall s$ and $\forall a$. After the convergence, we have the Q-value defined as $Q^*(s,a)$ given by $Q^*(s,a) = Q^*(s,a) + \alpha[R(s') + \gamma \max_a Q^*(s',a) - Q^*(s,a)]$. After a simple combination and elimination, we have

$$Q^*(s,a) = R(s') + \gamma \max_a Q^*(s',a). \tag{4}$$

As this equation will be executed for a large number of times given different samples of rewards, in essence, it is the averaged/expected reward. Thus, Eq. (4) can be written as $Q^*(s,a) = E\left(R(s')\right) + \gamma \max_a Q^*(s',a)$, which is exactly the probability of arriving on time.

After $Q^*(s,a)$ is reached, we can obtain an optimal path by first stopping the Softmax strategy and then determining the best action a^* at the specified intersection with a certain deadline as follows:

$$a^*(s) = \arg\max_a Q^*(s,a). \tag{5}$$

We first compute $a^*(\langle o, \tau \rangle)$ to determine the optimal driving direction at origin, i.e., the current state, with a user-defined time budget with respects to the deadline. The direction will determine the next intersection, i.e., next state. Then, we input the second intersection with the remaining time budget into Eq. (5), to find the next intersection. The same steps are repeated until the destination is reached. Thus, the optimal path can be found in an online manner.

D. Convergence Proof

Canonical Q-learning algorithms are guaranteed to converge, but with an explicit constraint that the discount factor γ should be strictly less than 1. However, in our Q-learning algorithm for the stochastic shortest path problem, we need to set $\gamma = 1$, which enables Q-value to represent the probability of arriving on time. Therefore, we prove the convergence of the algorithm with $\gamma = 1$, by answering two questions. Firstly, whether the simple iterative step as described in Eq. (2) converges? Secondly, where are they converging to? Also, we are targeting at the following converged Q-values for all the state-action pairs:

$$Q^*(s,a) = E\left(R(s')\right) + \gamma \sum P_{s,a}^{s''} \max_a Q^*(s'',a), \tag{6}$$

where we use $s'' = \langle v'', \tau'' \rangle$ to avoid later symbol confusions. Subtracting Eq. (2) with Eq. (6) from both sides with simple manoeuvres, we can get:

$$\begin{aligned} &Q_{t+1}(s,a) - Q^*(s,a) = (1-\alpha)(Q_t(s,a) - Q^*(s,a)) \\ &+\alpha\left(R(s') - E\left(R(s')\right)\right) + \alpha\gamma(\max_a Q_t(s',a) - \\ &\sum P_{s,a}^{s''} \max_a Q^*(s'',a)). \end{aligned} \tag{7}$$

Since $\sum P_{s,a}^{s''} = 1$, we can transform Eq. (7) into:

$$\begin{aligned} &Q_{t+1}(s,a) - Q^*(s,a) = (1-\alpha)(Q_t(s,a) - Q^*(s,a)) + \\ &\alpha(R(s') - E(R(s'))) + \alpha\gamma(\sum P_{s,a}^{s''} \max_a Q_k(s',a) - \\ &\sum P_{s,a}^{s''} \max_a Q^*(s'',a)). \end{aligned} \tag{8}$$

Taking the absolute value from both sides, and performing simple deductions, we can arrive at:

$$\begin{aligned} &|Q_{t+1}(s,a) - Q^*(s,a)| \leq (1-\alpha)|(Q_t(s,a) - Q^*(s,a))| \\ &+\alpha|(R(s') - E(R(s')))| + \alpha\gamma(\sum P_{s,a}^{s''} |\max_a Q_t(s',a) \\ &- \max_a Q^*(s'',a)|). \end{aligned} \tag{9}$$

Since the second term of Eq. (9) will filter out in the long run. This is from the fact that they are essentially the same component when we are sampling an infinite number of times. Eq. (9) can be further simplified to

$$\begin{aligned} &|Q_{t+1}(s,a) - Q^*(s,a)| \leq (1-\alpha)|(Q_t(s,a) - Q^*(s,a))| \\ &+\alpha\gamma(\sum P_{s,a}^{s''} |\max_a Q_t(s',a) - \max_a Q^*(s'',a)|) \leq (1-\alpha) \\ &|(Q_t(s,a) - Q^*(s,a))| + \alpha\gamma(\sum P_{s,a}^{s''} \max_a |Q_t(s',a) \\ &- Q^*(s'',a)|). \end{aligned} \tag{10}$$

We define $\Delta_t = \max_{s,a}(Q_t(s,a) - Q^*(s,a))$, Eq. (10) can be represented as follows:

$$\begin{aligned} &|Q_{t+1}(s,a) - Q^*(s,a)| \leq (1-\alpha)\Delta_t + \alpha\gamma\left(\sum P_{s,a}^{s''} \Delta_t\right) \\ &\leq (1 - \alpha + \alpha\gamma)\Delta_t. \end{aligned} \tag{11}$$

Since Eq. (11) holds for all the state-action pairs, which indicates that:

$$\Delta_{t+1} \leq (1 - \alpha + \alpha\gamma)\Delta_t. \tag{12}$$

In our vehicle routing, we set $\gamma = 1$, which derives that:

$$0 \leq \Delta_{t+1} \leq \Delta_t. \tag{13}$$

Consequently, we can conclude that Δ_t will converge as t goes to $+\infty$, but cannot simply say that Δ_t converges to zero. We assume that Δ_t converges to a constant $C > 0$ as t approaches $+\infty$, which satisfies both Eq. (2) and Eq. (6) (i.e., the Bellman equation), and we have $\max_{s,a} |Q_k(s,a) - Q^*(s,a)| = C$. However, this condition cannot hold as t goes to infinity. Because when t goes to infinity, it means that the remaining time budget is definitely below zero, and in that case, we should have $Q_t(s,a) = Q^*(s,a) = 0$, which has been initialized. Thus, the constant C must be equal to 0. Another perspective of the convergence with $\gamma = 1$ is the absorbing states in the constructed MDP. Since we have added the temporal constraint into this MDP, and whenever the travel time violates the predefined deadline, the process stops. In particular, the agent reaches an absorbing state. This special case assures that the agent will never accumulate infinitely large reward even though it may execute the algorithm infinitely.

III. Q-LEARNING FOR CONTINUOUS DEADLINES

The basic model of our Q-learning approach previously introduced will become prohibitively time-consuming in the case with continuous deadlines (i.e., continuous time budgets) and large road networks due to the huge size of the state space. Therefore, we introduce a dynamic neural network to approximate the value function, which maps the state-related features to the maximal Q value to efficiently handle continuous deadlines and large road networks. To this end, we firstly present the overall update scheme and skeleton of the approximator in Section III-A, which is a fully connected neural network. Then, we flesh the approximator with the requested features, labels, and procedures for parameter optimization in Section III-B.

A. Dynamic Neural Network for Value Function

In Q-learning, a value function is used to calculate the optimal Q-values [39]. Particularly, the value function under policy π is defined as follows:

$$V(s)^{\pi} = \max_{a} Q^{\pi}(s,a), \tag{14}$$

where $Q^{\pi}(s,a)$ is the expected reward under policy π, which is expressed as $\sum P_{s,\alpha}^{s'}[V^{\pi}(s') + R(s')]$. In view of this, the value function with any time budget can be learned from the iteration for each intersection, formally,

$$V_{k+1}(s) = \max_{a} \left\{ \sum P_{s,a}^{s'}[V_k(s') + R(s')] \right\}, \tag{15}$$

where $0 \leq V(s') \leq 1$, $V(s') = 0$ if $v' = d$, and k is the iteration number. $V(\cdot)$ is used to compute an optimal Q-value.

To approximate $V_k(\cdot)$, we adopt a two-layer dynamic neural network[3] $f_k(\cdot)$ to learn $V_k(\cdot)$ in each iteration, which represents the probability of arriving on time from a given intersection with a given time budget. Specifically, $f_k(\cdot)$ is expressed as follows:

$$f_k(\vec{\mathbf{z}}) = g_2(g_1(\vec{\mathbf{z}} \cdot \vec{\mathbf{w}}_1 + u_1) \cdot \vec{\mathbf{w}}_2 + u_2), \tag{16}$$

where $\vec{\mathbf{z}}$ is the feature input, $\vec{\mathbf{w}}_1$, u_1, $\vec{\mathbf{w}}_2$, and u_2 are parameters of the two-layer neural network, $g_1(\cdot)$ and $g_2(\cdot)$ are activation functions. Thus, we can compute $V_k(s')$ on the right-hand side of Eq. (15) through the learned $f_k(\vec{\mathbf{z}})$. $V_{k+1}(s)$ on the left-hand side can then be updated accordingly. It will be adopted as new labels of the neural network to learn $f_{k+1}(\vec{\mathbf{z}})$ in the next iteration. The steps are repeated until the value functions converge. Afterwards, $f_k(\cdot)$ will act as $V_k(\cdot)$.

B. Feature Selection and Algorithm Training

$f(\vec{\mathbf{z}})$ is used to compute the probability of arriving on time with respect to the deadline. First, the remaining time budget τ is a suitable feature in $\vec{\mathbf{z}}$. Besides, from Eq. (1), the probability of arriving on time is also associated with the mean and standard deviation of travel time. Hence, we calculate the K-shortest paths between the current location and destination. Then, we adopt the mean $\vec{\mu}$ and standard deviation $\vec{\sigma}$ of the travel time on the paths as additional features. They 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 $\vec{\mathbf{z}} = [\tau, \vec{\mu}, \vec{\sigma}]^{\top}$, where the lengths of $\vec{\mu}$ and $\vec{\sigma}$ are K. Since $f(\vec{\mathbf{z}})$ will be applied for continuous time budgets, τ in training samples should cover extensive values. Therefore, we randomly select N time budgets, 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. Thus, the entire feature space of an individual intersection v is expressed as follows:

$$\vec{\mathbf{Z}}_s = [\tau_1, \vec{\mu}, \vec{\sigma}; \cdots; \tau_N, \vec{\mu}, \vec{\sigma}]^{\top}. \tag{17}$$

Note that the features of all intersections can be obtained similarly. However, to reduce the computation time of training, we may adopt only some intersections[4] in the road network to train the learning function.

Traditionally, value function $V_0(\langle v, \tau \rangle)$ is initialized as 0, which may slow down the convergence speed [40]. Therefore, we propose a warm start method which approximates the probability of arriving on time for vehicles at intersection v with time budget τ as follows:

$$V_0(\langle v, \tau \rangle) = 1/(1 + e^{\zeta(\tau - T_e)}), \tag{18}$$

where ζ is the coefficient. Note that the warm start is also applicable to our basic model of the Q-learning approach for discrete deadlines as presented in Section II.

We summarize the dynamic learning process of value functions in Algorithm 1. Lines 1-2 initialize parameters and prepare feature values. In Lines 3-29, we dynamically learn the value functions for each intersection using the neural network. In particular, in Lines 5-14, for each intersection, we use the

[3]Other traditional 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.

[4] An intuitive way is to uniformly select the intersections according to the node index in the adjacency matrix of the map graph.

ALGORITHM 1: Value Function Learning

input : $G = (\mathcal{V}, \mathcal{E})$, road network; $\Omega_{v,v'}$, set of travel time data on $l_{v,v'}$; o-d pair;
N, size of travel time data on each road link; $\mathbb{N}$, the configured neural network;
ϵ_1, ϵ_2, ϵ_3, ϵ_{best} the average convergence error, the total convergence error, convergence error threshold (a small positive number), the current best convergence error;
k, k_{best}, the iteration number and the best iteration number; Φ, threshold for additional iterations.

1 Initialize $V_0(s)$ by Eq. (18); $\epsilon_1 = \epsilon_{best} = 1$; $\epsilon_2 = 0$; $k = k_{best} = 0$; $N = 1000$; $\zeta = 0.025$; $\Phi = 10$.
2 Import feature data $\vec{\mathbf{Z}}_s$ for each s by Eq. (17);
3 **while** $k - k_{best} < \Phi$ **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 Compute $V_k(s')$ in Eq. (15) by the learned function $f_k(\vec{\mathbf{z}})$ (if $k \geq 1$), or by initial values $V_0(s')$ (if $k = 0$);
9 **end**
10 **else**
11 Compute the reward $R(s')$ in Eq. (15);
12 **end**
13 **end**
14 Update $V_{k+1}(s)$ in Eq. (15);
15 **if** $k \geq 1$ **then**
16 Update $\epsilon_2 = \epsilon_2 + |V_{k+1}(s) - V_k(s)|$;
17 **end**
18 **end**
19 **if** $k \geq 1$ **then**
20 Update $\epsilon_1 = \epsilon_2 / |\mathcal{V}|$;
21 **if** $\epsilon_{best} - \epsilon_1 >= \epsilon_3$ **then**
22 $\epsilon_{best} = \epsilon_1$;
23 $k_{best} = k$;
24 **end**
25 Reset $\epsilon_2 = 0$;
26 **end**
27 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 $\mathbb{N}$;
28 Update iteration number: $k = k + 1$;
29 **end**

output: Converged $V^*(s)$ for the iteration of k_{best}.

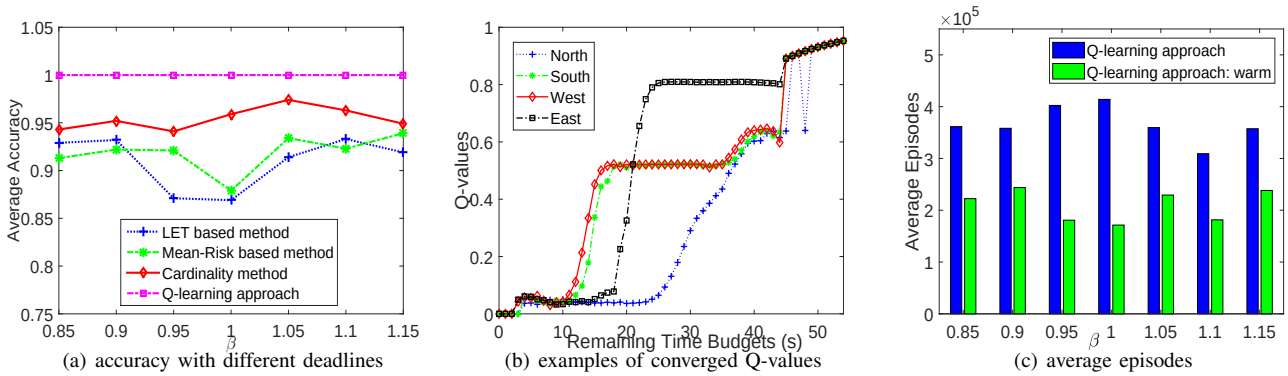
trained neural network from the preceding iteration to compute the value functions of the succeeding intersections, which are used to update the value function of the current intersection. In Lines 15-17, we accumulate the convergence errors of all intersections for iteration k. In Lines 19-26, we update the average convergence error for the learned value function, and compare it with the current best convergence error. If their difference is larger than a threshold, then the current best convergence error and the corresponding iteration number will be updated. Together with the condition in Line 3, the current best convergence error and the best iteration number will judge the convergence and decide the loop termination. In Lines 27-28, we train the neural network using feature values and updated labels, and then update the iteration number. Finally, we output the learned value functions that will be used to find an optimal path.

Remarks: Other than the basic Q-learning and approximate Q-learning, many other RL variants, such as DQN [41], DDPG [42] and A3C [43], can also be adopted to maximize the probability of arriving on time. We only select the basic RL model because the focus of this paper is to investigate whether RL can be applied to solve the arriving on time problem and outperform the conventional methods, rather than compare the performance of different RL variants. On the other hand, we can see from Section IV and V that the basic RL model already significantly outperforms the conventional methods. However, it is still worth to explore more advanced RL variants in future.

IV. Experimentation for Artificial Network and Discrete Deadlines

In this section, we conduct experiments to evaluate the proposed Q-learning approach for discrete deadlines. To this end, we first introduce the testing bed and baselines, where we also define the ground-truth path. Then, we analyze the performance of our approach against the baselines. All experiments are performed on a typical PC with Intel Core i7-3540M processor and 16GB RAM.

We implement the basic Q-learning approach introduced in Section II on an artificial road network, which is a grid network of 20×20 intersections. We first use $m_1 = 15$ as the mean and $\sigma_1 = 3$ as the standard deviation to randomly generate the mean travel time m_2 for each individual link. We then adopt m_2 and $\sigma_2 = 0.3m_2$ to generate 200 travel time samples for each corresponding road link to represent $P_{s,a}^{s'}$. All travel time is rounded up into positive integers, and the unit is assumed to be in second. At each grid, there are four traveling directions, i.e., north, south, west and east. We compare the proposed Q-learning approach 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 [15], and (3) *Cardinality method* approximates and computes a path of the probability tail model by formulating it as an MILP problem [24], the computation of which is speed up by the partial Lagrigian multiplier method [4]. To test the performance of our approach, we randomly select 100 o-d pairs and use $\beta = 0.85, 0.90, \ldots, 1.10, 1.15$ to represent different levels of time budgets (as defined in Section III-B). For each o-d pair and a specified time budget τ (obtained based on β), we consider a ground-truth path, which is determined as follows: If there are 200 travel time samples on each link, then we assume that there are 200 times of complete traveling on each path given an o-d pair. Thus, the ground-truth path has the maximal ratio of not being late regarding the given o-d pair and deadline. Intuitively, the ground-truth path can be found through enumerating all paths between the o-d pair and then comparing travel time on those paths against the time budgets. In light of this, we calculate the *accuracy* of hitting the ground truth path for all the methods. Particularly, the *accuracy* is defined as follows: Suppose that we need to find an optimal path for each query $q_i \in (< o_1, d_1, \tau_1 >, \cdots, < o_i, d_i, \tau_i >, \cdots, < o_{\mathcal{N}}, d_{\mathcal{N}}, \tau_{\mathcal{N}} >)$. If the route returned by a method is exactly the same with the ground-truth path regarding the same $< o_i, d_i, \tau_i >$, then



(a) accuracy with different deadlines (b) examples of converged Q-values (c) average episodes

Fig. 1: Results on Artificial Network with Discrete Deadlines. [**Best Viewed in Color**]

we call it a *success*. Meanwhile, we call it a *failure* if one or more road links are different from that of the ground truth path. Thus, the *accuracy* for this method is defined as the ratio of the quantity of *success* over $\mathcal{N}$. We would like to note that, although rare, multiple ground truth paths may simultaneously exist if they share the same highest ratio of not being late. In this case, we count it a *success* as long as the returned route belongs to them. Accordingly, we plot the accuracy of finding the ground-truth path for all methods in Fig. 1(a).

From Fig. 1(a) we can observe that the proposed Q-learning approach and cardinality method always achieve higher average accuracy for all selected deadlines, which are above 95%. The LET based method and Mean-Risk based method obtain the average accuracy of around 88%. The advantage of the former two methods comes from the fact that they incorporate the event of arriving at the destination before deadline in the reward and objective function, respectively. This superiority can also be verified 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). From Fig. 1(b), the converged Q-values are between zero and one, which denote the probabilities of arriving on time from the current location by taking the corresponding action. Meanwhile, the optimal action usually changes with the time budgets, e.g., traveling west is optimal as the remaining time budget is between 10 and 20, and traveling east is optimal as the time budget becomes larger than 20. The Q-value for traveling south is not higher than that of traveling west and east although the current location north to the destination, this happens because the cost of traveling south is always large according to the generated traffic data. Therefore, traveling south yields a lower chance of arriving on time. We also observe that the Q-values for all actions are zero as the remaining time budget is less than 4, which is too tight. Consequently, we conclude that an optimal path depends strongly on the time budget even though an origin is fixed. This factor is not incorporated in the LET based method or Mean-Risk based method. Therefore, there are fluctuations in their average accuracy as the remaining time budget is varied. In comparison with the cardinality method, the Q-learning approach always achieves the average accuracy of 100% for all selected time budgets. This firmly justifies that the Q-value is able to represent the probability of arriving on time. By contrast, the cardinality method adopts ℓ_1-norm to approximately minimize the frequency of not being later than the deadline. This inherently minimizes the summation of total delays. Consequently, the Q-learning approach achieves the highest accuracy.

Besides, we also apply the warm start mechanism proposed in section III-B to the Q-learning approach, and the result is shown in Fig. 1(c). The proposed Q-learning approach, which initializes the Q-value as zero, usually needs around 400,000 episodes before convergence. On contrary, with the warm start in Eq. (18), it needs around 180,000 episodes. The speed up of the convergence is 50%-60%.

V. Experimentation for Real Networks and Continuous Deadlines

In this section, we verify the proposed Q-learning approach for the arriving on time based SSP problem with continuous deadlines. To this end, we first introduce the new testing beds and settings. Then we evaluate our approach regarding accuracy, computation time, time dependent performance and average travel time, respectively.

We perform experiments on three large road networks, which are extracted from the city maps of Singapore, Munich, and Beijing, as shown in Fig. 2. To conduct the experiments, we adopt the same device, definitions of accuracy and ground truth path, and deadline parameter β, with the ones in Section IV, respectively. The scales of the three road networks are summarized in the first two rows of Table I. We randomly select 200 o-d pairs on each network, and the average minimal number of road links between each origin and destination is shown in the third row of Table I. Particularly, the *average minimal number of road links* is defined as follows: Suppose that the lengths of all road links are 1, then the shortest path between an o-d pair is also the one with minimal number of road links. Thus, for each o-d pair, there would be a path with minimal number of road links. Since we select multiple o-d pairs on each road network, then the average minimal number of road links is referred to the average number of road links over those o-d pairs. The average minimal number of road links roughly reflects the quantity of decision making a vehicle needs to perform before it reaches the destination. Note that, as defined previously, we call it a *failure* even if only one road link is different from that of the ground truth path.

(a) Singapore (b) Munich (c) Beijing

Fig. 2: The Three Real Road Networks.

TABLE I: Settings of the Three Real Road Networks

	Singapore	**Munich**	**Beijing**
#Nodes	6,476	51,517	129,607
#Links	10,225	115,651	294,868
Mean.Mini. #Links Btwn. O-D	86	221	358

Meanwhile, we prepare 1,000 travel time samples for each road link: (1) 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 the length[5]; (2) On Munich network, we use actual length of each road link to divide the collected real travel speed of vehicles[6] (i.e., from July-2014 to March-2015); (3) On Beijing network, we directly use the travel time data, which are collected from real travel trajectories of taxi (i.e., from September-2013 to October-2013) [45]. Thus, we assume that there are 1,000 complete traveling for each o-d pair. Then, our Q-learning approach adopts Algorithm 1 to dynamically learn the value function for each specified destination. Besides, we randomly select 100 destinations on the same network and concatenate their feature values to the same matrix (i.e., $\vec{\mathbf{Z}}_s$ in Eq. (17)). Then, we run Algorithm 1 only once for each network. We refer to this as the "Q-learning approach: generalized". Note that the Q-learning approach needs to learn one value function for each destination, no matter where the origin is. Nevertheless, the generalized Q-learning approach only learns one value function for the whole road network regardless of the origin and destination. We do not explicitly explore the dependence or correlation of travel time on different links, as it will be inherently included in the data if there is any, and the travel time data is the direct input to our approaches.

A. Accuracy

We test the accuracy of the five methods and plot the results in Figs. 3(a), (b) and (c). The accuracy for our proposed Q-learning approach and the generalized Q-learning approach approximately ranges from 90% to 97.5%, the average of which is about 94%, higher than that of all other methods for each deadline. Again, the accuracy of the cardinality method is higher than that of the LET based method and Mean-Risk based method. The proposed Q-learning approach is slightly better than the generalized Q-learning approach due to the fact that the Q-learning approach separately runs Algorithm 1 for each selected destination to learn the value function and then use it to find a path for the same selected destination. By contrast, the generalized Q-learning approach may only run Algorithm 1 once and adopt the learned value function to find a path for all selected destinations. However, the difference in the accuracy between the two approaches is small. Moreover, the accuracy of all methods becomes higher as the time budget increases on the three networks. Nevertheless, in comparison with the three baselines, the Q-learning approach and the generalized Q-learning approach are less affected because they always dynamically update the Q-values according to each specified time budget. Compared with the results in Fig. 1(a), the Q-learning approach achieves slightly lower accuracy. This is because, to handle continuous deadlines, we use a neural network to learn the value function in which errors due to learning always exist. Besides, the three networks are considerably larger than the grid network.

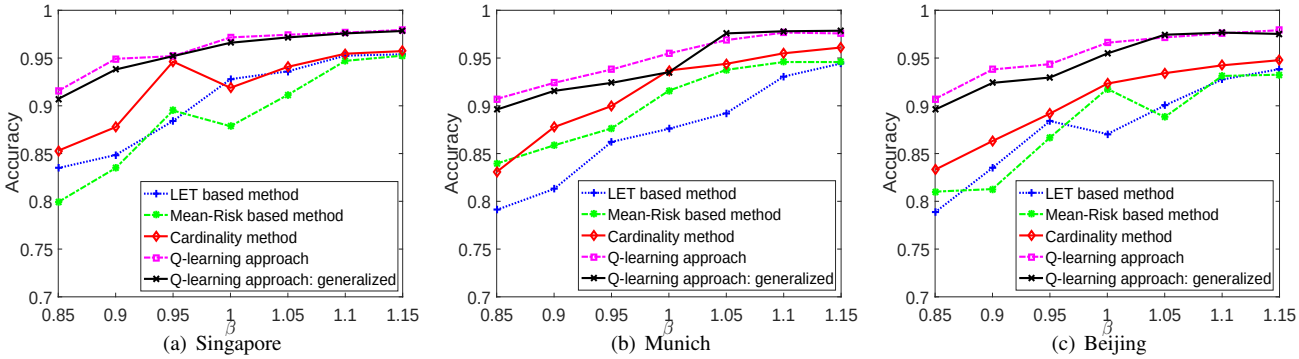
Besides, we record the overall accuracy of the five methods as well as the 2%-tolerance accuracy[7] of the generalized Q-learning approach, which is plotted in Fig. 4(a). We observe that the overall accuracy on the Munich network and Beijing network is slightly lower than that of the Singapore network for most of the methods. This is because the sizes of the Munich network and Beijing network are much larger, and the number of possible paths between an o-d pair is greater. Thus, it is more difficult to achieve the ground-truth path. In particular, the two Q-learning approaches will determine a road link to traverse at each intersection, and even if only one road link is not on the ground-truth path, the returned path is counted as a *failure*. In the case that there are at least 358 intersections on average (Table I) for each path finding on the Beijing network, the overall accuracy of around 94% for the generalized Q-learning approach is sufficiently high. Moreover, the 2%-tolerance accuracy of the generalized Q-learning approach is almost 100%.

Additionally, we test the accuracy against the quantity of

[5]All relevant information is obtained from OpenStreetMap [44].

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

[7]If the difference between the probability of arriving on time for the path returned by our approach and the probability for the ground-truth path is less than 2%, then the returned path is counted as a success in the calculation of the accuracy.



(a) Singapore (b) Munich (c) Beijing

Fig. 3: Results on Real Road Networks with Continuous Deadlines. [**Best Viewed in Color**]

travel time data on each road link. We take the Beijing network as an example since its size is the largest. We plot the results in Fig. 4(b). We observe that the Q-learning approach and the generalized Q-learning approach are almost not affected by the quantity of travel time data. They always achieve higher overall accuracy than that of other methods. By contrast, the LET based method, Mean-Risk based method, and cardinality method are likely to achieve high accuracy as the number of samples of travel time data increases, i.e., from 400 to 1000. However, their accuracy almost does not change as the quantity of data becomes larger.

B. Computation Time

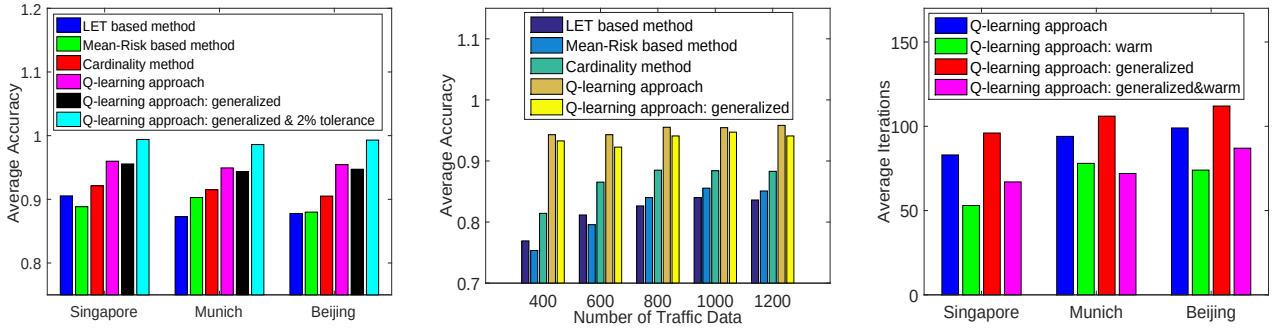
We also test the improvement brought by the warm start. The results are shown in Fig. 4(c). Clearly, the warm start saves around 20%-30% of iterations for the two proposed Q-learning approaches on the three networks. The generalized Q-learning approach needs larger number of iterations since the amount of training samples is much larger than that of the Q-learning approach. In comparison with the results in Fig. 1(c), the number of iterations in Fig. 4(c) is considerably small. The major reason is that in each episode of the Q-learning with discrete deadlines, the Q-values are updated using the local travel time data only for the intersections visited within that episode, while in the case of continuous deadlines, the value function is updated for all intersections by exploring all the travel time data on the road links in each iteration.

Moreover, in each iteration of the value function learning, we record the means of absolute errors for all training samples on the three road networks, which are shown in Fig. 5(a). From Fig. 5(a), we see that the average absolute errors become smaller as the dynamic neural network evolves and converges. This clearly justifies the efficacy of dynamic neural network to learn the more accurate value function. Meanwhile, the learned value function generally converges faster (i.e., around 60 iterations) on the Singapore network than that of the Munich network and Beijing network (i.e., around 75 and 85 iterations respectively).

Normally, the computation time in each iteration of the dynamic neural network is associated with the size of training samples. Previously, we adopted the training samples of all intersections in the road network to learn the value function. To improve the computation efficiency of learning the value function, we may only need samples from some intersections, while maintaining an acceptable level of accuracy. Therefore, we adopt the generalized Q-learning to find the optimal paths on the three networks and evaluate the average computation time in each iteration and accuracy with respect to different sizes of training samples. The results are plotted in Fig. 5(b). From Fig. 5(b), we observe that, as expected, the computation time and the accuracy decrease as the sizes of training samples reduce. On the Beijing network, the deterioration of accuracy is only around 1% while the reduction of computation time is around 460 seconds (i.e., almost half) as we use about 1/3 of the training samples[8]. Therefore, we may adopt 1/3 of the training samples to achieve satisfactory computation time and accuracy for the Beijing network. Similarly, we may determine proper sizes of training samples for the other two networks. We would like to note that, according to Fig. 5(b), the average computation time is around 400s per iteration for 1/3 on Beijing network, thus the total computation time is about 400sx80=32000s=8.8hrs. Although comparatively long, it is the time for training, which can be performed in advance using the historical data (note that the testing time is sufficiently short considering the result in Fig. 5(c), which is around 1.2 seconds on Beijing network).

We also record the average computation time of each routing for all the five methods. The results are shown in Fig. 5(c). We only count the time of path finding (i.e., also known as the testing time) for the generalized Q-learning since learning value function can be done off-line, although its computation time is comparatively longer. 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. From Fig. 5(c), the LET based method and Mean-Risk based method have the shortest average computation time, which slightly increases as the network size increases. However, the two methods do not take the deadline into account, and their accuracy of finding an optimal path is not high, especially on the Beijing network. The cardinality method needs around 187 seconds to compute a path

[8]In the adjacency matrix of the map graph, we select the nodes (or intersections) whose index is a multiple of 3. The similar principle also applies for ‘1/2,‘1/4,‘1/5.



(a) average accuracy (b) different data sizes (c) average iterations

Fig. 4: Comprehensive Results. [**Best Viewed in Color**]

on Beijing network, which is prohibitively long and caused by solving the MILP problem. Nevertheless, the cardinality method is able to achieve comparatively high accuracy. By contrast, the computation time of the generalized Q-learning approach is slightly longer than that of the LET based method and Mean-Risk based method. The computation time also increases as the network size increases. As it takes only 1.216 seconds to obtain an optimal path on the Beijing network, the generalized Q-learning approach is highly efficient.

With high accuracy and low computation time, the generalized Q-learning approach is appealing. Especially, it is able to avoid multiple runs of Algorithm 1 compared with the Q-learning approach.

C. Time Dependent Performance

Our Q-learning approach 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 (i.e., July and August) on the 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 the generalized Q-learning approach 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. Other settings are the same as the previous experiments, and the results are recorded in Table II.

TABLE II: 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

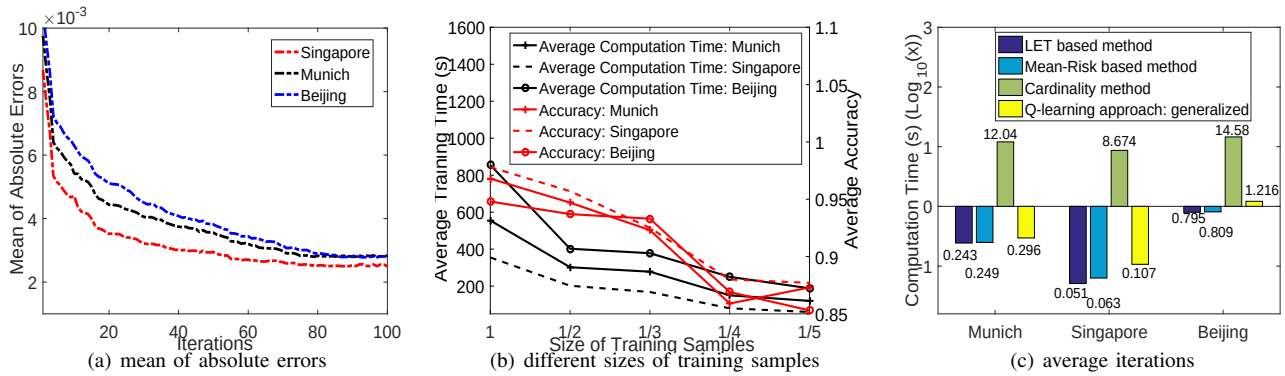
From Table II we can observe that, segment 1 usually achieves higher accuracy than that of the other seven segments. This is because the training data and testing data are the same for segment 1. Generally, as the segment index increases, the accuracy will decrease, but the deterioration is very slight, 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 (we do not consider minute, second and so on), the better the accuracy. This happens because the high time resolution usually captures the traffic characteristics better, i.e., peak hour pattern, and each segment for the **hour** shares this pattern. From Table II, we can conclude that the time dependent Q-learning approach helps to avoid frequent value function learning while achieving good accuracy, especially for the time unit **hour** (i.e., no need for re-training within at least two months from Table II). We also wish to note that, in most of the literature, 'time-dependency is characterized by a random variable, which represents the stochastic travel time for the corresponding road link and varies with time, such as [46]. Although the problem expression and the way to find a solution for our method are different from [46], they share the same goal: find an optimal path by considering the time-dependent travel time. On one hand, our method is data-driven, and we may not need to explicitly express the time dependency as it would be implicitly included inside the data if there is any. On the other hand, if our method can find an optimal path for 'Monday 7am-9am, it is supposed to be able to find optimal solutions for other time slots, days or weeks as well, which is solving the time dependent route optimization in a different way. In addition, we did not explicitly consider traffic congestion, as both saturated and under-saturated traffic conditions might be implicitly included in the real traffic data. Intuitively, the **hour** may refer to the former condition, while the **day** and **week** may refer to the latter condition.

D. Average Travel Time

TABLE III: Comparison of Average Travel Time (mins)

LET	**Mean-Risk**	**Cardinality**	**Q-Learning: generalized**
39.7	42.3	41.9	42.6

We further evaluate the performance regarding the average travel time. We take the Beijing network as an example,



(a) mean of absolute errors (b) different sizes of training samples (c) average iterations

Fig. 5: Comprehensive Results. [**Best Viewed in Color**]

and use the same setting as in Section V-A. Then the travel time is averaged over all O-D pairs and time budgets, and the results are recorded in Table III. From Table III we can observe that, there is no significant difference among them. If we zoom in, the LET achieves the lowest value, which is normal as it directly aims to minimize the expected travel time. The cardinality method ranks second because it approximately maximizes the probability of arriving on time, by minimizing the average delay regarding the time budget. The mean-risk method ranks third, as it considers average travel time, and the variance as well, in the objective function. Our approach produces relatively higher average travel time, which is normal as it does not directly consider minimizing the travel time. However, our approach is not far away from LET, because maximizing the chance of arriving on time also helps to reduce the travel time to some extent. However, on one hand, average travel time might not be the most important metric as it does not take into account the risk (or variance), e.g., the LET path cannot guarantee highest probability of arriving on time. On the other hand, there is a possibility in our approach that the vehicle goes on detour when the time budget is ample, because in this case the detour might not influence the probability of arriving on time, which is likely to increase the travel time. To avoid those unnecessary detours, an intuitive strategy might be that assigning negative rewards to the agent if it travels on unnecessary u-turns or circles.

VI. Conclusion and Future Work

In this paper, we have designed a Q-learning approach to solve the probability tail model based stochastic shortest path problem, which aims to minimize the probability of delay occurrence. In this approach, we have used the Q-value to represent the probability of reaching the destination before the deadline, which also has been theoretically proved. Experiments on both artificial network and real large road networks have shown the attractive results for finding an optimal path in terms of accuracy and computation time. Particularly, 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. However, we also acknowledge that shortcomings of the Q-learning approach also exist, e.g., it requires very big data set to learn the model and the results regarding a road network might not be generic to others.

In future, we plan to consider other reinforcement learning variants, such as deep Q-network [41] and DDPG [42] to solve the probability tail model based stochastic shortest path problem. Besides, we will also consider other factors, e.g., weather condition, number of lanes on the road link, traffic lights [47], and incentive schemes [48]. Additionally, we will come up strategies to avoid travelling on the unnecessary 'circle links' when the time budget is significantly ample.

VII. Acknowledgement

This work is supported by the National Natural Science Foundation of China (61803104, 61603169, 61772290), Fundamental Research Funds for the Central Universities (63192616), Fundamental Research Funds of Shandong University (62420079614084), and Singapore National Research Foundation (NRF-RSS2016-004).

References

[1] OICA, "Number of passenger cars and commercial vehicles in use worldwide from 2006 to 2015," Feb 2017.

[2] E. Demir, Y. Huang, S. Scholts, and T. Van Woensel, "A selected review on the negative externalities of the freight transportation: Modeling and pricing," *Transportation research part E: Logistics and transportation review*, vol. 77, pp. 95–114, 2015.

[3] D. Sever, L. Zhao, N. Dellaert, E. Demir, T. Van Woensel, and T. De Kok, "The dynamic shortest path problem with time-dependent stochastic disruptions," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 42–57, 2018.

[4] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Improving the efficiency of stochastic vehicle routing: A partial lagrange multiplier method," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3993–4005, 2016.

[5] H. Guo, Z. Cao, M. Seshadri, J. Zhang, D. Niyato, and U. Fastenrath, "Routing multiple vehicles cooperatively: Minimizing road network breakdown probability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 2, pp. 112–124, 2017.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[7] H. Yu, H. E. Tseng, and R. Langari, "A human-like game theory-based controller for automatic lane changing," *Transportation Research Part C: Emerging Technologies*, vol. 88, pp. 140–158, 2018.

[8] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, pp. 6348–6358, 2017.

[9] Z. Cao, H. Guo, J. Zhang, and U. Fastenrath, "Multiagent-based route guidance for increasing the chance of arrival on time," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3814–3820, 2016.

[10] Z. Cao, H. Guo, and J. Zhang, "A multiagent-based approach for vehicle routing by considering both arriving on time and total travel time," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 3, p. 25, 2018.

[11] H. Frank, "Shortest paths in probabilistic graphs," *Operations Research*, vol. 17, no. 4, pp. 583–599, 1969.

[12] E. D. Miller-Hooks and H. S. Mahmassani, "Least expected time paths in stochastic, time-varying transportation networks," *Transportation Science*, vol. 34, no. 2, pp. 198–215, 2000.

[13] S. T. Waller and A. K. Ziliaskopoulos, "On the online shortest path problem with limited arc cost dependencies," *Networks*, vol. 40, no. 4, pp. 216–227, 2002.

[14] M. G. Bell and C. Cassir, "Risk-averse user equilibrium traffic assignment: an application of game theory," *Transportation Research Part B: Methodological*, vol. 36, no. 8, pp. 671–681, 2002.

[15] E. Nikolova and N. E. Stier-Moses, "A mean-risk model for the traffic assignment problem with stochastic travel times," *Operations Research*, vol. 62, no. 2, pp. 366–382, 2014.

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

[17] S. Lim and D. Rus, "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)*, pp. 4814–4821, 2012.

[18] Y. Fan, R. Kalaba, and J. Moore II, "Arriving on time," *Journal of Optimization Theory and Applications*, vol. 127, no. 3, pp. 497–513, 2005.

[19] E. Nikolova, J. A. Kelner, M. Brand, and M. Mitzenmacher, "Stochastic shortest paths via quasi-convex maximization," in *European Symposium on Algorithms*, pp. 552–563, 2006.

[20] A. Christman and J. Cassamano, "Maximizing the probability of arriving on time," in *International Conference on Analytical and Stochastic Modeling Techniques and Applications*, pp. 142–157, 2013.

[21] Y. M. Nie and X. Wu, "Shortest path problem considering on-time arrival probability," *Transportation Research Part B: Methodological*, vol. 43, no. 6, pp. 597–613, 2009.

[22] S. Lim, H. Balakrishnan, D. Gifford, S. Madden, and D. Rus, "Stochastic motion planning and applications to traffic," *The International Journal of Robotics Research*, vol. 30, no. 6, pp. 699–712, 2011.

[23] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "A data-driven method for stochastic shortest path problem," in *Proceedings of the IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1045–1052, 2014.

[24] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: a cardinality minimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1688–1702, 2016.

[25] Z. Cao, Y. Wu, A. Rao, F. Klanner, S. Erschen, W. Chen, L. Zhang, and H. Guo, "An accurate solution to the cardinality-based punctuality problem," *IEEE Intelligent Transportation Systems Magazine*, 2018.

[26] A. Eiger, P. B. Mirchandani, and H. Soroush, "Path preferences and optimal paths in probabilistic networks," *Transportation Science*, vol. 19, no. 1, pp. 75–84, 1985.

[27] J. R. Current, C. S. Revelle, and J. L. Cohon, "The median shortest path problem: A multiobjective approach to analyze cost vs. accessibility in the design of transportation networks," *Transportation Science*, vol. 21, no. 3, pp. 188–197, 1987.

[28] J. L. Bander and C. C. White, "A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost," *Transportation Science*, vol. 36, no. 2, pp. 218–230, 2002.

[29] T. Rambha, S. D. Boyles, and S. T. Waller, "Adaptive transit routing in stochastic time-dependent networks," *Transportation Science*, vol. 50, no. 3, pp. 1043–1059, 2016.

[30] C. Mao and Z. Shen, "A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network," *Transportation Research Part C: Emerging Technologies*, vol. 93, pp. 179–197, 2018.

[31] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS)*, pp. 2692–2700, 2015.

[32] I. Bello and H. Pham, "Neural combinatorial optimization with reinforcement learning," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.

[34] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!," in *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[35] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2019.

[36] G. d. O. Ramosa, A. L. Bazzana, and B. C. da Silvaa, "Analysing the impact of travel information for minimising the regret of route choice," *Transportation Research Part C: Emerging Technologies*, vol. 88, 2018.

[37] Z. Jiang, W. Fan, W. Liu, B. Zhu, and J. Gu, "Reinforcement learning approach for coordinated passenger inflow control of urban rail transit in peak hours," *Transportation Research Part C: Emerging Technologies*, vol. 88, pp. 1–16, 2018.

[38] Z. Cao, H. Guo, F. Oliehoek, J. Zhang, and U. Fastenrath, "Maximizing the probability of arriving on time: A practical q-learning method," in *Proceedings of the 31th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4481–4487, 2017.

[39] R. S. Sutton and A. G. Barto, *Reinforcement learning: An Introduction (Second Edition)*. MIT press, 2018.

[40] S. Carden, "Convergence of a q-learning variant for continuous states and actions," *Journal of Artificial Intelligence Research*, pp. 705–731, 2014.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations*, 2016.

[43] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016.

[44] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

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

[46] Y. Pan, L. Sun, and M. Ge, "Finding reliable shortest path in stochastic time-dependent network," *Procedia-Social and Behavioral Sciences*, vol. 96, pp. 451–460, 2013.

[47] Z. Cao, S. Jiang, J. Zhang, and H. Guo, "A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 7, pp. 1958–1973, 2016.

[48] Z. Cao, Q. Li, H. W. Lim, and J. Zhang, "A multi-hop reputation announcement scheme for vanets," in *Proceedings of IEEE International Conference on Service Operations and Logistics, and Informatics*, pp. 238–243, 2014.

Zhiguang Cao received the Ph.D degree from Interdisciplinary Graduate School, Nanyang Technological University, Singapore, 2017. He received the B.Eng. degree in Automation from Guangdong University of Technology, Guangzhou, China, in 2009 and the M.Sc. degree in Signal Processing from Nanyang Technological University, Singapore, in 2012, respectively. He worked as a Research Fellow with Future Mobility Research Lab, and Energy Research Institute @ NTU (ERI@N), Singapore. He is currently a Research Assistant Professor with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore. His research interests include the application of AI to traffic and transportation management.

Hongliang Guo received his Bachelor of Engineering in Dynamic Engineering and a Master of Engineering in Dynamic Control at the Beijing Institute of Technology, China. He holds a PhD degree in Electrical and Computer Engineering from the Stevens Institute of Technology, USA. He later joined Almende in Rotterdam, the Netherlands as a postdoc research in 2011. His research interests include self-organizing systems and agent-based technologies. In 2013, he joined NTU as a research fellow. From September 2016, he becomes an Associate professor in University of Electronics Science and Technology of China.

Wen Song received the B.S. degree in automation and the M.S. degree in control science and engineering from Shandong University, China, in 2011 and 2014, respectively, and the Ph.D. degree in computer science from the Nanyang Technological University, Singapore, in 2018. He was a Research Fellow in the Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU). He is currently an Associate Research Fellow with the Institute of Marine Science and Technology, Shandong University, China. His current research interests include artificial intelligence, planning and scheduling, multi-agent systems, and operations research.

Kaizhou Gao received the B.Sc. and masters degrees from Liaocheng University and Yangzhou University, China, in 2005 and 2008, respectively, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2016. From 2008 to 2012, he was with the School of Computer, Liaocheng University, China. From 2012 to 2013, he was a Research Associate with the School of Electronic and Electrical Engineering, NTU, where he has been a Research Fellow from 2015 to 2018. He is currently an assistant professor with the Macau Institute of Systems Engineering, Macau University of Science and Technology. His research interests include intelligent computation, optimization, scheduling, and intelligent transportation. He has published over 60 refereed papers.

Zhenghua Chen received the B.Eng. degree in mechatronics engineering from University of Electronic Science and Technology of China (UESTC) in 2011, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He was a research fellow in Nanyang Technological University, Singapore. He is currently a scientist with the Institute for Infocomm Research (I2R), Singapore. His research interests include occupant sensing, indoor localization, and human activity recognition.

Le Zhang is currently a scientist with the Institute for Infocomm Research (I2R), Singapore. He was a researcher at the Advanced Digital Sciences Center (ADSC), the Singapore-based research center of the University of Illinois at Urbana-Champaign (UIUC). He received his Ph.D degree in school of EEE , Nanyang Technological University (NTU) in 2016. He also received his B.E. from University of Electronic Science and Technology Of China (UESTC) in 2011 and M.SC from NTU in 2012. His current research interests include machine learning, computer vision and pattern recognition.

Xuexi Zhang received the Ph.D degree from School of Automation, Guangdong University of Technology, China, 2009. He received the B.Eng. degree in Electric & Automations from Zhengzhou Engineering Institute, China, in 2000 and the M.Eng. degree in Control Theory and Engineering from Guangdongs University of Technology, China, in 2003, respectively. He is currently an Associate Professor with School of Automation, Guangdong University of Technology, China. His current research interests include machine learning and operations research.