Edge-based Situ-aware Reinforcement Learning for Traffic Congestion Mitigation

Chen-Yeou Yu, Wensheng Zhang and Carl K. Chang
Department of Computer Science, Iowa State University Ames, USA
{cyy232, wzhang, chang}@iastate.edu

***Abstract*—Traffic congestion may cause elongated travel time, increased fuel consumption and extra pollution. To mitigate congestion, we propose a new approach based on multi-agent reinforcement learning (RL) to learn policies dictating path selections for vehicles. The algorithm utilizes the interactions between RL agents with Q-Learning and edge servers in monitoring traffic at road intersections. As an important difference between this work and existing approaches, we take human desire and realistic rewards into account. Extensive simulation experiments show that the resulting mechanism is promising and more RL agents can be incentive to follow rerouting directions when congestion is detected. Also, this algorithm has comparable performance as the Dynamic Dijkstra Algorithm.**

Keywords—Traffic congestion, Reinforcement Learning, Edge Server, Multi-Agent, Q-Learning

I. INTRODUCTION

Traffic congestion happens when the influx of traffic is greater than road capacity. Many approaches have been proposed to alleviate congestion. One solution is to employ fixed-time traffic signal controls based on historic traffic data [24]. It is easy to be deployed, but may become ineffective when real-time traffic deviates from historical pattern, as the fixed-time traffic signal controls cannot adapt to sudden peaks or lows of traffic, which may be caused by special events such as concerts or sports games. To address the limitations, another possible solution is to install additional hardware with programmable capabilities. For example, cameras at intersections can monitor the busyness of roads and the information can be used to control traffic lights in favor of the road sections already congested.

However, this may not realistic when a city government has limited budget or human resources. Intelligent or programmable equipment need to be purchased and additional man power has to be allocated.

Recently, researchers have proposed more promising approaches [1]-[12] based on machine learning. For example, some studies use artificial intelligence (time-series analytics) to learn traffic patterns in extending or shortening the timing on traffic lights [1]-[4]. Some study [12] proposes to dynamically change the number of lanes in each direction based on analytics. There is study [5] on rerouting vehicles away from congested areas, but shortest-path based algorithms are used which cannot reflect the everchanging road conditions. Studies have also shown that reinforcement learning (RL) agents, with proper training, perform better in controlling traffic signals, opening-closing ramps and changing the direction of lanes in responding to dynamically varying traffic conditions [12].

Traffic congestions often have strong locational binding, for example, they may be caused by road construction, car crash site or an event such as a concert. Fortunately, with the fast evolution of cloud computing, edge servers serve as a revamping technology over its predecessors by offering a variety of services getting closer to the people. They push services to the edge. Edge servers can efficiently facilitate local information exchange, especially for the applications needing very short response time [13]. Hence, we can use edge servers to monitor traffic condition at road sections and communicate with RL agents on vehicles. When traffic congestion happens, it can be sensed by local edge server. A vehicle rerouting strategy can then be found quickly in responding to the congestion.

The purpose of this study is to propose a novel RL-based rerouting solution to mitigate congestion. Edge servers are set up at road intersections to collect local traffic information and synchronize with their peers. RL agents are deployed at vehicles and frequently get updates on local traffic information whenever they enter the range of an edge server. What makes our study different is that we incorporate a human-centered design in our agents as a desire to collect bonuses while path efficiency is still considered. To engage as many vehicles in cooperating with the direction to be rerouted away from congestion areas, drivers can get realistic rewards as good incentive. Meanwhile, the setup of punishments for RL agents is designed to avoid RL agents from being overly greedy and making too many detours. As a result, there is no additional need to learn about the historic traffic data and unnecessary controls on the traffic lights can be avoided. Fig. 1 is an illustration of congestion rerouting.

In the rest of the paper, Section II discusses related work. Section III presents the preliminaries and problem formulation. Section IV describes our proposed model in dynamically changing the behavior of RL agents by using fused information from edge servers. Section V reports the evaluation works and analytics. Finally, Section VI concludes the paper.

II. RELATED WORK

A. Situ

The Situ framework [14] is formulated as $S = (d, A, E)_t$, where d is the desire of a driver or an agent for achieving a goal,

such as receiving a reward. A is the set of actions performed by an agent and E is the observed environmental-contextual information at time t. This contextual information (E) can be collected by sensors, or any applicable hardware and software. Desire (d) belongs to internal mental state and is usually hard to detect, changing over time. Most researchers focus on expressive behavior (e.g., change a lane or turn right) [15]-[17] without considering the changes of a driver's desire. This change can affect the policy. Under the influence of different policy, RL agent takes different action to interact with the environment and yielding different rewards. Situ framework gives a good theoretical foundation indicating the desire that evolves. If the desire is changed, a list of actions (A), as options can be performed could change as well.

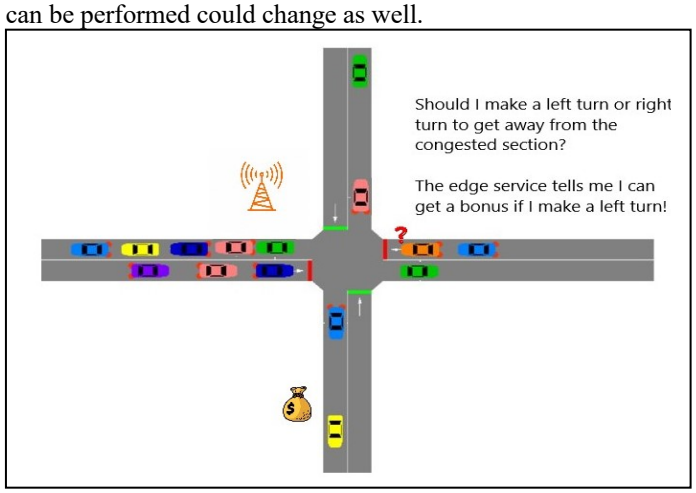

Fig. 1. An example of a rerouting scenario

B. RL-based Traffic Flow Control and Optimization

In [5], the authors present graph attention network-deep Q learning (GAQ) along with Entropy-based k Shortest Path (EBkSP) approach to alleviate congestion problem with consideration in path planning.

In [18], the authors succeed in reducing traveling time using reinforcement learning. Other researches use multi-agent reinforcement learning to control traffic lights with multiple objectives, minimizing traveling time, increasing safety and controlling the speed. For specific applications, in [19], reinforcement learning is used to dynamically change the sections of highway speeds and the study in [12] uses Q-Learning to control the opening or closing the ramps and change the number of lanes in different directions.

C. Edge / Fog Computing and Traffic Flow Management

In vehicular edge / fog computing networks, edge servers can detect the local traffic conditions and share the information with the cloud or peers that can be used by RL agents in adaptive rerouting. For example, Cao et al. [20] propose to use V2V and V2I communications to share the fused, real time traffic information. Multiple alternative routes are computed and randomly dispatched to vehicles to prevent the same road section from always being selected. Brennand et al. [21] introduced a fog-based architecture to assist Intelligent Transport System (ITS) in relieving traffic congestions in the urban area. They make use of information gathered from the road side units (RSU) as well as fog services to deliver alternative routes.

III. Preliminaries and Problem Formulation

A road network model is a commonly used to preserve characteristics of road networks. The model is composed of road segments called edges and road intersections called nodes. When driving on the network, vehicles can change their routes by making a left or a right turn.

A. Traffic-aware road networks

Considering that most of the road networks are designed with the bi-directional capability, we define our road networks as $G(V, E)$ with the following parameters

$V = \{ v_1, v_2, \ldots, v_n \}$ represents the set of nodes, corresponding to intersections in the road network.

$E = \{ e_1, e_2, \ldots, e_m \}$ represents the set of edges, corresponding to road segments each connecting two adjacent nodes in V

D is the distance matrix, represented by

$$D = \begin{bmatrix} d_{11} & d_{12} & \ldots & d_{1n} \\ d_{21} & d_{22} & \ldots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \ldots & d_{nn} \end{bmatrix}$$

where d_{ij} denotes the distance between v_i and v_j

t_m is a function which takes an edge e_i and outputs the free-flow travel time of e_i when there are no vehicles on that edge.

B. Traffic Flow (Number of Vehicles on an Edge)

We define traffic flow as the number of vehicles on an edge e_i at a particular time t, represented by the following matrix:

$$F(t) = \begin{bmatrix} f_{11} & \cdots & f_{1n} \\ \vdots & \ddots & \vdots \\ f_{n1} & \cdots & f_{nn} \end{bmatrix}$$

where f_{ij} represents number of vehicles form v_i to v_j at time t. It serves as a snapshot of the traffic flow on an edge. For ease of representation, we use f_e(t) to represent element f_{ij} of $F(t)$. As shown below, f_e (t) counts two types of vehicles: those following the rerouting on the congested road sections and the others.

$$f_e(t) = f_{e.follow}(t) + f_{e.not\_follow}(t) \quad (1)$$

C. Congestion Level

We assume that congestion level is computed by edge services. The congestion level on a road section i is defined as [5]:

$$Cong_i(t) = f_e(t)/(d_{ij} \times L_{i,j}) \quad (2)$$

Here $L_{i,j}$ represents the number of lanes for the edge connecting v_i and v_j. We further assume each road section has a fixed capacity $Cap(e_i)$. We can say the section is congested if:

$$Cong_t(e) \leftrightarrow Cap(e_i) < f_e(t) \quad (3)$$

D. Reinforcement Learning and Agent Design

The behavior of RL agents can be modeled as a discrete Markov Decision Process (MDP) along a route R to maximize the system reward. We define the variables that the RL agent needs to use as follows.

State

We model the road network as a gird map. Each road intersection corresponds to a coordinate where we have RL agent's observation on current position as state s_t. For a $N \times N$ grid map, the first part of a state s is expressed as:

$$v_t = (x_i, y_i) \in X \times Y$$

where X, Y $\in [0, N]$

If a RL agent's current state $s_t$is moving to the next state s_{t+1}, the agent moves to the next node (intersection). Finally,

$$s_t = \{v_i, \phi(v_i)\} \quad (4)$$

$\phi(v_i)$ is the minimal hops to destination.

Action

Actions are options that can be performed by RL agent. In each intersection, RL agent selects an action a_t according to policy $\pi(a_t|s_t)$ before receiving a reward r_t from the environment.

The policy $\pi(a_t|s_t)$, by definition from MDP, is actually a transition probability.

$$\pi(a_t|s_t) = P_r(a_t = a|s_t = s) \quad (5)$$

The policy can be changed dynamically. When the ITS wants more agents to follow rerouting policies, the transition probability can be adaptively tuned up to lure the RL agent to take the action a_t. Full action space for the agent can be expressed as

$$A = \{up(Forward), down(U-tuen), left, right\} \triangleq \{0, 1, 2, 3\} \quad (6)$$

Note that, when it is located in the border of the grid map, the RL agent might select an action and move out of the map. We set the Q-Value table with a $-\infty$ to decrease the chance on that option being selected. If it is selected, we determine the selected action is invalid and will have the agent select other option from remaining directions.

Reward

In this paper, we use the realistic rewards as incentive to deposit bills to vehicles, for example, E-ZPass. We expect that, by using this approach, more and more vehicles can follow the rerouting guidance from ITS before they enter congested road sections.

In our reward function, we consider both reward and penalty. The penalty is to prevent the agent from choosing routes in the opposite direction under similar road conditions, for example, unnecessary U-turns. Meanwhile, to reward the agent getting into the goal, $r_{goal}(s, a)$ is a "special" reward that can make sure the agent will finish the job.

The first part of reward is related to traffic flow. From (3), a RL agent gets more reward when an edge with lower traffic flow is chosen. We use $r_f$to denote the part of reward from choosing the right edge. K is the baseline and it can be set as any $K > 0$. α is the coefficient of baseline reward K.

$$r_f = \alpha K, Cap(e_i) > f_e(t)$$
$$r_f = K, Cap(e_i) < f_e(t) \quad (7)$$

The second part of reward is related to the length of the edge. A RL agent is encouraged to take the shortest edges to the destination. We use r_d to denote the part of reward from choosing the right edge. β is the coefficient of the baseline reward K. $\pi(v_i)$ means all the edges starts from the v_i

$$r_d = \beta K, d_{ij} = argmin_{d_{ij}}(d(\pi(v_i)))$$
$$r_d = K, d_{ij} \neq argmin_{d_{ij}}(d(\pi(v_i))) \quad (8)$$

The third part of reward is related to rerouting. A RL agent gets special bonus if it is a cooperating RL agent following rerouting policy. The bonus is set by the edge server and the amount of the bill depends on the policy from ITS. "B_r" is the rerouting bonus which can be a positive number. "$Bonus()$" is a Boolean function which is used by the RL agent for state observation and checking on the edge.

$$r_r = B_r, Bonus(e_{ij}) = 1$$
$$r_r = 0, Bonus(e_{ij}) = 0 \quad (9)$$

Then, the $Bonus_r$ matrix is created according to (9) on all edges in the road networks.

The fourth part of reward is a punishment. This part is to prevent the agent from meaningless wanderings. $-B_h$is the punishment for agents by checking $\phi(v_i)$, the minimal hops for a RL agent from its current node v_i to destination.

$$r_h = -B_h, \phi(v_i) < \phi(v_{i+1})$$
$$r_h = 0, \phi(v_i) > \phi(v_{i+1}) \quad (10)$$

The fifth part of reward is the bonus to encourage the RL agent to get to the destination. γ is a coefficient and works similarly like α and β.

$$r_{goal} = \gamma K, \text{ goal arrival}$$
$$r_{goal} = 0, \text{ otherwise} \quad (11)$$

From (7)-(11), we have:

$$r = r_f + r_d + r_r + r_h + r_{goal} \quad (12)$$

E. Traveling Time Function

Traveling time, by default, is always: $distance/velocity$. The travel time function is to compute simulated traveling time on an edge when the traffic flow is considered. $t_m(e)$ is a baseline of time consumed on an edge.

Following the study [23], $t(e)$ can be defined as:

$$t(e) = t_m(e) + \alpha \times f_e \times t_m(e) \quad (13)$$

α is the road specific parameter which is different by sections. It could be the any linear combination of speed limit, lanes or length of the section. In our simulation, we assume the speed limits on all the sections are the same and one lane on different bounds. We only consider the traveling distance and the traveling time. For **normal** traffic load ($Cap(e_i) > f_e(t)$), it can be formulated as traveling time = $1 \times t_m(e) \times 0.5$, for short sections and traveling time = $1 \times t_m(e) \times 1.5$, for long sections. Similarly, for **congested** traffic load ($Cap(e_i) < f_e(t)$) can be formulated as traveling time = $2 \times t_m(e) \times 0.5$, for short sections and traveling time = $2 \times t_m(e) \times 1.5$, for long sections. (14)

F. Problem Formulation

Given a traffic-aware road network $G(V, E)$, distance matrix D, finite set of RL agents $A = \{a_1, a_2, \dots, a_n\}$, each has source / destination pair (src_i, dst_i). The congestion rerouting algorithm is to find the path for each agent a_i along a route R_i.

Meanwhile, during the process of routing and rerouting, RL agent needs to use Sutton's recursive Bellman equation [22] to compute and update the Q-values in the Q-value table. The agent needs to make the decision at each node and yield the maximum rewards.

Different vehicles, because of the path planning might be different, they will have different traveling time. One of our goals is to minimize the traveling time for all vehicles f.

Finally, if a RL agent takes a reward for rerouting, that means f_{follow} is accumulated by one and f_{not_follow} reduces by one. Our algorithm is aiming to maximize the total $f_{follow}$and minimize the total f_{not_follow}

Objective:

Minimize

$$d(R_i) = \sum_{i=src, j=src}^{x=dst, y=dst} d_{ij} \quad (15)$$

$$f_{not\_follow} \quad (16)$$

Total Traveling Time for f (17)

Maximize

$$Q^{dst}(s_{dst}, a_{dst}) \quad (18)$$

$$f_{follow} = \sum_{i=1}^{m} f_{e_i.follow} \quad (19)$$

s.t.

$$f = f_{follow} + f_{not\_follow} \quad (20)$$

$$\pi(x) = argmax_a Q(s, a'), 1 - \varepsilon \quad (21)$$

In (21), $a'$is one of the selectable actions in the current state. For Q-learning perspective, the RL agent has to choose the action corresponding to the maximum Q value in the set of actions under the policy π. ε is the epsilon-greediness.

Note that, for each agent a_i in the route planning, it has a source and a destination. For different agents a_i in $G(V, E)$, they can share the same source and destination.

IV. PROPOSED FRAMEWORK

In this section, we present the framework of our solution, a reward-based multi-agent Q-Learning in distributed edge computing for adaptive rerouting. During the process of rerouting, Q-value table is maintained which contains value of state-action pairs.

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}}\Big)}^{\text{temporal difference}}$$

Fig. 2. Q-value (Bellman Equation [22])

The detail of Q-value is defined in Fig. 2. When the RL agent is at intersection i, he needs to take an action based on Q-value. Q-values are used to evaluate the action a. We can know how good this action is based on the value.

α is the learning rate. It is usually set between 0 and 1. "0" means that the Q-values are never updated. The agent learns nothing by simply exploiting its prior knowledge (old value) while the value set to "1" makes the agent to consider only the most recent information. Similarly, the discount factor determines how much the RL agent cares about rewards in the distant future relative to those in the immediate future. If $\gamma = 0$, the agent will be completely myopic. It will only learn about the actions that produce the immediate reward. If $\gamma = 1$, the agent will evaluate each of his actions and chooses the best action that can maximize the Q-value. Fig. 3 is the detail of system model.

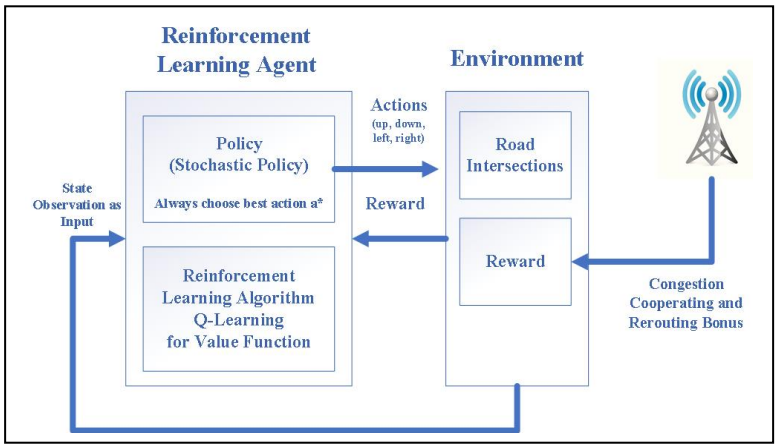

Fig. 3. System Model

A. Distributed Edge Learning for Congestion Management

Assume the infrastructure has allocated enough edge servers in performing our smart traffic congestion mitigation mechanism. In algorithm1, edge servers synchronize with RL agents. The purpose of this algorithm is to have edge services keep monitoring the real time congestions on road sections. If the congestion happens, the corresponding edge server can dynamically setup edges with bonuses. In Fig. 4, by using this kind of edge learning approach, RL agents can observe the real time road conditions and where the bonus is.

Algorithm 1: Distributed Edge Learning for Congestion Management

Require:
Edge servers $\boldsymbol{Eg} = \{\boldsymbol{Eg_1}, \boldsymbol{Eg_2}, \ldots, \boldsymbol{Eg_n}\}$ deployed at the road intersections, corresponding to V
RL agent from $\boldsymbol{Ag} = \{\boldsymbol{Ag_1}, \boldsymbol{Ag_2}, \ldots, \boldsymbol{Ag_p}\}$
Initialize: Shared memory matrix $D, F, Cap, Bonus_r$
Initialize: Shared variables $f_{follow}, f_{not\ follow}$

Share $D, F, Cap, Bonus_r$ to all Eg_i

While TRUE **do**
  for Edge server Eg_i from Eg_1 to Eg_n **do (in parallel)**
    if $Cap(e_i) < f_{e_i}$
      Set rerouting road sections on $Adj(e_i)$, where $|Adj(e_i)| \leq 2$,
      Set $r_r = B_r$ onto $Adj(e_i)\ \forall e_i \in E$, depending on the policy of ITS (9)
      Update corresponding $r_r \in Bonus_r$
    end if
  end for

  for RL agent Ag_i from Ag_1 to Ag_p **do (in parallel)**
    Initialize the Q-value table for Ag_i by setting all the Q-value cells to 0
    Notify Ag_i to perform Q-Learning routing and congestion rerouting
    if Ag_i.follow == TRUE
      $f_{follow} = f_{follow} + 1$
      $f_{not\ follow} = f - f_{follow}$, according to (20)
    end if
  end for

end

Fig. 4. Distributed Edge Learning Algorithm for Congestion Management

B. *Multi-Agent Adaptive Congestion Rerouting*

To speed up in the efficiency of exploration and rerouting, we propose an approach consisting of two steps, offline training and online adaptive congestion rerouting. In step one, we prepare a list of random source and destination pairs for RL agents to perform random walks. The purpose is to update the Q-value table based on the simple traffic conditions in the road sections. In step two, we put the RL agents online for simulation by randomly assigning them a source and destination pair. When an agent enters a new section, it has to register to the edge server in charge to synchronize all the parameters, $Cap(e), f_e(t), d_e, \phi_t(v_i)$ *and* the matrix $Bonus_r$ for decision making. After the Q-values are computed, it has to select an action based on epsilon-greedy method with parameter ε. The rewards are collected during the execution of the action and the agent moves from this intersection with s_t to the next one with s_{t+1}.

When the RL agent observes a new state, from Fig. 2, it estimates the optimal future value. Obviously, an agent doesn't take a congested section because there is no benefit (9). The agent can see the rerouting edges adjacent to the congested one and it will take it because its E-ZPass will have new incoming bills if it follows the detouring policy. As a result, the influx to the congested road section decreases and congestion is mitigated.

V. EVALUATION AND RESULTS

A. *Experimental Settings*

We build the simulation experiment on a 5×5 grid (16 intersections) [24] [25] with 7 and 10 vehicles.

Algorithm 2: Multi-Agent Adaptive Congestion Rerouting

Initialize: a list $L_{simulation\_pair}$ of random source and destination pairs $< src, dest >$ for simulation

Require:
Edge servers $\boldsymbol{Eg} = \{\boldsymbol{Eg_1}, \boldsymbol{Eg_1}, \ldots, \boldsymbol{Eg_n}\}$ deployed at the road intersections, corresponding to V
RL agent from $\boldsymbol{Ag} = \{\boldsymbol{Ag_1}, \boldsymbol{Ag_1}, \ldots, \boldsymbol{Ag_p}\}$, where $|L_{simulation\_pair}| = k$, is much smaller than $|\boldsymbol{Ag}|$
α, learning rate, where $(0 \leq \alpha \leq 1)$
γ, a discount rate, where $(0 \leq \gamma \leq 1)$

Offline Training:
for RL agent Ag_i from Ag_1 to Ag_p **do**
  Generate a list L_{train_pair} of random source and destination pairs $< src, dest >$ to Ag_i
  for each $< src, dest >$ in L_{train_pair}
    Randomly assign $< src, dest >$ to Ag_i
    Random actions for each Ag_i till reaching $dest$
    Update the Q-values in the Q-value table
  end for
end for

Online Adaptive Congestion Rerouting:
for RL agent Ag_i from Ag_1 to Ag_p **do (in parallel)**
  Randomly assign $< src, dest >$ from $L_{simulation\_pair}$ to Ag_i
  for episode $epi = 1, 2, \ldots, epi_{max}$ **do**
    Initialize the initial position src as state $s_1 \in S$
    for each time step t **do**
      if s_t is the terminal state
        $Q(s_t, a_t)$ is converged
      else
        Register Ag_i to the edge server in range, by the end of the road section
        Sync with the edge service
        Get the updated information about this road section in time step t: $Cap(e), f_e(t), d_e, \phi_t(v_i), Bonus_r$
        Calculate Q-values based on the current state s_t
        Update the Q-value table by the formula from Fig. 2
        Based on Q-values, choose an action a_t using epsilon-greedy method with parameter ε,
        where $a_t = argmax_a Q(s_t, a_t)$

        Go to the direction indicated by a_t
        Update $r_f, r_d, r_r, r_h, r_{goal}$
        if $r_r == B_r$
          $Ag_i.follow \leftarrow$ TRUE
        Obtain the reward r_t according to (12)
        Deregister Ag_i to the edge server in range
        Move from state s_t to state s_{t+1}, $s_{t+1} \leftarrow Transition(s_t, a_t)$
      end if
    end for
  end for
end for

Fig. 5. Multi-Agent Adaptive Congestion Rerouting

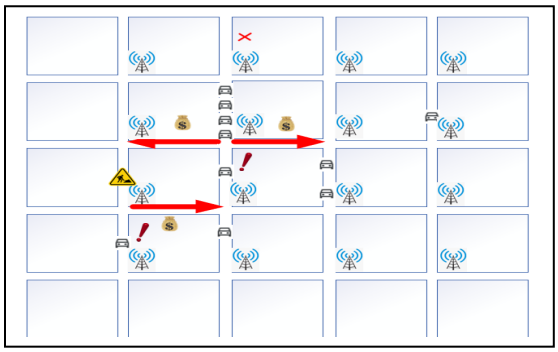

Fig. 6. 5 x 5 grid with 10 vehicles, 1 destination, 1 congested section, 1 road construction and the 2 rerouting schemes (it has directions)

It follows by 3×3 (4 intersections) with 4 and 7 vehicles. A RL agent needs to register to the edge server in charge of the section and deregister when it leaves. Fig. 6 is the use case of 5×5 grid with 10 vehicles. A congestion happens and a road construction is occupying another section. Rerouting scheme is set by edge servers. Three edges are set in the red color. Table I is the simulation parameters.

TABLE I. SIMULATION PAPAMETERS

	3 x 3 Grid		*5 x 5 Grid*	
Vehicles	$f = 4$	$f = 7$	$f = 7$	$f = 10$
Capacity / Distance	***Randomly initialized as:*** $2 \leq Cap(e_i) = d_{ij} \leq 4$			
Traveling Time	**Assuming $t_m(e) = 1$ *minute*,** for normal and high traffic load are set according to (14) **Congestion happens when,** $f_e \geq Cap(e_i)$**:** It will cost **2 more minutes** for all vehicles in the road section, then resolved.			
Episode	5 times, then 10 times, applied onto 4 use cases			
Baseline Algorithms	Random Selection [25], Shortest Path [25], Dynamic Dijistra [24]			
ε greedy	$\varepsilon_{train} = 0.5$, for training $\varepsilon_{online} = 0.2 \sim 0.0$, for online simulation			
α learning rate	0.9 [26]			
γ discount factor	0.95 [25]			
Reward Specific Parameters	K=0.01, $\alpha = 2, \beta = 2, \gamma = 10, B_r = 1, B_h = 1$			
Parallel Execution	**Yes**, for all vehicles can move at the same time in front of extrance			
Incidents	**Congestion**: Naturally formed by formula (3) **Construction or Car Crash:** Random (0~1)			
Sources, Destinations	**Sources:** Multiple sources **Destinations:** 1 for 3 x 3 graid; 1~2 for 5 x 5 grid			
Rerouting Sections	Set by edge server; Removed when resolved.			

The baseline methods are used to compare with our algorithm. The major difference is that, when congestion happens, there is no rerouting scheme for Random Selection, Shortest Path and Dynamic Dijkstra. For Random Selection, RL agent randomly selects an intersection as the direction of moving. For Shortest Path, RL agent moves along the shortest path according to the information d_{ij} in D matrix regardless of congestion. For Dynamic Dijkstra [24], it always chooses the sections with minimal traveling time. We can interpret the meaning as, when a rerouting is needed, the agent always chooses sections with the lowest traveling time.

The value of ε in the training stage, we give it a high exploration rate of 0.5. We hope the agent can find out as much information as possible about sections. For online simulation, we give it a 0.2 as initial exploration rate, then it decays 0.05 after a new state is observed. The setting is reasonable because when the RL agent is approaching destination, it has more chances getting into congested section and it has to follow rerouting policy set by ITS. Since the information is already collected in the training stage, RL agent can be greedy as it is described in (21). The α learning rate is set to fixed value 0.9. As for γ, this parameter is set to 0.95, as it will evaluate all the actions and choose the best action maximizing the Q-value. The experimental results and analysis are shown in the next section.

B. *Experimental Results*

Before experiment, all vehicles are uniformly placed in front of the entrance on the grid. Here are the results.

1) Average Route Distance (Shorter is better)

Shortest Path always gets the best traveling distance regardless of the traveling time as it is expected. Dynamic Dijkstra has slightly better performance than our Q-Learning algorithm as it is always choosing the edges with minimal traveling time, sometimes indicating shorter distance. The rerouting sections, randomly assigned by edge services adjacent to the congested section as detouring purpose sometimes increases the route distance.

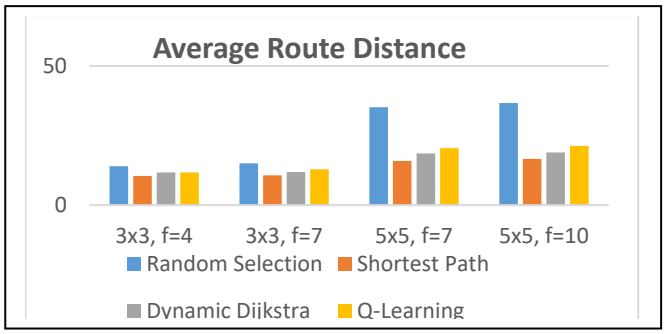

Fig. 7. Average Route Distance

2) Average Total Traveling Time (Shorter is better)

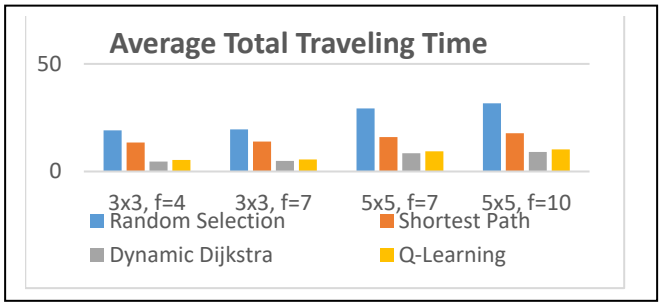

Fig. 8. Average Total Traveling Time

In Fig. 8, Shortest Path Algorithm takes a fairly long time whereas the Dynamic Dijkstra serves as a lower bound because of the greediness in choosing the edges with lowest traveling time. Our Q-Learning algorithm, the performance is at par with Dynamic Dijkstra. The root cause is noticed in the simulation. When the edge server suggests rerouting edges, it might be producing another potential congested section. The influx of the traffic is redirected and the vehicles gets stuck in the newly introduced sections. The edge servers will end up with creating more rerouting sections.

3) Average number of rerouting policy following vehicles

From Fig. 9, our rerouting algorithm works when more vehicles are placed into the simulation. By setting the bonus on the rerouting edges from the edge services, as well as the designs in our rewards, RL agents will take the road sections with bonus when it is an available option.

VI. CONCLUSION AND FUTURE WORKS

We propose a congestion mitigating algorithm by setting rerouting road sections adjacent to congested ones from edge servers. Our goal is to minimize the traveling time, traveling distance and engaging more vehicles in rerouting policy. Unlike other studies, our approach does not heavily rely on historical data or complicated training on machine learning model. The RL agents interacts with the environment by observing situational environment, learning and making decision by itself.

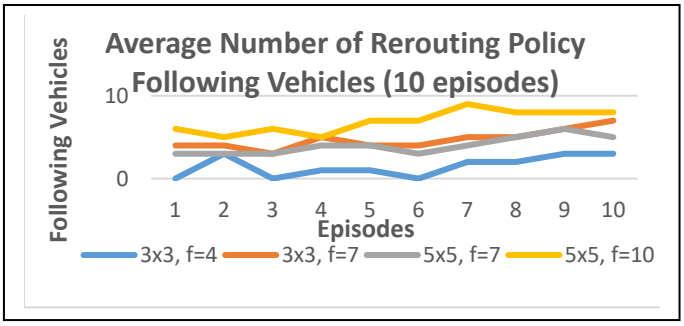

Fig. 9. Average Policy Following Vehicles (10 episodes)

Our proposed method doesn't need additional hardware investments or roadmap changes. Simulation results show that the performance of our method can be at par with the Dynamic Dijkstra and still saves about 40% of traveling time compared with traditional Shortest Path provided by GPS boxes.

Note that, a "congestion shifting" [5] happens in the simulation. The distance and capacity of sections are randomly generated. When the distance is too short or the capacity is lower, congestion happens more frequently. It is obvious when the volume of vehicle is increasing with the relative smaller grid (3 x 3). In our simulation, we only consider the approach to randomly assign a rerouting edge adjacent to the congested ones from the upstream of the influx. In the future, the case of congestion shifting has to be studied. The placement of rerouting edges, not only the number but also the locations has to be researched as well.

REFERENCES

[1] F. I. Shashi, S. M. Sultan, A. Khatun, T. Sultana, and T. Alam, "A Study on Deep Reinforcement Learning Based Traffic Signal Control for Mitigating Traffic Congestion," 2021 IEEE 3rd Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), 2021. Available: https://doi.org/10.1109/ECBIOS51820.2021.9510422

[2] H.Joo, S. H. Ahmed, and Y. Lim, "Traffic signal control for smart cities using reinforcement learning,"Computer Communications, vol. 154, pp. 324–330, March 2020.

[3] S. Touhbi, M. A. Brbram, H. T. Nguyen, N. Marilleau, M. L. Hbid, C. Cambier and S. Stinckwich, "Adaptive Traffic Signal Control : Exploring Reward Definition For Reinforcement Learning," Procedia Computer Science, vol. 109, pp. 513-520, 2017

[4] J. Gu, M. Lee, C. Jun, Y. Han, Y. Kim and J. Kim, "Traffic Signal Optimization for Multiple Intersections Based on Reinforcement Learning," Applied Sciences, November 2021. Available: https://doi.org/10.3390/app112210688

[5] R. Du, S. Chen, J. Dong, P. Ha and S. Labi, "GAQ-EBkSP: A DRL-based Urban Traffic Dynamic Rerouting Framework using Fog-Cloud Architecture," 2021 IEEE International Smart Cities Conference (ISC2), 2021. Available: https://doi.org/10.1109/ISC253183.2021.9562832

[6] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1140–1150, 2013.

[7] M. Aslani, S. Seipel, M. S. Mesgari, and M. Wiering, "Traffic signal optimization through discrete and continuous reinforcement learning with robustness analysis in downtown tehran," Advanced Engineering Informatics, vol. 38, pp. 639–655, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034617302598

[8] K. Ahmadi and V. H. Allan, "Checking the reliability of information sources in recommendation based trust decision making," in PRIMA 2015: Principles and Practice of Multi-Agent Systems, Q. Chen, P. Torroni, S. Villata, J. Hsu, and A. Omicini, Eds. Cham: Springer International Publishing, 2015, pp. 367–382.

[9] K. Yau, J. Qadir, H. Khoo, M. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," ACM Computing Surveys, vol. 50, no. 3, Jun. 2017.

[10] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," ACM Computing Surveys 2021, vol. 54, no. 6. Available: https://doi.org/10.48550/arXiv.2005.10619

[11] E. Walraven, M. T. Spaan, and B. Bakker, "Traffic flow optimization: A reinforcement learning approach," Engineering Applications of Artificial Intelligence, vol. 52, pp. 203–212, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197616000038

[12] K. Ahmadi and V. H. Allan, "Smart City: Application of Multi-agent Reinforcement Learning Systems in Adaptive Traffic Management" 2021 IEEE International Smart Cities Conference (ISC2), 2021. Available: https://doi.org/10.1109/ISC253183.2021.9562951

[13] C. Yu, C. K. Chang and W. Zhang, "A Situation Enabled Framework for Energy-Efficient Workload Offloading in 5G Vehicular Edge Computing," 2020 IEEE World Congress on Services (SERVICES), 2020. Available: https://doi.org/10.1109/SERVICES48979.2020.00027

[14] Carl K. Chang, "Situation Analytics: A Foundation for a New Software Engineering Paradigm," Computer, vol. 49, no. 1, 2016, pp. 24-33.

[15] Jiayin Cen, Zhuowen Wang, Chao Wang, Fuqiang Liu, "A System Design for Driving Behavior Analysis and Assessment," 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016. Available: https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.182

[16] M. Jensen, J. Wagner and K. Alexander, "Analysis of in-vehicle driver behaviour data for improved safety," International Journal of Vehicle Safety (IJVS), vol. 5, no. 3, 2011, pp. 197-212.

[17] J. Paefgen, F. Kehr, Y. Zhai and F. Michahelles, "Driving behavior analysis with smartphones: insights from a controlled field study," MUM 2012, no. 36, pp. 1-8.

[18] M. Zolfpour-Arokhlo, A. Selamat, S. Z. M. Hashim, and H. Afkhami, "Modeling of route planning system based on Q value-based dynamic programming with multi-agent reinforcement learning algorithms," Eng. Appl. Artif. Intell., vol. 29, pp. 163–177, Mar. 2014

[19] E. Walraven, M. T.J. Spaan and B. Bakker, "Traffic flow optimization: A reinforcement learning approach," Engineering Applications of Artificial Intelligence, vol. 52, pp. 203–212, June 2016.

[20] A. Cao, B. Fu, and Z. He, "ETCS: An efficient traffic congestion scheduling scheme combined with edge computing," 2019. Available: https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00378

[21] C. A. R. L. Brennand, A. Boukerche, R. Meneguette and L. A. Villas, "A Novel Urban Traffic Management Mechanism Based on FOG," 2017 IEEE Symposium on Computers and Communications (ISCC). Available: https://doi.org/10.1109/ISCC.2017.8024559

[22] R. S. Sutton, "Temporal Credit Assignment in Reinforcement Learning," (PhD thesis). University of Massachusetts, Amherst, MA, 1984

[23] S. Lim and D. Rus, "Stochastic distributed multi-agent planning and applications to traffic," ICRA, pp. 2873–2879, 2012.

[24] H. Su, Y. D. Zhong, B. Dey and A. Chakraborty, "A Decentralized Reinforcement Learning Framework for Efficient Passage of Emergency Vehicles," Artificial Intelligence and Humanitarian Assistance and Disaster Recovery (AI + HADR) workshop, NeurIPS 2021. Available: https://doi.org/10.48550/arXiv.2111.00278

[25] Y. Geng, E. Liu, R. Wang and Y. Lu, "Deep Reinforcement Learning Based Dynamic Route Planning for Minimizing Travel Time," 2021 IEEE International Conference on Communications Workshops (ICC Workshops), November 2020. Available: https://arxiv.org/abs/2011.01771v1

[26] C. Chen, J. Jiang, N. Lv and S. Li, "An Intelligent Path Planning Scheme of Autonomous Vehicles Platoon Using Deep Reinforcement Learning on Network Edge," IEEE Access, vol. 8, May 2020. Available: http://dx.doi.org/10.1109/ACCESS.2020.2998015