Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 146070, 10 pages
http://dx.doi.org/10.1155/2015/146070

Research Article

A Cooperative Q-Learning Path Planning Algorithm for Origin-Destination Pairs in Urban Road Networks

Xiaoyong Zhang, Heng Li, Jun Peng, and Weirong Liu

School of Information Science and Engineering, Central South University, 22 South Shaoshan Road, Changsha 410075, China

Correspondence should be addressed to Jun Peng; pengj@csu.edu.cn

Received 25 May 2015; Accepted 21 September 2015

Academic Editor: Chronis Stamatiadis

As an important part of intelligent transportation systems, path planning algorithms have been extensively studied in the literature. Most of existing studies are focused on the global optimization of paths to find the optimal path between Origin-Destination (OD) pairs. However, in urban road networks, the optimal path may not be always available when some unknown emergent events occur on the path. Thus a more practical method is to calculate several suboptimal paths instead of finding only one optimal path. In this paper, a cooperative Q-learning path planning algorithm is proposed to seek a suboptimal multipath set for OD pairs in urban road networks. The road model is abstracted to the form that Q-learning can be applied firstly. Then the gray prediction algorithm is combined into Q-learning to find the suboptimal paths with reliable constraints. Simulation results are provided to show the effectiveness of the proposed algorithm.

1. Introduction

Recent years have seen a growing interest in the study of route-guidance system in intelligent transportation systems, due to its advantages in reducing traffic congestion and CO_2 emissions, minimizing travel time, and conserving energy [1]. More and more vehicle manufacturers have installed the route-guidance system into their products to assist the drivers' travel.

As an essential part of the route-guidance system, path planning is usually modeled as the shortest-path problem in graph theory [2–8]. When a vehicle departs from the origin and travels to its destination, the map it is involved in can be abstracted as a graph by treating streets as edges and intersections as nodes. The weight of an edge represents the average travel time over the street, which may dynamically change when traffic flows fluctuate. For the graph of a static network, the most efficient one-to-one node shortest path algorithm is Dijkstra's algorithm [2]. When the dynamic graph is considered, A^* algorithm might be a better choice to solve the Origin-Destination shortest path problem [3]. A^* algorithm estimates the minimum distance between the destination and a node to determine whether the node is on the optimal route.

However, even if the generated route is the shortest one, it may not be always available because of traffic emergencies such as sudden accidents. So it may be more practical to provide a number of candidate paths rather than just one optimal path. Lee revealed that finding multiple paths instead of one is a good way to avoid the path overload phenomenon [4]. This optimal path will even accelerate the deterioration of the road network when the overload phenomenon occurs.

Traditionally, the alternative paths could be calculated by two categories of algorithms in graph theory, namely, the k-shortest path algorithm proposed by Eppstein [5] and Jiménez and Marzal [6] and the totally disjoint path algorithms proposed by Dinic [7] and Torrieri [8]. These so-called alternative path planning methods typically find the optimal path using Dijkstra's algorithm first. Then the candidate path set can be generated by applying link weight increment methods. These algorithms seek for the next suboptimal path iteratively until the generated alternative path satisfies some given constraints.

However, the generated way of alternative paths of these algorithms unavoidably lengthens the response time, especially when the network is huge and the traffic load is crowded and time varying. These algorithms need to adjust the link weight of the generated optimal path and then recalculate

the suboptimal paths using Dijkstra's algorithm repeatedly, thus leading to heavy computation burden. In addition, these algorithms generally concern path planning of just one vehicle, while it is essential to simultaneously consider all vehicles' path in practical city road networks.

With the development of intelligent science, some researchers have focused on path planning using reinforcement learning in guidance systems. Reinforcement learning is a category of machine learning algorithms, in which a group of agents can decide how to behave according to their interaction with environment and achieve an optimal objective [9]. Recently, multiagent reinforcement learning has been proposed to find the best and shortest path between the origin and the destination. Some studies treat each intersection as one agent, which needs a large amount of information interaction between traffic intersections to find the optimal path [10] while more studies cast each intersection as the state and take each link as the action in the model, which could deal with the road networks on the whole [11, 12]. Thus our proposed Q-learning adopts the latter method, treating the intersections as states in the model.

With Q-learning, the computational complexity of path planning algorithm could be reduced significantly and the efficiency would be improved. While most existing Q-learning algorithms are designed to solve the optimal path planning for just one OD pair in the literature, the proposed Q-learning algorithm in this paper aims to seek multiple paths for different OD pairs simultaneously. By choosing the suboptimum Q-value of every intersection, it is convenient to provide some alternative paths rather than seeking every alternative route incrementally. This paper makes the following contributions in particular.

First, the multipath set is found for different OD pairs simultaneously using Q-learning. Compared with other multipath algorithms, the proposed algorithm significantly reduces the computational complexity.

Second, some reliability constraints are introduced to choose suboptimal paths in Q-learning. It would not be appropriate to increase the dimension of the multipath set without considering the overall reliability, which ensures that at least one alternative path is available at all times [13].

Third, the FNN prediction is combined with Q-learning. In order to improve the real-time capability, short-term traffic prediction is essential [14, 15]. This paper adopts the FNN prediction mechanism in the Q-learning scheme to predict the traffic condition, with which the reward of the action can be computed in advance.

Fourth, the multiagent cooperative mechanism is applied to path planning. The cooperative mechanism introduced in Q-learning coordinates the actions and strategies among agents with different OD pairs for long-time benefits.

In this paper, we propose a new multiagent reinforcement learning (MARL) algorithm using Q-learning with prediction for multipath planning for OD Pairs in the road navigation system. Compared with traditional multipath algorithms, it reduces computational complexity and improves the efficiency of vehicles' guidance with traffic prediction. The scheme could improve the overall performance of urban traffic networks and balance the traffic flow.

Figure 1: Eastern Town of Changsha.

The rest of the paper is organized as follows. Section 2 describes the model of road networks. The Q-learning based cooperative multiagent multipath planning algorithm for OD pairs is proposed in Section 3. The simulation results are shown and analyzed in Section 4. The conclusion is drawn in Section 5.

2. Model of Road Networks

2.1. Graph Abstraction of Road Networks. For urban areas, two important elements of traffic guidance are intersections and roads. During the process of modeling, the intersection can be seen as the node and the road can be seen as the edge connecting two nodes. The weight on the line stands for the traffic condition of the road, and the arrows mean the allowable direction of forward motion for vehicles. By this abstracting, a graph $G = (S, E)$ with a nonempty finite set of intersections (nodes) $S = \{s_1, s_1, \ldots, s_N\}$ and a set of roads $E \subseteq S \times S$ can be used to describe the road map. Once we have the model and the route algorithm, we can find the needed optimal route.

For instance, Eastern Town of Changsha in China could be taken as an example, whose map is shown in Figure 1. The abstract graph model of Figure 1 is showed in Figure 2. S_i stands for each intersection that is taken as one state in reinforcement learning. S_i has three or four directions to neighbor intersections, including the loop direction that returns to S_i. For example, if one vehicle at intersection S_1 drives west, it will return to S_1. The setting is convenient to model the complex road networks.

The weight of each direction will contain two elements: traffic condition (w) and the distance from the destination S_3 (r). These two elements will be illustrated in the next section.

2.2. Model Using Reinforcement Learning. To address the model more clearly, it is necessary to provide the background on reinforcement learning (RL). Reinforcement learning is a kind of multiagent intelligent algorithms, in which agents select the best actions to maximize the cumulative reward by interacting with the environment. The RL agent interacts with its environment over a sequence of discrete time steps to pick out the optimal actions. The agent in this paper is a processing

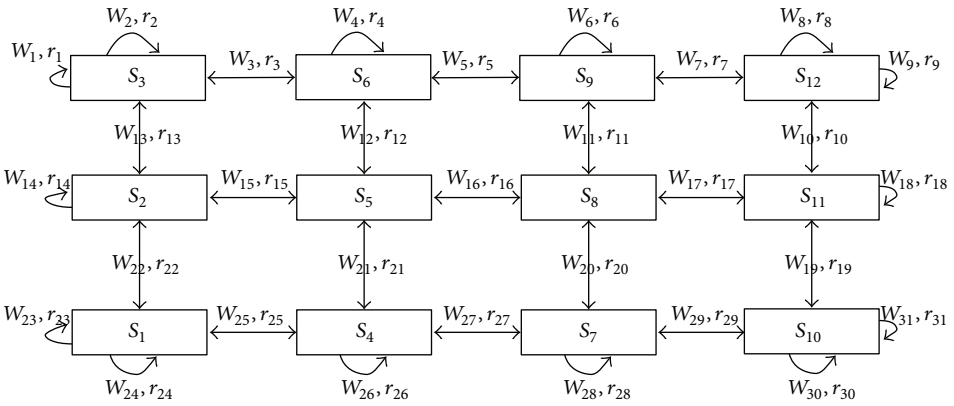

Figure 2: Road model of Figure 1.

center that deals with the path planning from one origin to one destination.

The underlying concept of RL is the finite Markov Decision Process (MDP), which is defined by the tuple $\langle S, A, \phi, \rho \rangle$, where S is a finite set of environment states, A is a finite set of agent actions, ϕ: $S \times A \times S \rightarrow [0, 1]$ is the state transition probability function, and ρ: $S \times A \times S \rightarrow R$ is the reward function. The MDP models an agent's action in an environment where it learns (through prior experiences and short-term rewards) the best control policy (a mapping of states to actions) that maximizes the expected discounted long-term reward. This mapping can be stochastic π: $S \times A \rightarrow [0, 1]$ or deterministic π: $S \times A \rightarrow 0 \parallel 1$.

For deterministic state transition models, the transition probability function ϕ reduces to ϕ: $S \times A \times S \rightarrow 0 \parallel 1$ and, as a result, the reward is completely determined by the current state and the action; that is, ρ: $S \times A \rightarrow R$. The state-action pair's value is called the Q-value and the function that determines the Q-value is called the Q-function. An agent can find the optimal control policy by approximating its Q-values using prior estimates iteratively, the short-term reward $r = \rho(s, a) \in R$, and discounted future reward. This model-free successive approximation technique is called Q-learning. One way to satisfy this criterion is adopting ε greedy approach where a random action is performed with probability ε and the current knowledge is exploited with probability $1 - \varepsilon$.

This paper denotes the link from intersection i to intersection j as a paired index of ij. And, accordingly, the reward of link ij is defined as r_{ij}; the mean travel time of link ij is defined as t_{ij}; the distance between the intersection i and the intersection j is defined as d_{ij}. To cast the path planning problem of the road network as a RL problem, we identify an individual agent i's states (S_i), available actions (A_{S_i}), and reward (r_{ij}).

First, the two weights of each direction should be illustrated. Traffic condition can be described by the mean travel time t_{ij} [13, 16–19]. The distance from the destination that points out the close degree of destination is needed by the idea of reinforcement learning.

Mean travel time t_{ij} could be obtained by probe vehicles such as taxis. Equipped with GPS sensors, probe vehicles can collect data on position, speed, and direction, store them, and send reports at regular intervals of time. By analyzing these data, the mean travel time of each link could be calculated. For instance, Hellinga and Fu advanced the method of probe based arterial link travel time estimation [16]. Tomio et al. used probe vehicle data to identify routes and predict travel times [17]. The distance d_{ij} stands for the Euclidean distance between the intersection i and the intersection j.

States. States of each interaction are form the basis for making choices. We model one intersection as one state S_i, which can easily indicate the location of vehicles.

Actions. Actions are the choices made by the agent. In one state, the vehicle will have four actions A_{S_i}: turning left, turning right, going straight, and turning round. For example, $A_{S_i} = \{\text{up}, \text{down}, \text{right}, \text{left}\}$.

Rewards. Rewards are the basis for evaluating choices. We model the link weight of each link as the rewards, which will contain two elements: the reward of mean travel time (r_{ij}^t) and the reward of the distance from the destination (r_{ij}^d).

For reinforcement learning, everything inside the agent should be completely known and controllable by the agent; everything outside is incompletely controllable but may be or may not be completely known. A policy is a stochastic rule by which the agent selects actions as a function of states. The agent's objective is to maximize the amount of rewards it receives over time. The return R_{ij} is the function of future rewards that the agent seeks to maximize:

$$R_{ij} = r_{i^1 j^1} + \beta r_{i^2 j^2} + \beta^2 r_{i^3 j^3} + \cdots = \sum_{k=0}^{N} \beta^k r_{i^{k+1} j^{k+1}}, \quad (1)$$

where β $(0 < \beta < 1)$ is called the discount rate. A whole path has N links.

3. Cooperative Multipath Planning for OD Pairs

In this section, we propose a cooperative multipath planning method for OD pairs. In the proposed method, the agent is a processing center that deals with the path planning from

one origin to one destination. In practical applications, there are typically many paths that are planned synchronously. By introducing the concept of multiagent systems, the path planning problem can be modeled to a form such that reinforcement learning is applicable. Then we introduce a multiagent reinforcement learning mechanism to optimize the Q-value of different paths.

3.1. Reward of Traffic Flow Using FNN Prediction. The travel time of each link ij is defined as t_{ij}. For given OD pairs (O_i, D_j), a set of binary variables are given to represent the selection of links on a path (i.e., a path solution). Thus, for a given path P, its travel time T could be calculated by [20]. Consider

$$T = \sum_{ij \in P} t_{ij}. \tag{2}$$

After getting the history data of travel time t_{ij} of each link ij, we can use some prediction algorithms to compute the future data for improving the real-time characteristics of the guidance system. In this paper, T-S FNN (fuzzy neural network) is introduced to predict the future travel time. T-S FNN is a highly adaptive fuzzy system that can automatically update the membership function of fuzzy subset [13, 21]. This FNN is defined by the following if-then rules. In the case of R^i, fuzzy reasoning is as follows:

$$\begin{aligned} R^i&: \text{If } t_1 \text{ is } A_1^i,\ t_2 \text{ is } A_2^i, \ldots,\ t_k \text{ is } A_k^i, \\ &\text{then } \widehat{t}^i = p_0^i + p_1^i t_1 + \cdots + p_k^i t_k, \end{aligned} \tag{3}$$

where A_j^i is the fuzzy set of the fuzzy system and p_j^i is the parameters of the fuzzy system, $(j = 1, 2, \ldots, k)$. $\widehat{t}^i$ is the predictive output based on the ith fuzzy rule. The input part is fuzzy, while the output part is deterministic, which is the linear combination of the input.

Suppose that the input of t_{ij} is $[t_1, t_2, \ldots, t_k]$, and then the membership degree of each input variable t_j can be computed by fuzzy rules:

$$\mu_{A_j^i} = \exp\left(-\frac{\left(t_j - c_j^i\right)^2}{b_j^i}\right) \qquad j = 1, 2, \ldots, k;\ i = 1, 2, \ldots, n, \tag{4}$$

where c_j^i and b_j^i are the center and width of the membership function, respectively, k is the number of the inputs, and n is the number of fuzzy subsets.

All membership degrees are computed by the fuzzy operators:

$$w^i = \mu_{A_1^i}(t_1) * \mu_{A_2^i}(t_2) * \cdots * \mu_{A_k^i}(t_k) \qquad i = 1, 2, \ldots, n. \tag{5}$$

The output of this fuzzy model is computed by the above results:

$$\widehat{t} = \frac{\sum_{i=1}^n w^i \left(p_0^i + p_1^i t_1 + \cdots + p_k^i t_k\right)}{\sum_{i=1}^n w^i}. \tag{6}$$

FNN is divided into four layers: input layer, fuzzy layer, fuzzy rules calculating layer, and output layer. The input layer is connected with the input vector t_k, so the number of nodes is equal to the dimensions of input vectors. Fuzzy Layer obtains fuzzy membership values μ using membership functions and fuzzy input values (4). Fuzzy rules' calculating layer gets w by the fuzzy multiply equation (5). The output layer uses (6) to calculate the fuzzy neural network outputs.

The parameters of FNN are updated by the following equations. The error e between the desired output and actual output is defined as follows:

$$e = \frac{1}{2}\left(\widehat{t} - t\right)^2, \tag{7}$$

where $\widehat{t}$ is the desired output, t is the actual output, and $\widehat{t}$ is the predictive mean travel time.

The parameters p_j^i of FNN are updated by

$$\begin{aligned} p_j^i(k) &= p_j^i(k-1) - \alpha \frac{\partial e}{\partial p_j^i} \\ \frac{\partial e}{\partial p_j^i} &= \frac{\left(\widehat{t} - t\right) w^i}{\sum_{i=1}^n w^i \cdot t_j}, \end{aligned} \tag{8}$$

where α is the learning rate, t_j is the input, and w^i is the weight computed by (5).

The center c_j^i and the width b_j^i of membership function are updated by (9) and (10), respectively. Consider

$$c_j^i(k) = c_j^i(k-1) - \beta \frac{\partial e}{\partial c_j^i}, \tag{9}$$

$$b_j^i(k) = b_j^i(k-1) - \beta \frac{\partial e}{\partial b_j^i}. \tag{10}$$

When one finds multiple paths using Q-learning, it is important to determine how every action is assessed. A link can be simply described as unblocked, normal, and busy. To simplify the learning process, the precise specific flow density is neglected because finding multipath is the final goal. Thus some links having a very small difference can be regarded as the same optimal choice.

This paper gives discrete weights in terms of the traffic condition and the distance from the destination to simplify the Q-learning reward's iterative calculations. For discretion of the mean travel time t_{ij}, we first gather the maximum travel time of current intersection as "−1"; then we get the difference value between the current maximum travel time and the minimum travel time. So the travel time can be graded into "0," "1," "2," and "3" as the reward r_{ij}^t. The distance from the destination (r_{ij}^d) can be graded into two levels: "1" and "0"; the nearest neighbor intersections are "1," while the others are "0."

So r_{ij} is deduced in the following equation, where τ $(0 \le \tau \le 1)$ is the scaling factor between r_{ij}^d and r_{ij}^t:

$$r_{ij} = \tau r_{ij}^t + (1 - \tau) r_{ij}^d. \tag{11}$$

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a direction a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'

$$Q_{n+1}^{i}(s,a) = (1-\alpha_n) Q_n^i(s,a) + \alpha_n^i \left[r + \gamma^i \max_{a' \in A} Q_n^i(s', a') \right]$$

Until that s is the terminal state.

Algorithm 1: The policy iteration algorithm.

3.2. Cooperative Multiagent Multipath Planning Algorithm. If each agent acts independently without cooperation, the Q-learning procedure at node i can be written as

$$Q_{n+1}^{i}(s,a) = \left(1-\alpha_n^i\right) Q_n^i(s,a) + \alpha_n^i \left[r + \gamma^i \max_{a' \in A} Q_n^i\left(s', a'\right) \right], \tag{12}$$

where $\alpha_n \in (0, 1]$ is the learning factor and $\gamma \in [0, 1)$ is the discount factor.

RL have been well developed for discrete-time systems to solve the optimal problem online by using adaptive learning techniques to determine the optimal value function.

An iterative solution technique is given by Algorithm 1.

An agent is a processing center that deals with the path planning from one origin to one destination. The above algorithm focuses on one single agent, while path planning agents with different destinations will observably impact on each other. Thus we propose a cooperative multiagent reinforcement learning (MARL) multipath path planning method, in which all Q-values of different path plans for every intersection are considered, and the maximum value is chosen to ensure that all path planning is optimized under the consideration of each other. It is worth mentioning that, in the proposed algorithm, the decision-making process is assumed ideal, and then the waiting time at the intersections is thus ignored.

In this approach, the Q-value estimated at each autonomous agent is updated based on the individual rewards as well as on information obtained from other agents in the neighborhood. "The neighborhood" here refers to a group of agents that own different destinations. Every agent exchanges the largest Q-value that is associated with its current state with every other agent in its neighborhood. The value iteration procedure at agent i for the state-action pair (s_i, a_i) can be summarized as

$$Q_{n+1}^{i}\left(s^i, a^i\right) = \left(1-\alpha_n^i\right) Q_n^i\left(s^i, a^i\right) + \alpha_n^i \left[r^i\left(s^i, a^i\right) + \gamma^i \sum_{j \in N^i} w(i,j) \cdot \max_{a^j \in A^j} Q_n^i\left(s^j, a^j\right) \right], \tag{13}$$

where $w(i, j)$ is the weight to reflect the effect of agent j to agent i and N_i refers to the set of neighboring agents of i. The simplest strategy for computing the weights $w(i, j)$ is to just consider the total number of agents in the neighborhood; that is, $w(i, j) = 1/|N_i|$, in which case $\sum_j w(i, j) = 1$. It is possible to adopt more complex strategies to take into account the different effects of the different neighbors. When the additional information obtained from agents was incorporated into the value iteration procedure, each agent can ensure that the agent's strategies are decided based on all its neighbors' actions.

3.3. Constraint Conditions of Multipath Set. By the policy iteration algorithm in Section 3.2, we can derive the Q-value table for multipath planning. By comparing the four Q-values of one intersection, we can easily find which action is the best. Then the optimal path is easily obtained.

Although the obtained optimal path is the fastest one, it may encounter traffic emergencies such as a sudden accident, resulting in unavailability of the optimal planning path for the running vehicle. So it is essential to provide several candidate paths rather than just one path, avoiding the deterioration of the road network environment when one guided vehicle is well popular.

In most cases, there may be several actions forming a best action set. Then we can find the multipath by choosing the best actions. However, when the road network is huge, it is difficult to find multipath because there exists only one optimal action for all intersections in most cases. So we should find the suboptimum action that satisfies the following constraints.

First, we introduce B_i as the average Q-value of one intersection. Then we compute the average difference between each $Q(s, a)$ and the average Q-value of one intersection. Furthermore, we can get τ value, the average difference for all states:

$$B_i = \frac{\sum_{a_j \in A_{S_i}} Q\left(S_i, a_j\right)}{4}, \qquad \tau = \partial * \frac{\sum_{S_i \in S} \sum_{a \in A_{S_i}} \left| B_i - Q\left(S_i, a\right) \right|}{4N}. \tag{14}$$

When vehicle arrives at one intersection, it has to make the choice about which way to go by computing the difference

Table 1: Q value table of $4 * 4$ road network with destination Intersection 11.

$Q(s, a)$	Value
$Q(1, 2)$	−25.2838
$Q(1, 5)$	−25.2838
$Q(2, 6)$	−21.8556
$Q(2, 3)$	−22.7127
$Q(10, 11)$	0
$Q(6, 2)$	−27.2838
$Q(6, 7)$	−16.4274
$Q(6, 10)$	−16.4274
$Q(7, 11)$	0
$Q(7, 3)$	−22.7127

between this $Q(s, a)$ and the average Q-value of this intersection:

$$\left|Q\left(S_i, a\right) - B_i\right| \leq \tau. \tag{15}$$

Once the Q-value table has been calculated, we can select ∂ ($\partial \in (0, 1)$) value and compute the corresponding τ value. So we must solve the problem of how to ensure the ∂ value. When ∂ is closer to 1, τ is larger. And there are more paths that could be taken as candidates. When ∂ is closer to 0, τ is smaller, resulting in fewer candidate paths. While more candidate paths sometimes are not stable, the reliability of a path set S should be taken into account.

The reliability of a path can be defined as the probability of not encountering an abnormal delay during a trip along the path, which can be estimated by the reliability of a series of links of the path. Under dynamic conditions, some or all candidate paths may fail together, resulting in a joint failure. In this situation, the reliability of the path set shown in (10) will be weakened. Thus, in the calculation of the candidate paths, it is important to reduce the chance of joint failure of candidate paths [18, 19]:

$$\Phi s = \sum_{i=1}^{M} \left[\prod_{j_{1i}=1}^{J_{1i}} r_{j_{1i}} \prod_{j_{2i}=1}^{J_{2i}} \left(1 - r_{j_{2i}}\right) \right], \tag{16}$$

where Φs stands for the reliability of the path set S, M is the number of disjoint subpaths in the subpath set S, j_{1i} is the jth link in a normal state on the ith disjoint subpath, $r_{j_{1i}}$ is the reliability of the jth link on the ith disjoint subpath, J_{1i} is the number of links in a normal state on the ith disjoint subpath, j_{2i} is the jth failed link on the ith disjoint subpath, $r_{j_{2i}}$ is the reliability of the jth link in a failed state on the ith disjoint subpath, and, finally, J_{2i} is the number of links in a failed state on the ith disjoint subpath.

Generally, the higher the reliability of the candidate path set, the less the chance that all candidate routes will be unacceptable during one trip. Given Φs value (such as 0.9), we can choose ∂ value. Then we find the stable multipath paths.

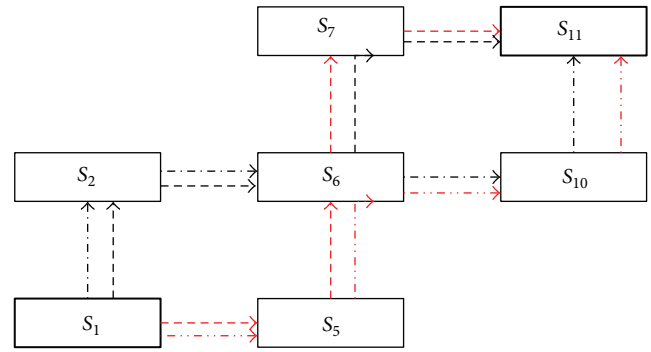

Figure 3: Multipath planning from Intersection 1 to Intersection 11.

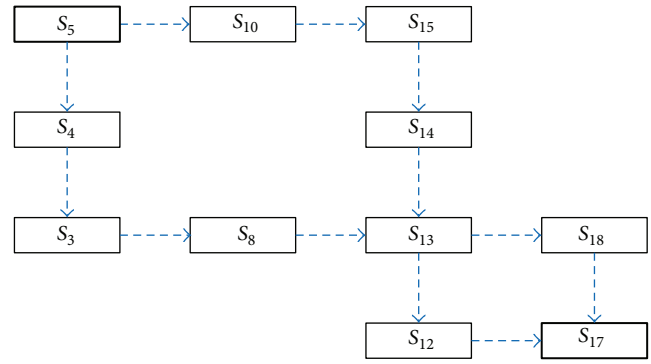

Figure 4: Multipath planning from Intersection 5 to Intersection 17.

4. Simulation Results and Analysis

To test the proposed method in different traffic environments, simulations have been conducted in randomly generated grid road networks. In the simulation scenarios, there are 6 to 25 nodes with 10 to 50 edges. The general network discussed in this paper is shown in Figure 1. In the graph, a node is represented by a box with the node's number shown in it. The start and destination nodes are represented by the rectangles with thicker outlines. The mean travel time and the distance from the destination of all links are set by a random matrix.

4.1. Path Planning for Single Agent. The scenario is shown in Figure 1. The single agent only computes the path towards one single destination. The main objective of Figure 3 and Table 1 is to calculate the shortest path from Intersection 1 to Intersection 11. This is drawn (shortest path) by the use of MATLAB software. Therefore four candidate paths exist. When the car arrives at Intersection 1, it can choose either Intersection 2 or Intersection 5. When the car arrives at Intersection 6, it can choose either Intersection 7 or Intersection 10. In this case, ∂ is equal to 0.1, and Φs is equal to 0.91 by computing.

Given the matrix of Q-value, we can quickly get the optimal multipath. Compared with the other multipath algorithm, our proposed algorithm only uses the information of the whole road networks once.

Table 2: Q value table of 5 ∗ 5 road network with destination Intersection 17.

$Q(s, a)$	Value
$Q(5, 10)$	−112.7293
$Q(5, 4)$	−112.9003
$Q(4, 3)$	−106.3563
$Q(4, 9)$	−107.0302
$Q(3, 8)$	−97.5428
$Q(3, 2)$	−100.6263
$Q(8, 13)$	−79.9226
$Q(10, 15)$	−105.3436
$Q(15, 14)$	−96.4487
$Q(15, 20)$	−98.3531
$Q(14, 13)$	−79.9226
$Q(13, 12)$	−59.5865
$Q(13, 18)$	−59.1124
$Q(12, 17)$	0

Table 3: Q value table of 5 ∗ 5 road network with destination Intersection 14.

$Q(s, a)$	Value
$Q(1, 2)$	−87.6625
$Q(1, 6)$	−93.7241
$Q(2, 3)$	−75.1119
$Q(2, 6)$	−84.9330
$Q(3, 4)$	−62.8019
$Q(3, 8)$	−67.8715
$Q(8, 13)$	−57.0315
$Q(8, 9)$	−46.9100
$Q(13, 14)$	0
$Q(4, 9)$	−46.9100
$Q(4, 5)$	−60.3841
$Q(9, 10)$	−50.4666
$Q(9, 8)$	−67.8715
$Q(9, 14)$	0

Table 4: Cooperative Q value table of 5 ∗ 5 road network with destination Intersection 17.

$Q(s, a)$	Value
$Q(5, 10)$	−112.729
$Q(5, 4)$	−112.9003
$Q(4, 3)$	−106.3563
$Q(4, 9)$	−107.0302
$Q(3, 8)$	−97.5428
$Q(3, 2)$	−100.6263
$Q(8, 13)$	−79.9226
$Q(10, 15)$	−105.3436
$Q(15, 14)$	−96.4487
$Q(15, 20)$	−98.3531
$Q(14, 13)$	−79.9226
$Q(13, 12)$	−82.4741
$Q(13, 18)$	−69.2806
$Q(18, 17)$	0

Table 5: Cooperative Q value table of 5 ∗ 5 road network with destination Intersection 14.

$Q(s, a)$	Value
$Q(1, 2)$	−100.6263
$Q(1, 6)$	−93.7241
$Q(6, 7)$	−87.8625
$Q(6, 11)$	−87.7966
$Q(7, 8)$	−87.5428
$Q(7, 12)$	−92.4741
$Q(8, 13)$	−79.9226
$Q(8, 9)$	−79.0302
$Q(9, 14)$	0
$Q(11, 12)$	−82.4741
$Q(11, 16)$	−94.6919
$Q(12, 13)$	−79.9226
$Q(12, 17)$	−86.1356
$Q(13, 14)$	0

4.2. Cooperative Path Planning for Multiple Agents. This simulation contains two agents. One agent generates the path from Intersection 5 to Intersection 17. The other gives the path from Intersection 1 to Intersection 14. The proposed algorithm introduces the idea of multiagent to lessen the influence that is displayed in Tables 2–5 and Figures 6–9. From the results we can see advantages of MARL.

There is a significant difference between Figures 6 and 7 because Intersection 3 is still busy in Figures 4 and 5 (after cooperation). So after cooperation in Figure 7, all subsequent paths avoid Intersection 3 and 2. Instead, all paths choose Intersection 6 as the next intersection. There is a smaller difference between Figures 4 and 5 as there are only two overlapping intersections between the path from Intersection 5 to 17 and the path from Intersection 1 to 14. Intersection 13 is considered to be the essential intersection, so the algorithm gives up Intersection 12.

4.3. The Comparison of Several Multipath Algorithms. This simulation is made up of 30 nodes with 49 edges in Figure 8. Each circle stands for one intersection S_i. Each edge has three parameters: mean travel time (3 or 7), levels of mean travel time (−1 or −3), and reliability of this edge (0.98 or 0.87).

Under the same simulation environment, we have the following results in Table 6 by Dijkstra's algorithm, k-shortest planning, Q-learning multipath path planning. From the results, we can draw the follow conclusions: Dijkstra's algorithm can give the shortest path with the least time, while the reliability of this path is lower because reliability of each edge is less than 1. The last two algorithms are almost the same. The mean cost of the four paths of k-shortest planning is less than Q-learning multipath path planning, while the reliability of Q-learning multipath path planning is superior to k-shortest planning.

Next, the performance comparison is given for the last two algorithms on the planning time elapsed and the path

Table 6: The comparison results of different algorithms.

Algorithm	Multipath	Costs	Reliability	Time
Dijkstra's algorithm	1-6-7-12-13-14-19-24-25	25	0.57	0.010142
k-shortest path planning	1-6-7-12-13-14-19-24-25	25	0.77	0.104754
	1-6-11-16-21-22-23-24-25	26		
	1-2-7-12-13-14-19-24-15	27		
	1-6-7-12-13-14-19-20-25	27		
Q learning multipath path planning	1-6-11-16-21-22-23-24-25	26	0.95	0.110398
	1-6-11-16-17-18-19-20-25	31		
	1-2-7-12-13-18-19-20-25	30		
	1-2-3-13-18-23-19-20-25	29		

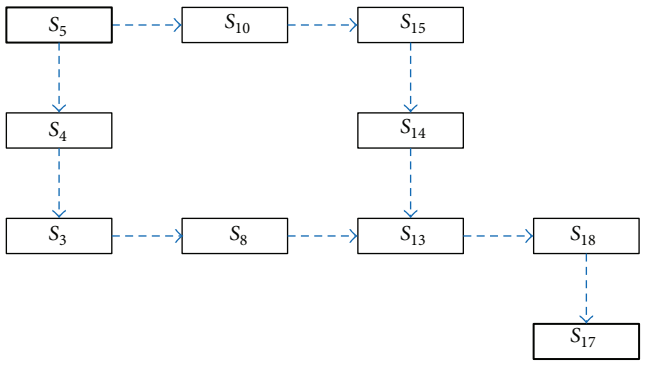

Figure 5: Cooperative multipath planning from Intersection 5 to Intersection 17.

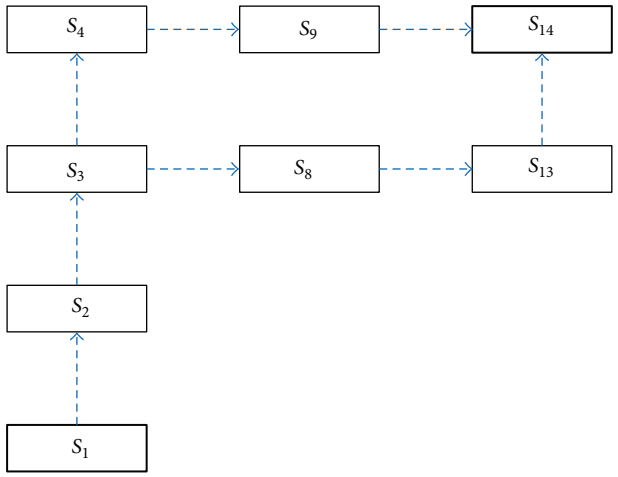

Figure 6: Multipath planning from Intersection 1 to Intersection 14.

reliability. Figure 9 shows that k-shortest planning's time increases linearly over time, while Q-learning multipath planning increases logarithmically. In addition, we can ensure that the reliability of Q-learning multipath planning is more than 0.9. But the reliability of k-shortest planning is less than 0.8.

5. Conclusion

This paper proposes a new Q-learning algorithm to solve the multipath planning problem for OD pairs in a city urban road network. Different from traditional multipath algorithms, this paper focuses on multipath planning via FNN-based Q-learning, an algorithm that makes it easier to choose alternative paths by recurring to the suboptimum Q-value. Furthermore, the paper uses logic to increase the response speed and imposes constraint conditions on path generating to ensure the reliability of the set of candidate paths, which has rarely been taken into account in existing works. Simulation results validate the efficiency and adaptability of the proposed algorithm. In the future work, we will further consider the waiting time at intersections in the algorithm.

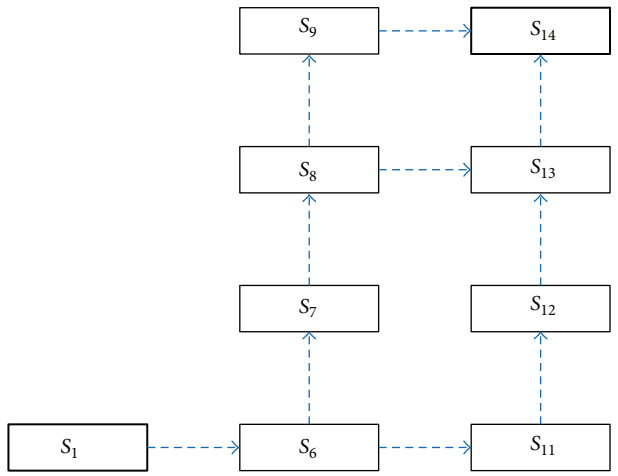

Figure 7: Cooperative multipath planning from Intersection 1 to Intersection 14.

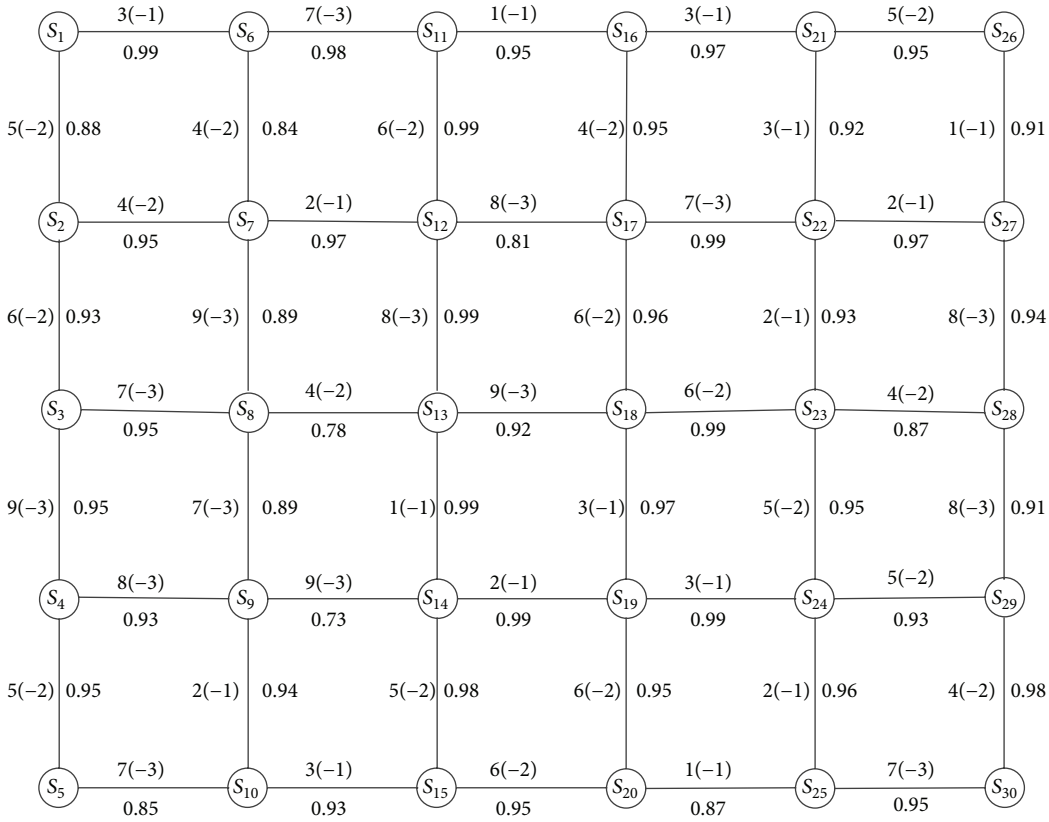

Figure 8: Road networks with 30 nodes.

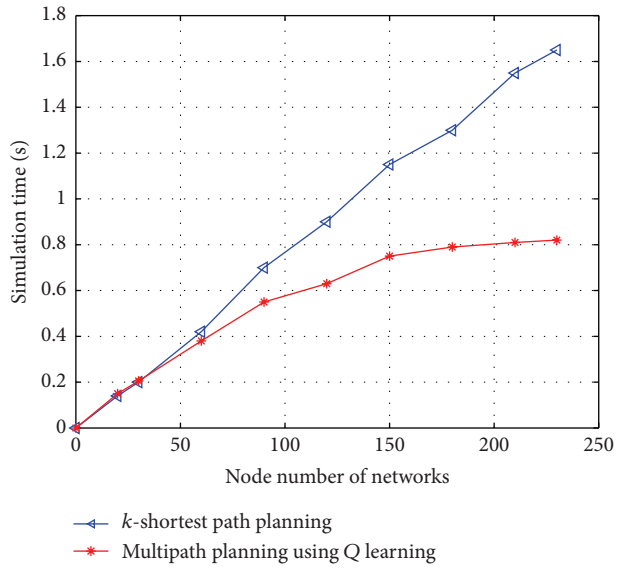

Figure 9: Planning time over the increase of node number of networks with different methods.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] H. Shimoura and K. Tenmoku, "Development of elemental algorithms for future dynamic route guidance system," in *Proceedings of the Vehicle Navigation and Information Systems Conference (VNIS '94)*, pp. 321–326, Yokohama, Japan, 1994.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[3] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[4] C.-K. Lee, "A multiple-path routing strategy for vehicle route guidance systems," *Transportation Research C: Emerging Technologies*, vol. 2, no. 3, pp. 185–195, 1994.

[5] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 652–673, 1999.

[6] V. M. Jiménez and A. Marzal, "Computing the k shortest paths: a new algorithm and an experimental comparison," in *Algorithm Engineering*, vol. 1668 of *Lecture Notes in Computer Science*, pp. 15–29, 1999.

[7] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Math*, vol. 11, no. 5, pp. 1277–1280, 1970.

[8] D. Torrieri, "Algorithms for finding an optimal set of short disjoint paths in a communication network," *IEEE Transactions on Communications*, vol. 40, no. 11, pp. 1698–1702, 1992.

[9] R. S. Sutton and A. G. Barto, "Reinforcement learning: an introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, p. 1054, 1998.

[10] M. Z. Arokhlo, "Route guidance system using multi-agent reinforcement learning," in *Proceedings of the 7th International Conference on Information Technology in Asia (CITA '11)*, pp. 1–5, Kuching, Malaysia, July 2011.

[11] M. Zolfpour-Arokhlo, A. Selamat, and S. Z. M. Hashim, "Self-adaptive and multi-agent reinforcement learning in route guidance system," in *Proceedings of the 5th Malaysian Conference in Software Engineering (MySEC '11)*, pp. 383–387, IEEE, Johor Bahru, Malaysia, December 2011.

[12] Z. Zhang and J.-M. Xu, "A dynamic route guidance arithmetic based on reinforcement learning," in *Proceeding of the 4th International Conference on Machine Learning and Cybernetics*, pp. 3607–3611, August 2005.

[13] C. Wu, O. Satoshi, S. Ohzahata, and T. Kato, "Flexible, portable, and practicable solution for routing in VANETs: a fuzzy constraint Q-learning approach," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 9, pp. 4251–4263, 2013.

[14] K. Y. Chan and T. S. Dillon, "On-road sensor configuration design for traffic flow prediction using fuzzy neural networks and taguchi method," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 1, pp. 50–59, 2013.

[15] C. Li, S. G. Anavatti, and T. Ray, "Short-term traffic flow prediction using different techniques," in *Proceedings of the 37th Annual Conference of the IEEE Industrial Electronics Society (IECON '11)*, pp. 2423–2428, Melbourne, Australia, November 2011.

[16] B. R. Hellinga and L. Fu, "Reducing bias in probe-based arterial link travel time estimates," *Transportation Research, Part C: Emerging Technologies*, vol. 10, no. 4, pp. 257–273, 2002.

[17] M. Tomio, S. Takaaki, and M. Taka, "Route identification and travel time prediction using probe-car data," *International Journal of ITS Research*, vol. 2, no. 1, pp. 21–28, 2004.

[18] Y. Y. Chen, M. G. H. Bell, and K. Bogenberger, "Reliable pretrip multipath planning and dynamic adaptation for a centralized road navigation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 14–19, 2007.

[19] P. Jindahra and K. Choocharukul, "Short-run route diversion: an empirical investigation into variable message sign design and policy experiments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 388–397, 2013.

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

[21] K. Y. Chan, S. Khadem, T. S. Dillon, V. Palade, J. Singh, and E. Chang, "Selection of significant on-road sensor data for short-term traffic flow forecasting using the Taguchi method," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 255–266, 2012.