TWO TIER HYBRID ROUTING SCHEMES FOR VEHICULAR AD HOC NETWORKS BASED ON REINFORCEMENT LEARNING

NARGES HAGHIGHATI BOROUJENI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF APPLIED SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING & COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

MAY 2021

Abstract

Vehicular ad-hoc networks(VANETs) consist of vehicles and roadside units collaborating to provide mobile wireless communication services. VANETs have been created to address the need of managing and analyzing the traffic flow to keep up the roads' safety and avoid collisions while avoiding overwhelming costs that using cellular networks can add.

This research addresses the issue of routing messages from vehicles to specific geographical locations in VANETs. To overcome this challenge we propose a hybrid system consisting of traffic-aware located RSUs and a reinforcement learning routing strategy. Our proposed method includes two main tiers. We first divided the geographical area into equal-sized grids; then using the traffic flow patterns of the area, the protocol will suggest the optimal locations for RSUs with the objective of minimizing the number of RSUs in the system to cut down the cost while improving the delivery ratio at the lowest possible delay.

Moreover, there are some patterns in the traffic flow of every city. This feature makes machine learning algorithms to be a potentially promising tool to work with. We have chosen Q-learning, a reinforcement learning algorithm, to extract these patterns and provide a routing policy for the vehicles. This policy determines which grid the message should be

sent to in the next step along the way for each location in the city. After the Q-learning algorithm learns the policy, it is broadcast to all of the city's vehicles.

In the next step, we choose the vehicle which is supposed to receive the packet, and we do that by greedily choosing the one closest to the destination or an RSU. The result of our extensive experiments shows that our proposed method improves the message delivery rate and decreases the delay in comparison to a well-known conventional method, GPSR, and a Q-Learning-based method, QGrid_G. Moreover, we have minimized the performance's sensitivity to VANETs' attributes such as velocity or vehicle density on the roads.Furthermore, compared to one of the latest related work, which also considers hybrid communication, we managed to deploy a far fewer number of roadside units to achieve the same performance.

Keywords – Vehicular ad hoc network, routing, position-based routing, Q-learning.

Acknowledgements

Dr. Suprakash Datta, my supervisor, for his guidance and support throughout the last two years.

My husband, my family and friends, without whom it wouldn't have been possible to make it through during the COVID crisis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The rapid growth of the number of vehicles and the development of related technologies has highlighted the need for Intelligent Transport Systems (ITS). An efficient ITS should address transportation challenges like managing and analyzing the traffic flow to enhance road safety, avoid collisions, and improve traffic flow using strategies like road congestion-aware speed management. This infrastructure can also yield an entertainment platform for sharing videos, playing games, and communicating without any need for internet access. This can considerably decrease the costs of accessing the internet using cellular networks.

Since an ITS is built using a Vehicular Ad-hoc Network (VANET) as one of its components, an ITS's performance depends on the efficiency of the VANET architecture. In this thesis, we study one aspect of VANET architectures, namely routing. Routing refers to the problem of finding paths in networks to transfer a packet from its source to its destination. In particular, we address two problems associated with efficient packet routing in VANETs. The first is

concerned with the construction of a VANET.

Problem 1: A VANET is a network formed with two kinds of nodes, vehicles (that may be mobile) and fixed Roadside Units (RSU), a Roadside Unit is a DSRC (Dedicated Short Range Communications) transceiver that is mounted along a road or pedestrian passageway. Vehicles using onboard wireless transceivers to connect to other vehicles or RSUs. RSUs are connected among themselves using wired or wireless connections that form a *backbone* network. Thus VANETs utilizes two kinds of connections. The first kind is vehicle-to-vehicle(V2V) connections. These are highly dynamic and typically used for exchanging information such as an accident or traffic jam among vehicles. The second kind is vehicle-to-infrastructure(V2I), where the infrastructure is the (static) network of RSUs. In this kind of communication, RSUs collect data from vehicles passing by. Figure 1.1 shows a typical VANET architecture. Note that it is customary to assume that RSU-to-RSU links are cheaper and reliable and are not included in this list.

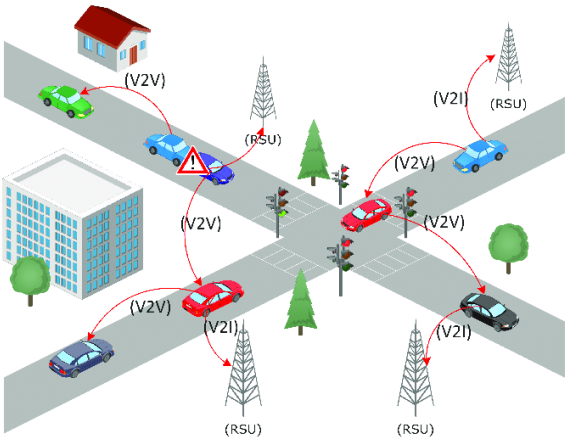


Figure 1.1: VANET Architecture [16]

It is customary to assume that data transfers over V2V connections are less reliable than that using RSUs [19]. On the other hand, providing the extra infrastructure of RSUs (both the RSUs and the connections between them) adds complexity and cost to the network. Thus there is a trade-off between the improvement in network performance due to the RSU infrastructure and the extra cost incurred by it. While one expects that more infrastructure should result in better performance, it is hard to quantify the trade-off intuitively.

The second problem deals with the design of efficient routing algorithms.

Problem 2: By its very nature, VANET topologies are extremely dynamic. Cars move at various speeds but may not move at all times. As well, they may be switched off (parked) and drop out of the VANET. Furthermore, lots of VANETs scenarios occur in urban areas in which there are numerous obstacles, such as buildings, and this degraded the link quality very significantly. Implementing the routing functionality thus becomes far more challenging than not only wired but many other kinds of wireless networks. For example, if the routing algorithm used depends on knowledge of the current topology then an enormous amount of resources must be used to keep every node informed of all changes in network topology. If routing table are not used/maintained then new ideas are needed to route packets. Finally the presence of RSU infrastructure has a big impact on the routing problem.

So without RSU, the problem is simply of routing in VANETs with no RSUs and thus reminiscent of mobile ad hoc networks (MANET). With RSUs available, the problem is to design good routing algorithms given a fixed number of RSUs.

One could combine the two problems and attempt to design good routing algorithms

while minimizing the number of RSUs. This work addresses all the above problems.

We now turn to the literature to get a sense of existing research relevant to these problems.

1.0.1 Routing in VANETs

As mentioned earlier, the network topology in a typical VANET changes rapidly. This makes topology and route maintenance very challenging. Therefore, routing algorithms used in more traditional networks require a relatively steady topology in which nodes barely move. Therefore, these algorithms cannot provide quality of service factors such as throughput, latency, and delay in packet delivery for VANETs.

A generic approach taken to address the difficulty of topology and route maintenance is called position-based routing (PBR). Position-based routing uses vehicles' geographical positions, maps, and road information to route messages[4].

In general, routing methods may be divided into three main categories:

1. **Proactive routing:** In this type of routing, each node has a routing table containing information of every node inside the network. Due to the highly dynamic nature of VANETs, this table has to be updated frequently [21].

2. **Reactive routing:** Instead of regularly updating the position of all the nodes in the network, it only stores the paths which are currently in use in the network, and by this way, it will not only considerably diminish the overhead of proactive routing but also improve the performance of the communication by saving the bandwidth for exchanging messages.

3. **Hybrid routing:** It is a combination of reactive and proactive routing [1].

An essential class of reactive, position-based routing algorithms are called geographical routing. Geographical routing neither establishes the whole path before forwarding data nor stores the position of all nodes. Instead, each node collects the position information of its neighbouring nodes. Then, based on the destination node's position, it selects the next node to forward the data and so on. Researchers have shown that these methods improve the packet delivery delay[7].

The routing problem is also affected by the density of nodes assumed. Naturally, the density of vehicles is higher in urban areas and much lower in rural areas. VANETs seem to be more useful in urban areas, so we will mostly explore the existing geographical routing schemes in urban environments.

We look at some details of position-based algorithms next and evaluate the complexity and applicability of these in VANETs.

Position Based Routing Algorithms

We now delve deeper into greedy approaches for position-based routing. These are based on forwarding a message to the nearest nodes towards a fixed destination. We will also review improved greedy schemes in which nodes consider parameters other than position while selecting the node to forward a packet from neighbouring nodes. PBR methods can fit the high mobility of nodes in VANET, and they also provide the advantage of scalability[11]. These methods can be divided into three main sub-groups: Delay Tolerant Network (DTN),

non-Delay Tolerant Network (non-DTN), and Hybrid [20].

Delay Tolerant Network (DTN) In numerous applications of VANETs, packets become useless after a threshold delay. So, while DTN-based PBR schemes can be used for routing in VANETs, they cannot present acceptable performance in irregular network connections. Delayed information in some of VANETs scenarios not only does not help in traffic management but also can cause an accident and worsen traffic jams.

One of the most popular DTN protocols is Vehicle-Assisted Data Delivery Protocols (VADD), In which vehicles carry the pocket until they reach a junction, then they will forward it to the next vehicle. So, it is evident that these protocols can cause a delay in delivering the packet. However, an existing work utilizes Linear system equations to achieve minimum packet delivery delay based on the density of road, vehicle speed and direction, and distance of the road [20].

Non-Delay Tolerant Network (non-DTN) Non-Delay Tolerant PBR schemes are more efficient in the city environment and densely areas. In these protocols, the source node chooses the next hop for forwarding the message by a greedy approach. The principal rule in selecting the next vehicle is the minimum distance from the destination. This strategy helps to reduce the chance of losing connections.

Beacon Based Non-Overlay Routing Protocols In these methods, each node stores the position information of its neighbour nodes in the destination's direction. Routing is done greedily by forwarding a packet to the neighbour that is closest to destination and

closer to the destination than the current node (thus making progress towards reaching the destination at each step). All greedy position-based routing algorithms suffer from a key problem.

The local optimum problem: The problem is illustrated in Fig.1.2. In which Here x represents the source node, y, w are neighbouring intermediary nodes the destination node is D. Since both neighbouring nodes of x is farther from D than x itself, the greedy routing algorithm fails.

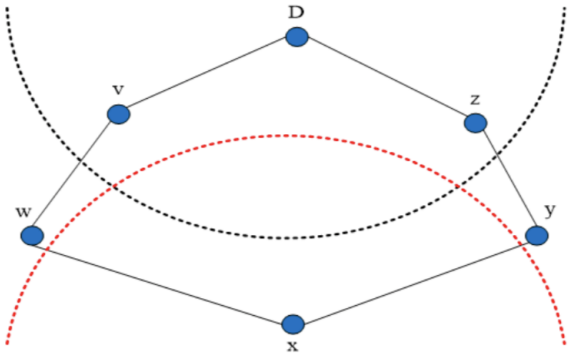
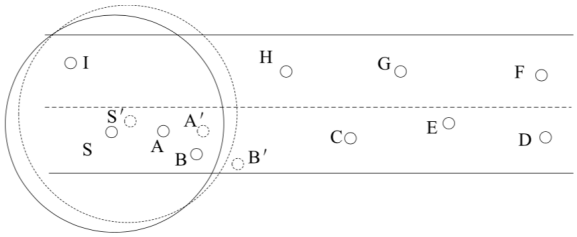


Figure 1.2: No neighbouring node of x is nearer to D than x itself.

Greedy Perimeter stateless routing (GPSR) [9] is one of the most popular solutions to the local optimum problem. In GPSR, each node greedily selects the nearest node that is closer to the destination to forward a packet. When greedy forwarding fails due to the local optimal problem, GPSR will turn into perimeter forwarding, selecting the next node to forward the package based on the right-hand rule [25].

While GPSR addressed the problem of local optima, but it still suffers from two main problems:

1. In the greedy forwarding phase considering the high speed of vehicles selected, the next hope could easily move out of communication range which causes dropping the package.



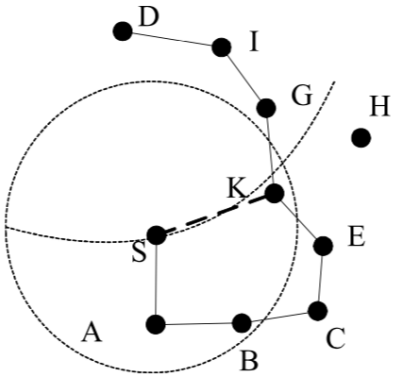
[25]

Figure 1.3: Greedy forwarding approach problem

In Fig. 1.3, S represents the first position of the source node, and the circle shows its communication range; A, B, and I are the source's neighbour nodes, and D is the destination node. Based on GPSR criteria, S chooses B as the next node because it is nearest to D among S's neighbours. Since VANETs consist of moving vehicles, in the next time-step let S', A', and B' be the new position of nodes. Even though all of them have moved in the almost same direction but due to their different speeds B' goes out of the communication range of S', and so node B fails to receive the rest of packets.

2. In perimeter forwarding, the right-hand rule can result in suboptimal paths and increase packet delay, as shown in Fig. 1.4. Here GPSR chooses the path $S \to A \to B \to C \to E \to K \to G$ but the path $S \to A \to K \to G$ is clearly shorter.

Nevertheless, there have been efforts to improve on GPSR. I-GPSR [25] is another routing protocol based on GPSR, which improved the performance by taking the vehicle's speed and orientation into account. I-GPSR made two changes to the original GPSR. First, it



[25]

Figure 1.4: Perimeter forwarding approach drawback

augments greedy forwarding by involving the stability of neighbour nodes, which is defined based on cumulative communication duration, in selecting the next node. Second, in the case of perimeter forwarding, it uses minimum angle with the destination as the criterion for choosing the next node instead of using the right-hand rule. The E-GPSR routing strategy [8] enhances GPSR by considering the available size of node buffers in addition to position and routing parameters in node selection process.

Beacon Based Overlay Routing Protocols In these protocols a path to the destination is computed before the source node forwards a packet. One of the most popular methods of this group is Geographic Source Routing (GSR) [6]. In this method, each node uses beacons to build a neighbour table. Based on this table source node will establish the whole path before sending and the packet's header include all of the nodes on this shortest path. Using map information in the routing process enhances the performance of GSR over GPSR in terms of Packet delivery ratio and delay. However, GSR also fails to solve the local optima

problem.

Seet et al. [15] proposed "Anchor-based Street and Traffic-Aware Routing (A-STAR)" for urban environments. A-STAR also builds the whole forwarding path in the source node but is based on different criteria. It uses vehicular traffic data for city buses for analyzing end-to-end links. It uses Dijkstra's algorithm in establishing the road path, which is called the anchor path. Furthermore, if the local maximum problem happens, the algorithm marks the path as "out-of-service" and rebuilds the forwarding route. A-STAR outperforms GSR and GPSR in the city environment because of its traffic awareness approach. Nevertheless, its performance is highly dependent on the presence of buses. On the other hand, they do not consider network traffic. Most of the buses move in main streets, which makes the bandwidth overcrowded in these paths, while the secondary streets are barely chosen, which may provide better connectivity and less delay[6].

Due to the complexity and overhead of position-based routing as well as the local optimum problem, we focus on a different class of routing algorithms next.

Grid Routing

Broadly speaking, grid-based routing refers to the idea of dividing up a geographical area into grids and forward packets to a grid square rather than a specific car. The following sampling of the literature will provide an idea of different variants of grid-based routing that exist.

GVGrid [17] route packets from a fixed source to all vehicles in a specific grid square. This method assumes that all vehicles use a digital map to geo-locate their position and their

direction.

TTBR [5] is a two-tier routing method in which first the source will send a packet to discover a grid path, then choose the next-hop vehicle using a local map in the following grid.

HHLS [2] is another grid-based method is formed from merging Greedy Perimeter Stateless Routing(GPSR) and Hierarchical Location Service(HLS [10]). This combination leads to a decrease in the location overhead and improving the performance of the network.

CBR [12] is a cluster-based routing method in which cluster formation is based on square grids. This method uses the cluster head for forwarding based on geographical information. CBR does not consider the velocity and direction of vehicles.

HarpiaGrid [3] is also a grid-based method that establishes the sequence of grids with minimum distance between source and destination based on the digital map.

Use of Machine Learning It may not be clear at first glance why machine learning is relevant to our problems. We have argued that computing and maintaining routing tables in real time is very expensive and routing without routing tables has its share of problems. Both these problems can be avoided if we use an alternative idea - that of learning routing tables from history. This can be precomputed offline and used in real-time. Clearly for this approach to work, some conditions must be met by traffic patterns. As well, this approach needs aggregation of cars in some manner, since the movement of individual cars has too much variance. Grid based routing satisfies the latter since the VANET chooses a grid to forward to and not a specific car.

The more traditional machine learning algorithms were categorized into supervised learning,

where the "true" labels were supplied along with the training data and unsupervised learning, where such labels are not used. Naturally one expects supervised learning algorithms to be more accurate in general. However for routing problems, it is not feasible for optimal routes to be computed for the learning algorithm. As well, since the set of all possible paths are too large to be computed efficiently, unsupervised learning is also not applicable. A more recent development, reinforcement learning [18], is more applicable to our problem. This method of learning needs no labelled data or even a large precomputed set of possible solutions. Instead the algorithm interacts with an environment that supplies rewards or penalties for decisions made by the algorithm, and it uses this feedback to learn good solutions. A rough analogy would be a young person learning to play a game like tic-tac-toe well by playing repeatedly and learning from successes and failures.

Among the existing reinforcement learning (RL) algorithms, Q-learning seems the most appropriate for us. RL algorithms are formalized using models called finite Markov decision process (FMDP) (see [18] for definitions and results, and the next chapter for a summary of relevant facts on MDP and RL). In the language of FMDP, Q-learning can identify actions (decisions) optimally for any given FMDP, given infinite exploration time. Informally, this means Q-learning is a principled approach to the use of RL in our work.

The literature on the use of reinforcement learning to VANET routing is very sparse. The most prominent work we found was QGrid [11].

QGrid uses a greedy approach to grid-routing. It first divides a map into a uniform-sized square grid. Then, it uses Q-Learning to compute a path (sequence of grid squares) for every

pair of grid squares. This is done offline, and then later, packets are routed using these paths. QGrid uses a simple greedy strategy to choose a vehicle inside the selected grid square. QGrid suffers from several issues.

1. **Scalability:** QGrid builds path between each pair of grids, and this implies that for an $n \times n$ grid the number of grid square pairs is $\Theta(n^4)$. This follows from the fact that there are n^2 squares and therefore $\binom{n^2}{2}$ possible pairs. So even for a medium sized city, say $10Km \times 10Km$, assuming a grid square size of $500m \times 500m$ as is often done in the literature, we get $\frac{20!}{2!18!} = 190$ different pairs which mean 190 Q-learning tables and 190 sequences of grids. In contrast, the Greater Toronto Area(GTA) has an area of 7,124 km^2, and London has an area is 1,572 km^2. Thus it is clearly infeasible to run QGrid for even moderately large cities.

2. **Network traffic:** QGrid establishes only one fixed path of grids for each pair offline, which results in overloading certain grids while others do not participate in forwarding packets. As shown in Fig.1.5.

3. **Unable to handle changes:** Since, in QGrid, vehicle density and traffic flows do not have a notable impact in training the reinforcement learning agent, like the conventional methods QGrid is susceptible to drastic changes in performance even with small changes in the network parameters such as vehicle's velocity or density.

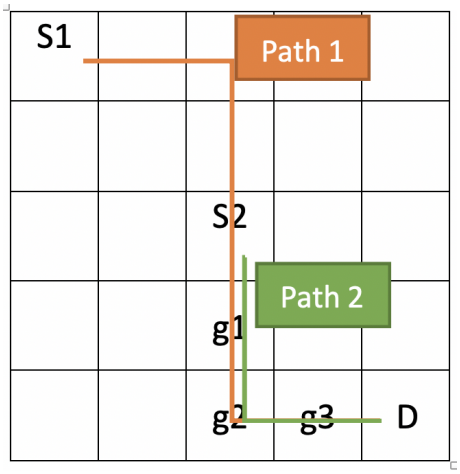


Figure 1.5: An Illustration of grid paths

1.0.2 Our contributions

We provide a solution to the problem of designing an efficient routing algorithm that uses Road Side Unit(RSU) infrastructure. Specifically, we proposed a hybrid routing algorithm ReLeVR (for Reinforcement Learning based VANET routing, pronounced "reliever") that utilizes the backbone network. Our algorithm uses Q-learning to compute paths in the grid. Then it greedily chooses the car closest to the destination or an RSU within the grid square, for forwarding a packet.

Next we consider the problem of minimizing the number of RSUs and positioning them well. We use a traffic-aware allocation method to compute locations of RSUs.

We evaluate the performance of ReLeVR using real traffic patterns and simulation of network communication. The results of our extensive experiments show that ReLeVR outperforms the GPSR, and a Q-Learning-based method, QGRID_G in terms of metrics like the message delivery rate and decreases packet delay. Alternatively, ReLeVR can achieve the

same performance as QTAR,[24], which is also a hybrid method, while using far fewer RSUs.

QTAR puts RSUs in city with fixed distances from each other. Fig.1.6 depicts their proposed architecture. QTAR based on these assumptions selects vehicles using Q-Learning and traffic information to deliver the packets.

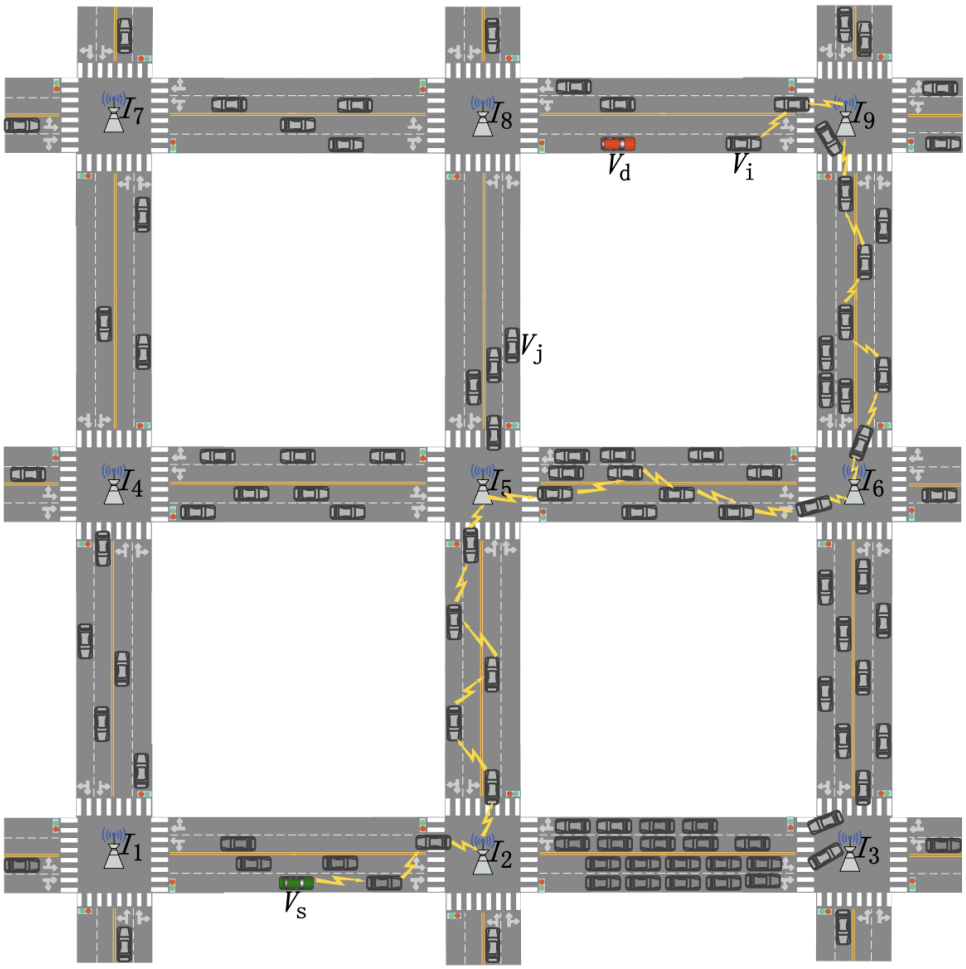


Figure 1.6: An Illustration of QTAR's proposed architecture

1.0.3 Roadmap

The remainder of the thesis is structured as follows.

Chapter 3 elaborates on foundation of reinforcement learning, starting with a brief review on Markov Decision making process which is the base of the learning process in reinforcement learning. then we dive deeper in the mathematical background of the Q-Learning method which we use to solve our routing problem in future chapters.

In the next chapter, we go through a primary analysis of the traffic data and then we apply couple of prepossessing steps on it to while keeping the data accurate manage the time of the simulation and also cover all possible scenarios in a typical day of Antwerp.

Chapter 5 explains the details of our proposed method and how it maps to the routing problem. First, We illustrate the scoring method that we defined to find optimal location for RSUs based on the traffic flow of the city. Then we move to the next step and clarify how we train our reinforcement agent and how ReLeVR forwards the packets based on the established policy.

Finally, in chapter 6 we represent the result of ReLeVR in comparison with Q_Grid and GPSR in different traffic scenarios which could happen in a typical day of a city. Moreover, we compare the performance of the ReLeVR in the absence of RSUs' aid with the other method to evaluate the proposed reward function solely.

Chapter 2

Background

In this chapter, we dive deeper in the theory and mathematical foundation of reinforcement learning to clarify why these learning method is a suitable fit for our problem and how a typical reinforcement learning system model a problem.

First, we briefly reviews Markov decision process which is a mathematical model to make a decision in partly random processes. Markov decision process is the base of learning process of a typical reinforcement learning agent.

2.0.1 Markov Decision Processes

Markov Decision Processes (MDP) is a mathematical framework to model the problems whose results depend on a partly random process or a decision-maker which could be an RL agent. In other words, MDPs are a mathematical tool for dealing with optimization problems which can be considered as a discrete-time stochastic control process.

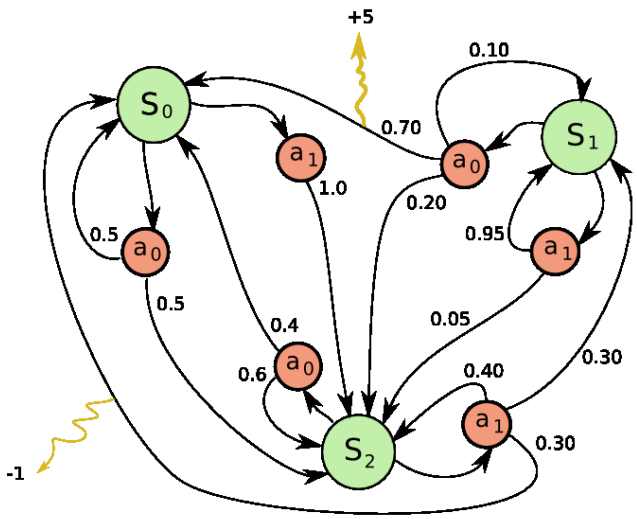


Figure 2.1: Example of a simple MDP with three states (green circles) and two actions (orange circles), with two rewards (orange arrows)[23]

Generally, we define an MDP problem with a tuple (S, A, p, r) which S similar to a set of environment states in RL includes the infinite number of states which are possible in our environment; A is similar to action set in RL, p is a transition probability from state s to state $s^{'}$ after action a. p has been modelled as Transfer Function in RL algorithm. r also could be directly mapped to the reward function in the RL structure. The output of an MDP would be a policy which is a mapping from a state to an action, and its goal is to maximize the reward function. For the finite time horizon MDP, an optimal policy π^* to maximize the expected total reward is defined by Eq.2.1.

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} r_t(s_t, \pi(s_t))\right] \quad where\ a_t = \pi(s_t) \tag{2.1}$$

In the case of infinite-time horizon optimization problems, we can consider optimal discounted total reward or average reward as objective which changes the Eq.2.1 to Eq.2.2 or Eq.2.3 correspondingly.

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma r_t(s_t, \pi(s_t))\right] \quad where\ a_t = \pi(s_t) \tag{2.2}$$

$$\max_{\pi} lim\ inf_{T\to\infty}\ \mathbb{E}\left[\sum_{t=0}^{T} \gamma r_t(s_t, \pi(s_t))\right] \quad where\ a_t = \pi(s_t) \tag{2.3}$$

Where $\gamma \in [0, 1]$ and it is called the discount factor. The amount of discount factor determines the significance of the current amount of reward compared to future amounts of it. If $\gamma = 0$, it will just depend on immediate reward. On the other hand, if it approaches one role of future rewards becomes more important [13].

2.0.2 Reinforcement Learning (RL)

Reinforcement Learning is one of the main categories of machine learning methods alongside supervised learning and unsupervised learning. This method is based on trial-and-error search, which is trying to find the most suitable strategy to map a set of situations to a set of actions to maximize a numerical reward function. The biggest challenges of reinforcement learning are balancing the trade-off between exploration and exploitation and the curse of dimensionality. Reinforcement learning has addressed the later one with parametrized approximators. The major sub-elements of an RL system are as follow:

1. **policy:** It is a mapping from states of the environment to actions to be selected when

the RL agent is in these states. Policy's complication can change in a wide range depending on other elements of the RL model from a simple function or even a look-up table to a statistic function which allocates different probabilities to each of actions or a function that required extensive computation such as a search function to select an action in each state.

2. **Reward signal:** this element specifies the goal of the whole model, and it is the primary basis for modifying the system's policy. RL models have been designed to maximize a numerical signal in each time step which is called reward. The reward signal should be defined as a function that will assign a numerical amount to each pair of actions and states.

3. **Value function:** The reward signal indicates the immediate result, but the main strength of RL is that it considers the long-term consequences of the agent's decisions in defining the model's policy. The value function is the model's elements that evaluate the performance of the model in the long run. A state's value is the total amount of rewards an agent can expect to gather over the future, starting from this state. The agent takes the actions in each state which can provide the highest value, not the highest reward. As a result, it is of high importance to define a value function that could lead the agent to better policy.

4. **Model (optional):** this element is supposed to mimic the environment's behaviour. All of the RL methods do not necessarily require a model for environments [18]. For

instance, in problems that can be described as a finite Markov Decision Process we can skip defining a complicated model for environment's behaviour and instead of it we can describe our environment with the following elements:

(a) **Set of agent's actions (A):** Which include all possible action that the RL agent can take.

(b) **Set of environment states (S):** This consists of all states that our RL agent may be in during the process.

(c) **Transfer Function** $F_t(s_t, a_t)$**:** It gives the next sate in case of taking action in the state [11].

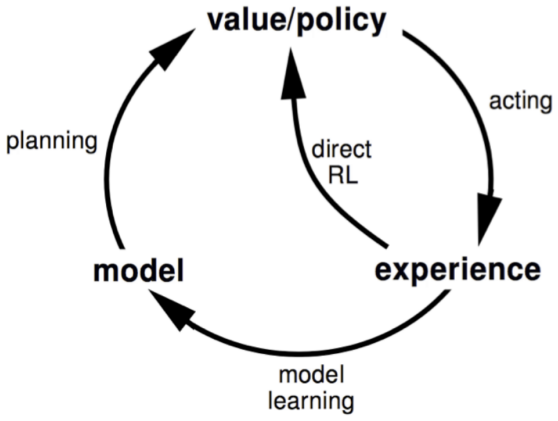


Figure 2.2: An illustration of Reinforcement Learning structure

Q-Learning

Q-Learning is a model-free reinforcement learning method. This method could be a perfect fit for problems that can be modeled as an infinite Markov Decision Process (MDP). Q-learning

could be used to find the optimal policy even for the cases that the agent does not have any prior knowledge about the environment[11].

Like any typical RL problem, in Q-Learning we aim to find optimal policy(π^*) corresponding to state-action pairs. For this purpose, we first define the value function as follow:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{\infty}\gamma r_t(s_t, a_t) \mid s_0 = s\right] = \mathop{\mathbb{E}}_{\pi}\left[r_t(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) \mid s_0 = s\right] \tag{2.4}$$

So,optimal policy can be reached by solving Equation 3.4.

$$V^*(s) = \max_{a_t} \mathop{\mathbb{E}}_{\pi}\left[r_t(s_t, a_t) + \gamma V^{\pi}(s_{t+1})\right] \tag{2.5}$$

In Q-Learning we define a real value named Q-value as follow.

$$Q(s, a) = r_t(s_t, a_t) + \gamma \mathop{\mathbb{E}}_{\pi}\left[V^{\pi}(s_{t+1})\right] \tag{2.6}$$

As a result we can rewrite the optimal value equation as Equation 3.6 Where $Q^*(s, a)$ is optimal Q-value.

$$V^*(s) = \max_{a}\{Q^*(s, a)\} \tag{2.7}$$

Consequently, the optimization problem reduces to finding the optimal amount for Q-value for each pair of (s, a) which can be calculated through an iterative process. Q-value each time will update by following equation.

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha_t[(r_t(s, a) + \gamma \max_{a'} Q_t(s, a') - Q_t(s, a)] \tag{2.8}$$

In which, α_t is the learning rate which determines the importance of new information about Q-value in comparison with our previous Q-value. Similar to other learning methods, the

learning rate can be a constant amount or can be changed dynamically as far as satisfying the following conditions that guarantee convergence of Q-Learning.

1. $\alpha_t \in [0, 1]$
2. $\sum_{t=0}^{\infty} \alpha_t = \infty$
3. $\sum_{t=0}^{\infty} (\alpha_t)^2 < \infty$

The iterative process for updating Q-value continues until either the algorithm reaches a certain number of iterations or none of the Q-values changes for a certain number of iterations. The output of these iterative process will be the optimal policy which maps each state to a specific action and it will be the result of Equation 2.9. According to the fact that Q-values are known real values for each pair of state and action, the Equation 2.9 always has an answer.

$$\pi^*(s) = arg \max_a \{Q^*(s, a)\} \tag{2.9}$$

In addition, as it has been proven in [22] if learning rate conditions were satisfied that the Q-learning algorithm converges to the optimum action-values with probability of one.

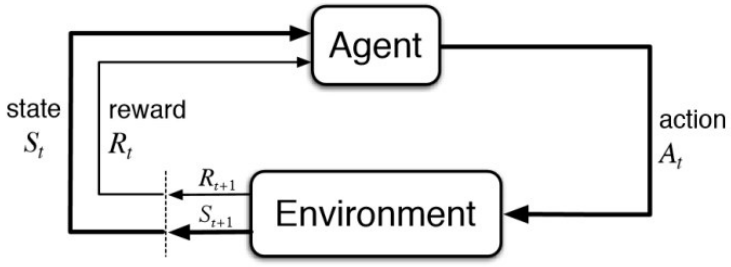


Figure 2.3: An illustration of Q-learning

Chapter 3

Methodology

ReLeVR is consist of two main tier. First tier consists of two step, in first one we find the optimal location for the RSUs based on the regular traffic flow in the city and in the second step we train a reinforcement learning agent to allocate one specific action for the cars in each grid of the map. By this way our agent will be able to establish only one grid path that can route any packet towards its destination.

In the second layer we select a vehicle in suggested next grid by our agent based on a greedy approach.

3.0.1 First Layer: Grid Routing

Locating RSUs

Reviewing the literature, we come across different works that leverage roadside units to address the routing problem in VANETs. A common drawback among these works is that

most of them presume there is a roadside unit located after every few miles, e.g. [24] is a machine learning-based routing method that assumes there are intersections with exact equal distance from each other, and there is a roadside unit at each of these intersections.

This assumption introduces some undesirable attributes to the system; having an RSU at every corner makes it extremely expensive to build. Besides, the goal of VANETs is to lower the load on the backbone network, and installing too many roadside units defeats that purpose.

To address this issue, we first divide the city into neighbourhoods consisting of 25 girds. We then define a fitness function that we use to find the most suitable grid to host an RSU. If the neighbourhood's average traffic is lower than a certain threshold, we do not allocate any RSU to any of the constituting grids, and instead, we force the vehicles in the neighbourhood to either keep and carry the message or forward it to a vehicle closer to the destination. The fitness function is defined as in 5.1.

$$GS_i = NoV_i + \sum_{j=1, j\neq i}^{25} \left(\frac{NoV_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}\right) \tag{3.1}$$

where GS_i refers to the i^{th} grid's score, and NoV_i denotes to the number of vehicles in the i^{th} grid. The denominator in this formula calculates the Euclidean distance between two grids.

The grid that hosts the RSU should have a high average traffic flow. The reason is that a higher number of vehicles translates into a higher number of messages that need routing, and having an RSU closer to these messages decreases the delay of receiving them by the RSU

and the chance of losing the packets, and consequently, the throughput of the whole system would be improved.

Training the Reinforcement Learning Agent

Usually a town has similar traffic pattern in a specific day of week and same time intervals of day which make it possible to train an RL agent to learn this pattern and use to forward the packets towards their destinations. We are going to learn the Q-Value for best routs from each Geographical area (each grid) to one of its nearest RSUs.As evident in Fig. 3.1 there might be some RSUs in proximity of a vehicle, but the RL agent chooses one of the RSUs, based on the traffic flow, to send the packet to. We should note, that the chosen RSU is not necessarily the closest one to the source.

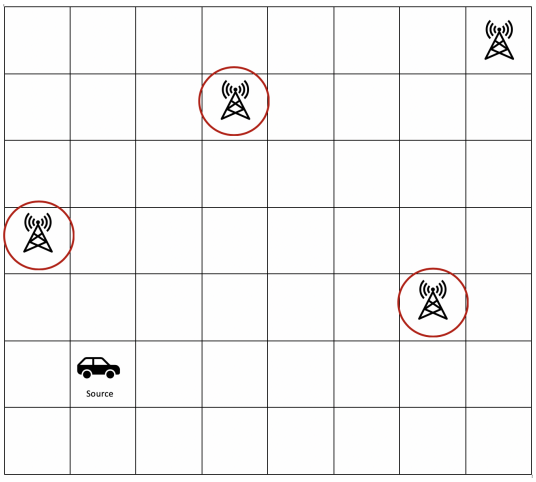


Figure 3.1: nearest RSUs to the source

The Q-learning algorithm provides us with a routing policy which consists of an action for each grid in the gridded map of the city.

The action will be chosen from: Actions = {up, up-right, up-left, down, down-right, down-left, left , right, keep carry}

The another set that should be defined for a RL system is possible state set. As we explained in previous chapter, the city map has divided to equal-sized grids. We consider the set of these grids as our state set.

Next important parameter of RL system i sits reward function. Finding the optimal route will is based on training a Q-Learning model whose reward function is related to average number of vehicles in the grids that route consist of, total number of grids in the path and most common direction of vehicle in each grid. This reward function is defined in Eq.5.4.

In Eq.5.4, $NoV_{next\ grid,t}$ denotes the number of vehicles in next grid in the established path when time equals to {t} and $NoV_{Total,t}$ represent the number of all of vehicles in the city in the time {t}.

$$reward(time = t) = \begin{cases} 200 & If\ the\ destination\ is\ inside\ the\ grid; \\ -40 & Action = Carry\ the\ packet\ by\ the\ source; \\ -35 * \frac{NoV_{next\ grid,t}}{NoV_{Total,t}} & Action = Forward\ the\ packet; \\ -1000 & If\ there\ is\ no\ vehicle\ in\ the\ next\ grid. \end{cases} \tag{3.2}$$

These parameter will be used to find optimal policy(π^*) corresponding to state-action pairs. For this purpose, we first define the value function as follow:

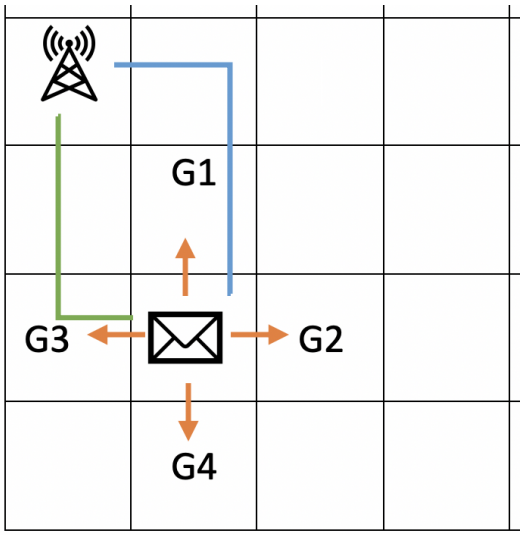


Figure 3.2: An illustration of the Q-Learning model, Orange arrows show the possible actions to take from the cell that message is now, the blue and green lines depict two possible route to the RSU with minimum number of grids their values will differ with each other based on number and direction of vehicles in the grids that they include

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\pi}[\sum_{t=0}^{\infty} \gamma r_t(s_t, a_t) \mid s_0 = s] = \mathop{\mathbb{E}}_{\pi}[r_t(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) \mid s_0 = s] \tag{3.3}$$

So, optimal policy can be reached by solving Equation 3.4.

$$V^{*}(s) = \max_{a_t} \mathop{\mathbb{E}}_{\pi}[r_t(s_t, a_t) + \gamma V^{\pi}(s_{t+1})] \tag{3.4}$$

In Q-Learning we define a real value named Q-value as follow.

$$Q(s, a) = r_t(s_t, a_t) + \gamma \mathop{\mathbb{E}}_{\pi}[V^{\pi}(s_{t+1})] \tag{3.5}$$

As a result we can rewrite the optimal value equation as Equation 3.6 Where $Q^{*}(s, a)$ is

optimal Q-value.

$$V^*(s) = \max_a \{Q^*(s,a)\} \tag{3.6}$$

Consequently, the optimization problem reduces to finding the optimal amount for Q-value for each pair of (s, a) which can be calculated through an iterative process. Q-value each time will update by following equation.

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha_t[(r_t(s,a) + \gamma \max_{a'} Q_t(s,a') - Q_t(s,a)] \tag{3.7}$$

In which, α_t is the learning rate which determines the importance of new information about Q-value in comparison with our previous Q-value. Similar to other learning methods, the learning rate can be a constant amount or can be changed dynamically as far as satisfying the following conditions that guarantee convergence of Q-Learning. In our case after running a hyper-parameter optimization we assigned the α as 0.4.

Also, γ the coefficient which determines the importance of future amounts of Q-value in updating the policy. With the same approach, γ is set to 0.5.

We have optimized the reinforcement learning model's parameters based on following metrics:

1. **Delivery ratio:** It is the portion of packets that are received in the destination out of total number of packets generated.

2. **Delay:** The average time that it takes to deliver packets successfully.

3.0.2 Second Layer: vehicle selection

After the direction is determined by the Q-learning agent, we need to forward the packet to the next receiver. The duty of the algorithm's second layer is to choose the next receiver and we propose a greedy approach to address this task. We choose the vehicle in the destination grid which is closest to the destination or an RSU and forward the packet to that vehicle.

We should note, even when that Q-learning outputs a "keep" command we follow the same rule, i.e. we send the packet to the vehicle which is closest to the destination or an RSU in the current grid. RL agent might output a "keep" command based on the traffic flow in the neighboring grids, e.g. it is possible that no vehicle is available in the destination grid at the moment, so there is no choice rather than keeping the packet in the current grid.

Chapter 4

Traffic Data and Message Generation

We have conducted our simulation based on real-life traffic data of Antwerp during a typical business day [14]. Antwerp is the second largest city of Belgium with area of 204.5 km^2 and a population of almost 500,000.

4.0.1 Data Preprocessing

The data set includes vehicle speed and position, as well as global indicators of transport reported every 60 seconds for 24 hours. Since usually forwarding a data packet take much less than one minute, we calculated estimation of vehicle's positions and their speed between each two time steps with interpolation. Consequently, final data set contains information of each car with 1 sample per 5 second frequency.

The original reported indicators for each vehicle in each time step are as follow.

1. **Time:** Time flag (in seconds)

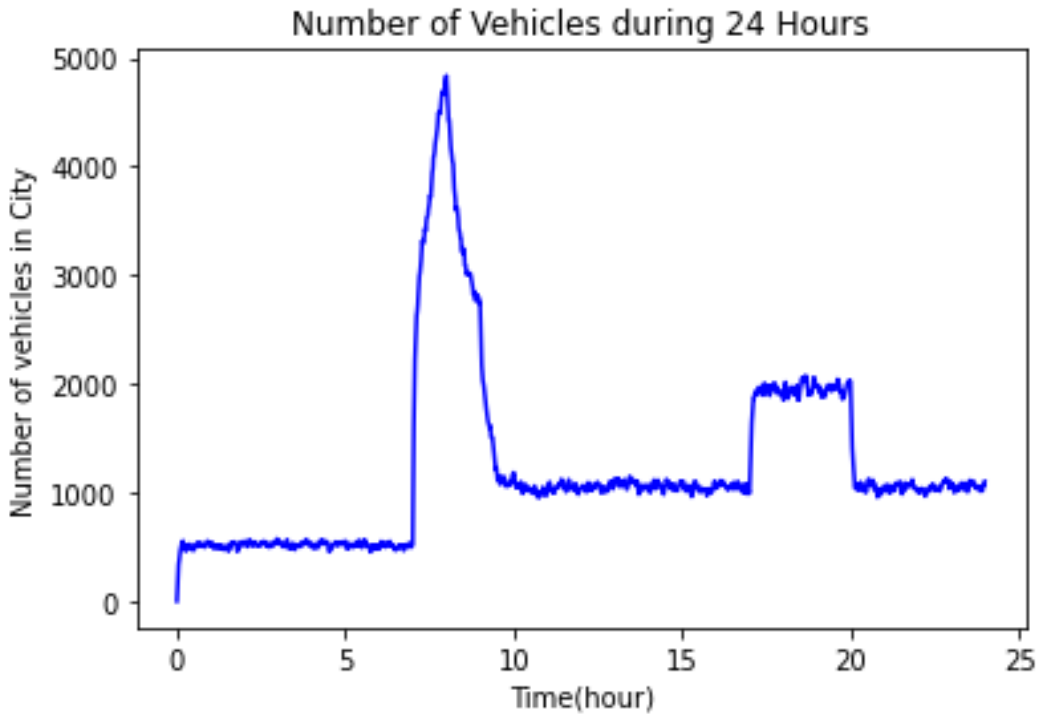


Figure 4.1: Number of vehicles in city during 24 hours based on the original data set

2. **VehicleID:** Unique identifier for each vehicle

3. **Speed:** Current vehicle speed (km/h)

4. **Dist_front:** Distance from preceding vehicle (meters)

5. **xCoord:** Longitude of current vehicle position

6. **yCoord:** Latitude of current vehicle position

7. **SectionID:** unique identifier for each road network link

8. **Av_Sect_Speed:** Current mean speed of all vehicles in section

9. **Sect_Density:** Number of vehicles currently in section

In order to make the data more feasible to have in real life without forcing any extra computational load to each vehicle, we have removed Dist_front, SectionID, Av_Sect_Speed, Sect_Density from data.

Moreover, our proposed method is based on grid routing so we needed to divide our map to grids. The city's area has been split to 868 grids (28*31). Grids are square shape and their length has been considered approximately 500 m.Also, as you can see in Fig.4.2 traffic load varies with similar pattern all over the city.

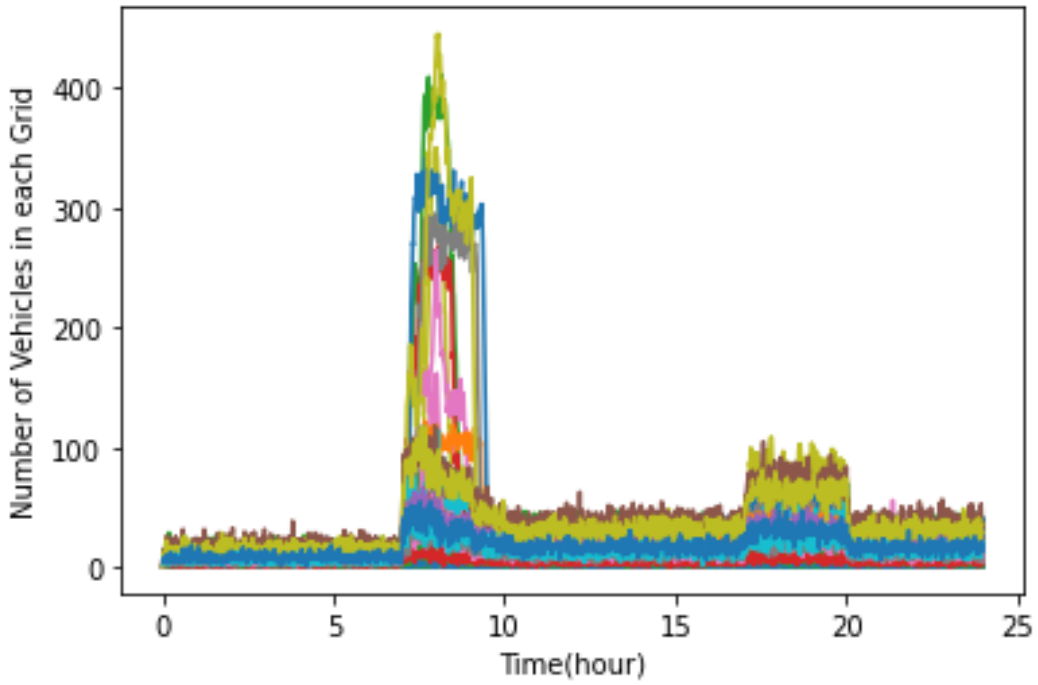


Figure 4.2: Number of vehicles per grid during 24 hours based on the original data set

Speed of Vehicles varies in range 0 to 146 km/h and as it is shown in Fig.4.3 it has reverse relation with number of vehicles in the city.

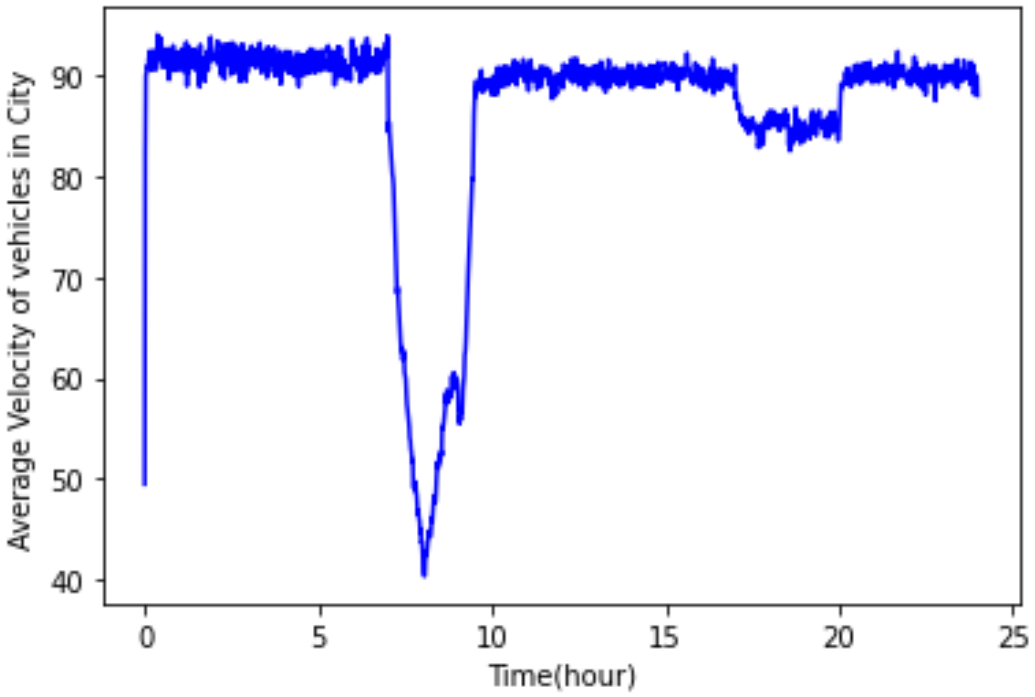


Figure 4.3: Average of vehicles' speeds during 24 hours based on the original data set

Considering 24 hours interval is too long to simulate, we selected 8 hours of day which contain various traffic loads and average speeds to evaluate reliability of our method (From 8:20am until 4:40pm). As mentioned before, the frequency of sampling has increased by interpolation and Fig.4.4 depicted the number of vehicles in city based on modified data which has been used for simulation.

The selected interval of the data still include maximum and minimum velocities and Fig.4.5 represent the average speed of vehicles in the selected interval.

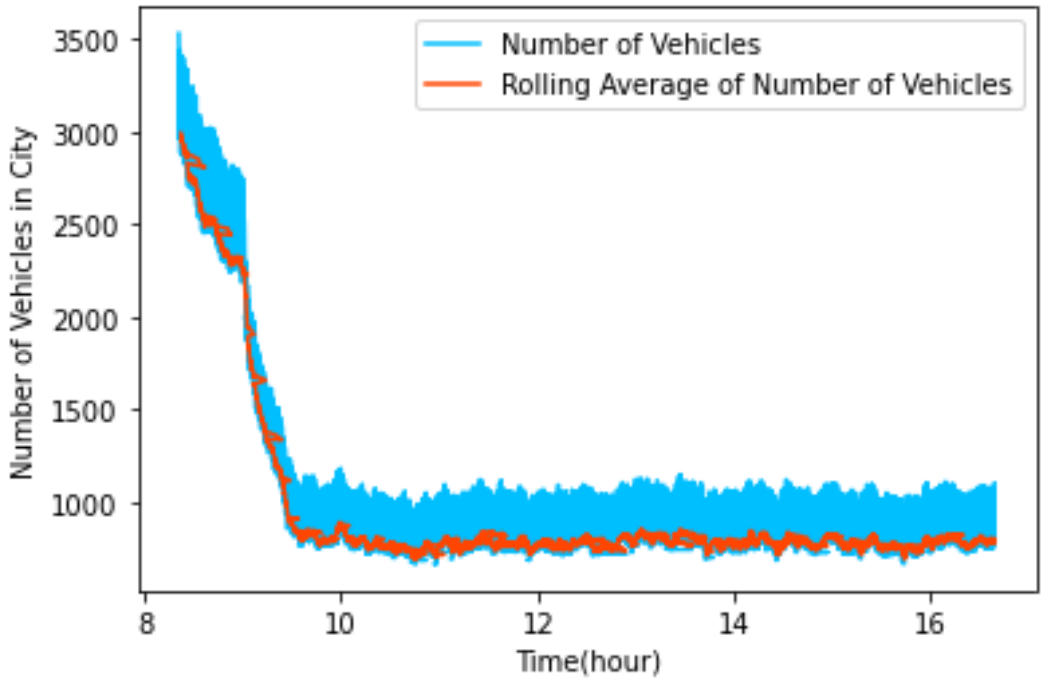


Figure 4.4: Average of vehicles' speeds during 8 hours based on the modified data set

4.0.2 Message Generation

Number of messages per time step have been generated Poisson-distributed with $\lambda = 10$. In total, 59726 messages have been produced. Fig.4.6 shows the number of generated messages during simulation. As expected, the average number of messages per time step equals to almost 10 (accurately 9.9943). Additionally, in Fig.4.7 histogram of number of generated messages per time step has been depicted.

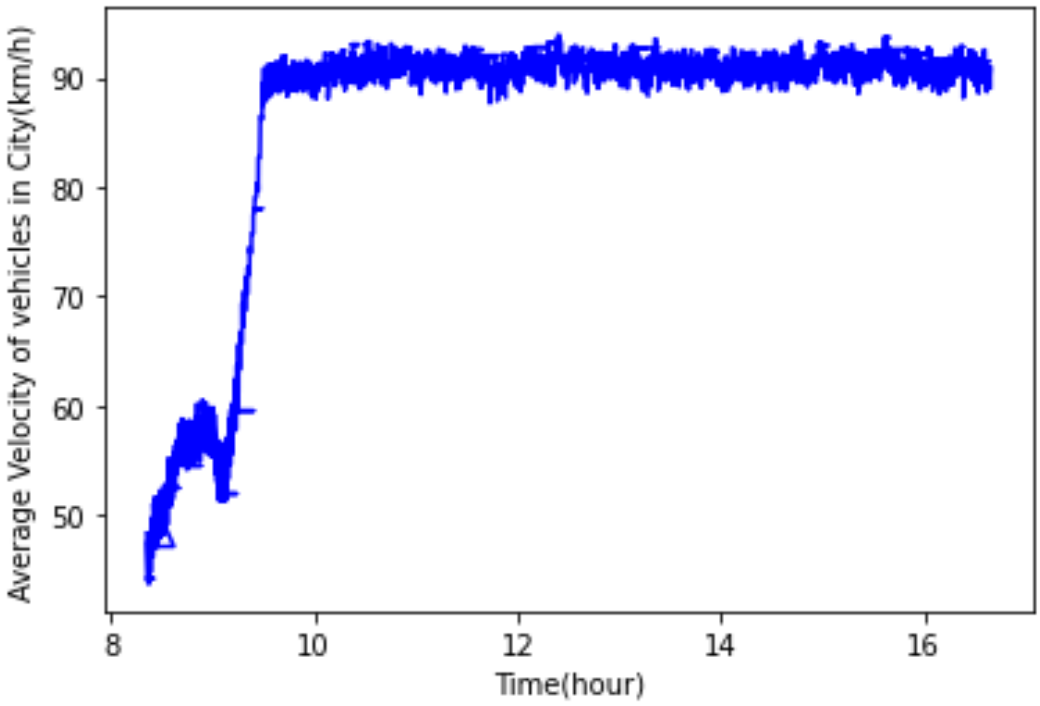


Figure 4.5: Average of vehicles' speeds during 8 hours based on the modified data set

Vehicle which is source of message has been selected randomly. We forward the message to a fixed location and destination has been chosen randomly as well. Fig.4.8 depicts the distance between the source and destination. Average distance is 6807.15m and standard deviation of it is equal to 3928.86 which shows we cover forwarding messages for both short and long distances in the simulation.

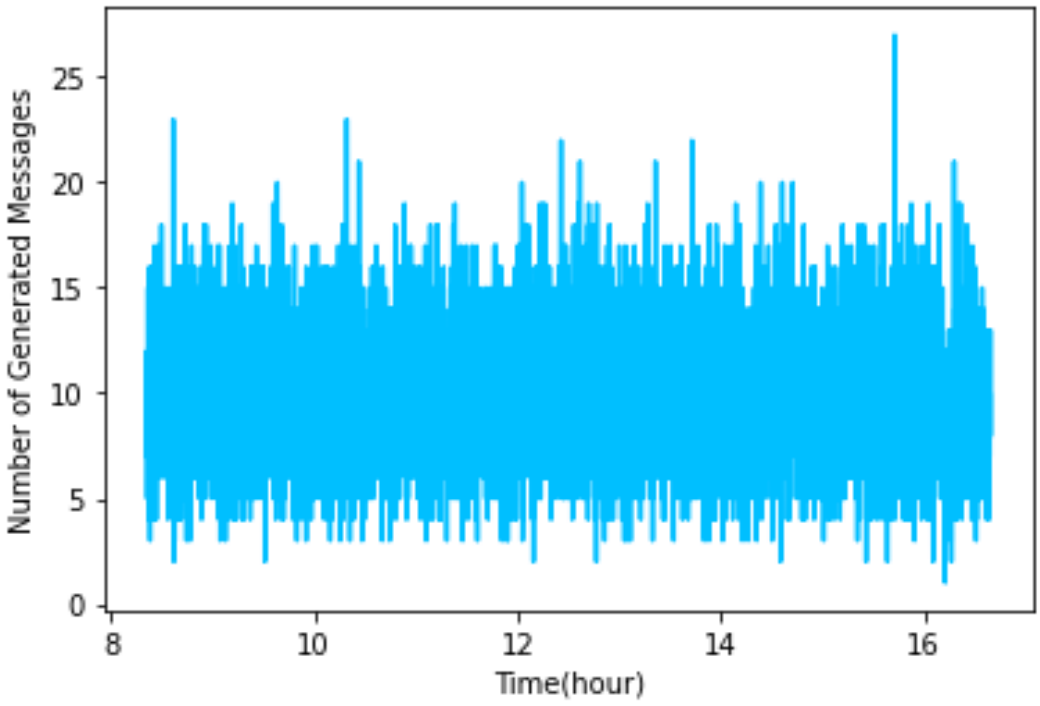


Figure 4.6: Number of generated messages during 8 hours

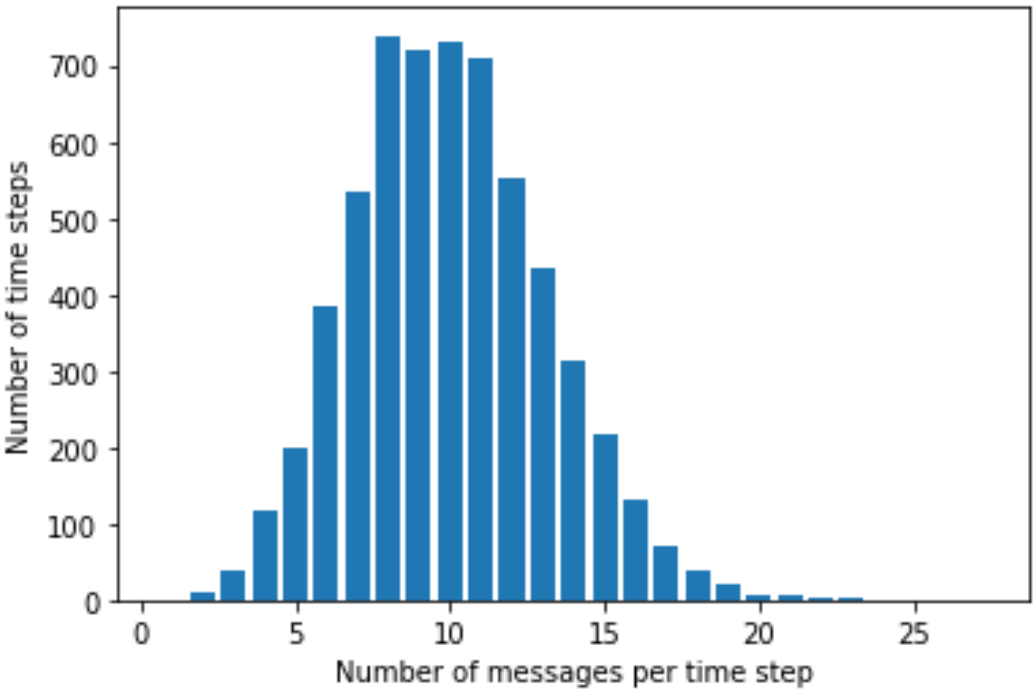


Figure 4.7: Histogram of number of generated message per time step

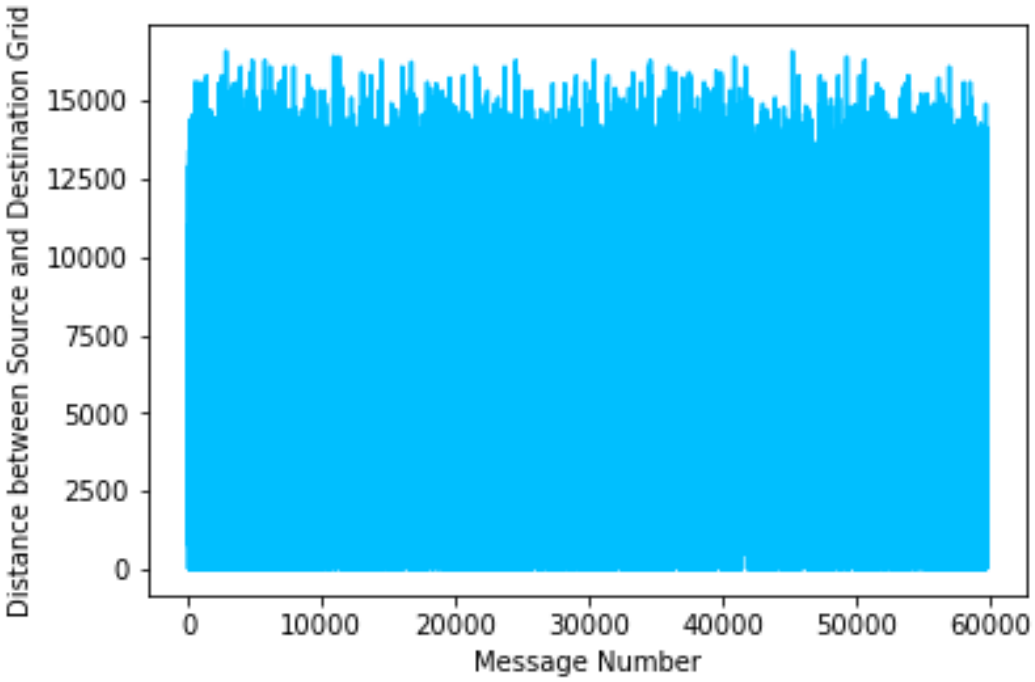


Figure 4.8: Distance between source and destination by message's ID

Chapter 5

Simulation Result

We conducted the simulation for the real-time traffic data of Antwerp between 8:20 am to 4:40 pm. Our result has been compared with two other routing protocols, one traditional geographic-based routing(GPSR) and one Q-Learning-based method(QGrid).

5.0.1 Compared Routing Protocols

1. **Greedy Perimeter Stateless Routing(GPSR):** As we stated in the Introduction chapter, GPSR is a geographic-based routing that greedily forwards messages to the next nearest vehicle to the destination.

2. **QGrid:** It is a novel hierarchical routing for Vehicular Ad Hoc Networks(VANETs), which conduct routing in two steps; first, establish a grid path by Q-Learning and then utilize it to forward packets greedily. This method has been discussed more in the Introduction chapter. QGrid, as it stated in [11], has outperformed the following

well-known routing methods:

(a) **HarpiaGrid:** It is a grid-based routing which establishes a path of grid based on a digital map [3].

(b) **CBS_like:** [26], [27] proposed a routing backbone for VANETs based on bus systems. The defined bus system is community-based, where communities are formed based on the contact relationship of bus lines. Since taxies do not follow a predefined route, they cannot constitute a community. In the original CBS work, ten copies of each message are used in the routing protocol; however, for the sake of comparison [11] has only considered one copy of each message during the routing. This explains why this method is called CBS_like instead of CBS.

3. **Q-learning-based Traffic-Aware Routing (QTAR):** The authors of this work [24] proposed a Q-learning-based algorithm to learn the road segment traffic information. In order to deliver each packet to its destination, QTAR ensures the routing path benefits from multiple highly reliable connections through vehicles between intersections.[24] assumes there are intersections with equal distance between them, and there is an RSU located at each intersection to process the forwarding tasks. This seemed like an unrealistic assumption; thus, we compare the number of RSUs that cite QTAR has assumed to the number of RSUs in our work required to obtain a similar delivery ratio.

5.0.2 Evaluation Metrics

We used the following metrics to evaluate the results:

1. **Delivery ratio:** The portion of packets that are received in the destination out of the total number of packets generated.

2. **Delay:** The average time that it takes to deliver packets successfully.

3. **Number of hops:** The average number of vehicles that carry the packet before it reaches the destination.

5.0.3 Evaluation Scenarios

The desired method should perform well in different traffic loads, vehicle speeds and distance between source and destination.

1. **Various traffic loads:** During a typical day, each city experiences rush hours. Considering Antwerp's daily traffic flows, we defined rush hour as times that there are more than 1000 vehicles in the city. 22.95% of the simulation time is categorized as rush hours.

   The rest of the days have a regular traffic flow. For Antwerp, there are less than 1000 cars traversing the city during these hours, the average number of cars on the street is 785.44 during regular hours, and 77.05% of simulation time is spent in this condition.

2. **Various average velocities:** In mobile networks, the velocity of nodes can play a crucial role in routing performance. So we evaluate the routing methods in various

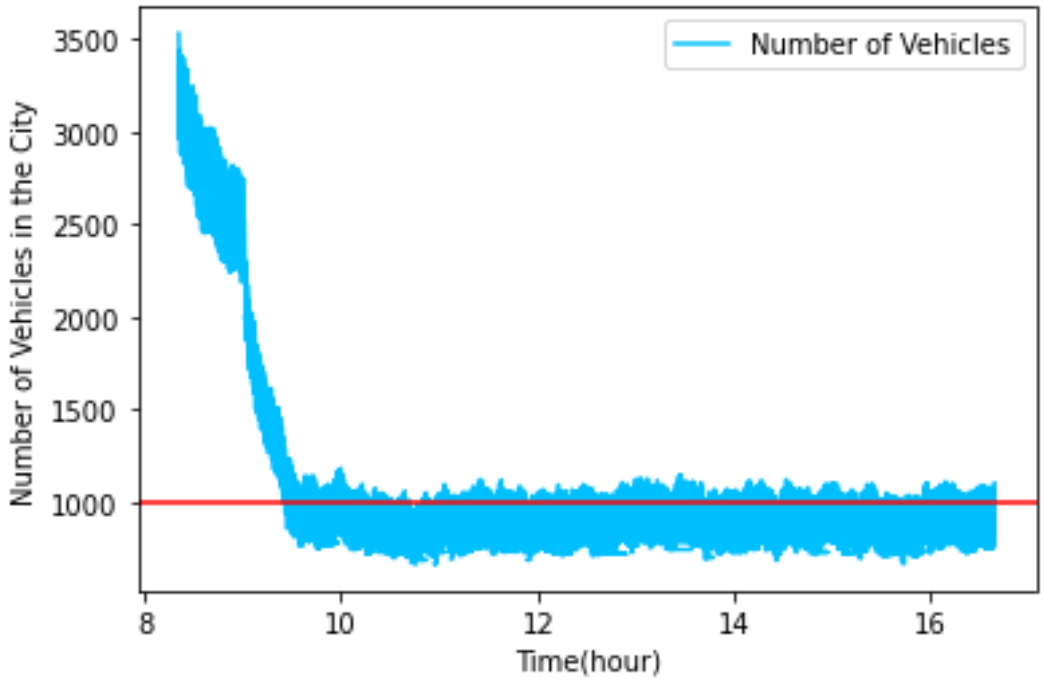


Figure 5.1: Number of vehicles in the city during the simulation and the threshold for classifying different traffic loads' scenarios

scenarios with various average vehicle's speed range.

Regarding Antwerp's traffic data on a typical day, we consider the times in which the average speed of cars in the city is higher than 65 km/h as the high-speed interval. In the simulation time interval, 88% of the time belongs to this category. Intervals with average speed lower than 65 km/h have been classified as the low-speed interval (usually city is experiencing traffic jam in these intervals).

3. **Various distances between source and destination:** Another factor that has a significant impact on the performance of routing protocol is the distance between source and destination. Also, the ability to perform well in long-distances indicates the routing

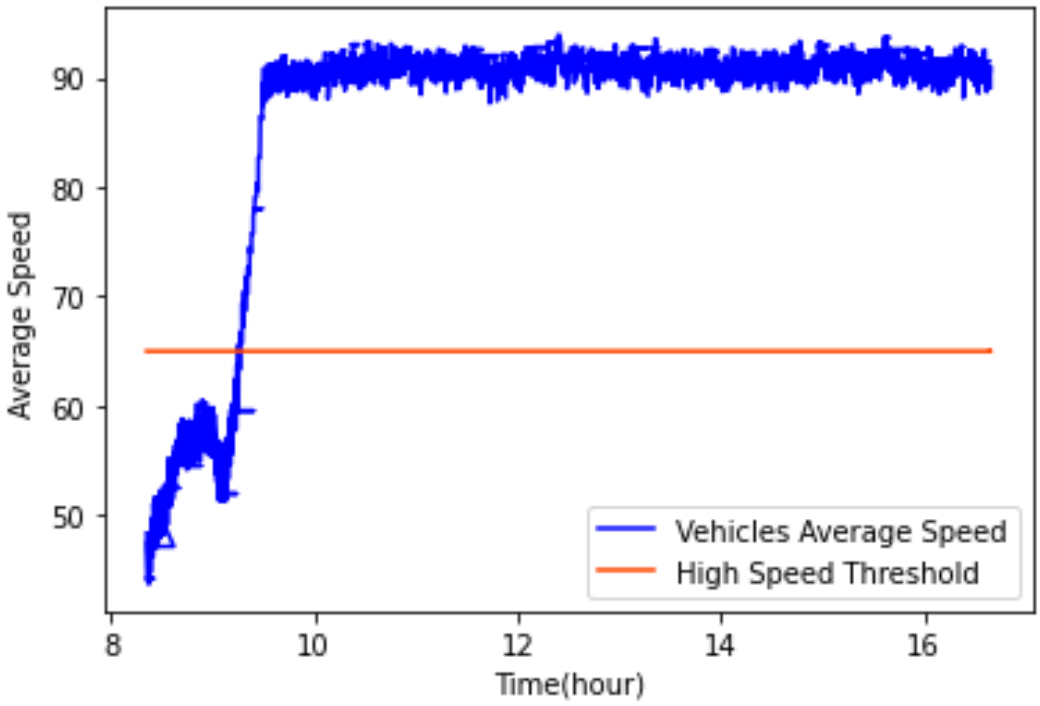


Figure 5.2: Average speed of vehicles in city during simulation and the threshold for classifying different speed range scenarios

protocol's scale-ability and makes it more practical.

5.0.4 Simulation's Assumptions

In VANETs most of messages are traffic-related which means we wants to deliver a message about the traffic to all the vehicles in a specific geographical location. Our system is designed to route these kinds of packets. So, it routs the messages generated by moving sources to a specific geographical location. Also, we assume that each vehicle broadcasts its location for vehicles in its radio range.

In wireless system radio range is usually between 1000m to 2000m and we assumed the

radio range in our system equal to 1414m to make sure that each vehicle can forward a packet to any of vehicles in its neighbouring grids.

5.0.5 Road Side Units (RSUs)

As we stated in the previous chapter, most of routing protocols in VANETs that benefits from RSUs suffer from putting extensive number of RSUs in the network which results in an overwhelming extra cost. To address this issue, we first divide the city into neighbourhoods consisting of 25 girds. We then define a fitness function that we use to find the most suitable grid to host an RSU. If the neighbourhood's average traffic is lower than a certain threshold, we do not allocate any RSU to any of the constituting grids, and instead, we force the vehicles in the neighbourhood to either keep and carry the message or forward it to a vehicle closer to the destination. The fitness function is defined as in 5.1.

$$GS_i = NoV_i + \sum_{j=1, j\neq i}^{25} \left(\frac{NoV_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}\right) \tag{5.1}$$

where GS_i refers to the i^{th} grid's score, and NoV_i denotes to the number of vehicles in the i^{th} grid. The denominator in this formula calculates the Euclidean distance between two grids.

The grid that hosts the RSU should have a high average traffic flow. The reason is that a higher number of vehicles translates into a higher number of messages that need routing, and having an RSU closer to these messages decreases the delay of receiving them by the RSU and the chance of losing the packets, and consequently, the throughput of the whole system

would be improved.

QTAR [24] method has used 16 RSUs for area of 9.241 km^2 assuming the transmission range is equal to 355 m. We are using 23 RSUs to cover a area of 204.5 km^2 and we considered the transmission range 1400 m. To have a fair comparison, we estimate the number of RSUs QTAR need for our case would be 89 (you can find the calculation in Eq.5.2).

$$Needed\ RSUs\ based\ on\ QTAR = 16 \times \frac{355}{1400} \times \frac{204.5}{9.241} = 89.625 \tag{5.2}$$

5.0.6 Simulation Results

Delivery ratio for different values of Time To Live(TTL) is depicted in Fig.5.3. As we expect, the delivery ratio will gradually increase with enhancing the TTL since it will provide more time for messages to reach the destination. We can see that the proposed method has a better delivery ratio overall, but the difference in performance becomes more significant after a reasonable distance which indicates better scale-ability of ReLeVR.
Fig5.3 shows the delivery ratio for TTL < 25. Since the proposed method and QGrid_G both use the traffic data to select the next grid and then in the selected grid greedily chose the next vehicle, they perform better than GPSR, which misses any information about traffic flow and just forwards the packet to the nearest neighbour to destination.

The reason that our proposed method is performing better than QGrid_G mainly stem from the definition of our reward functions. As you may recall the QGrid_G reward function is defined as follow Eq.5.0.6.

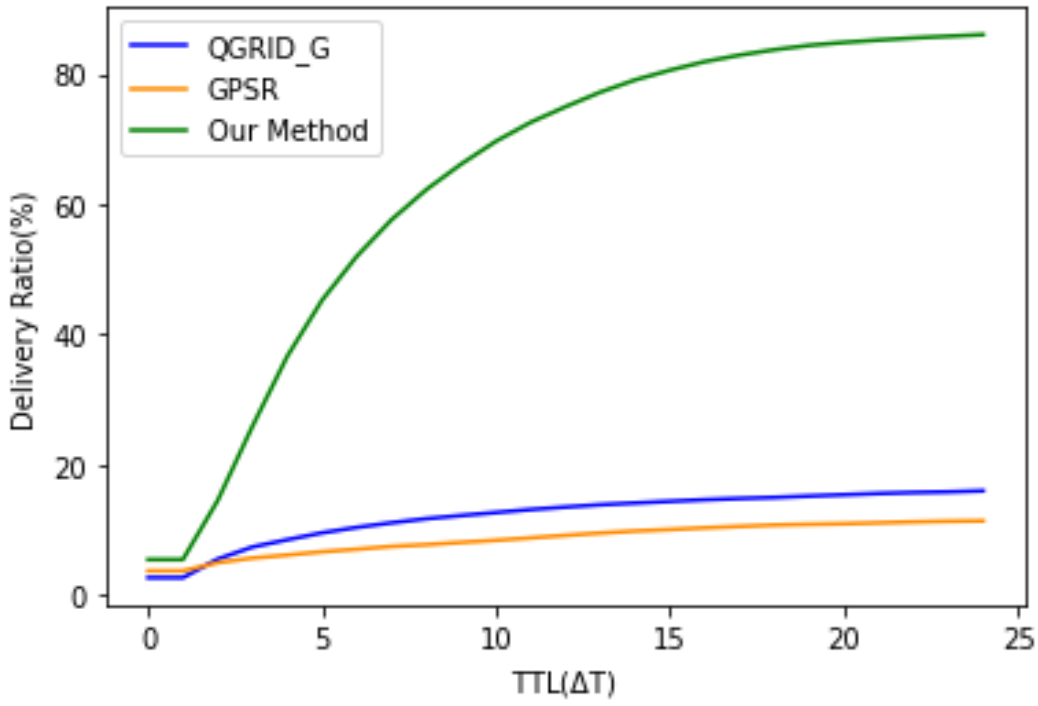


Figure 5.3: Delivery ratio of packets for different TTLs

$$\text{reward} = \begin{cases} 100 & \textit{if the destination is inside the grid}; \\ 0 & \textit{otherwise}. \end{cases} \tag{5.3}$$

In which, the only way that the reinforcement learning agent receives feedback is by reaching the destination. Increasing the size of the map will cause a drastic rise in the number of series of grids that do not include the destination; As a result, as the size of our map grows, it becomes harder for the reinforcement learning agent to search the possible paths to find the optimal path or even find a path to the destination in the reasonable number of episodes.

, On the other hand, our reward function Eq.5.4 is designed to encourage the reinforcement learning agent to forward the packet through most populated neighbour grids even when it does not have any information about which path can lead it to the destination. In Eq.5.4, $NoV_{next\ grid,t}$ denotes the number of vehicles in next grid in the established path when time equals to {t} and $NoV_{Total,t}$ represent the number of all of vehicles in the city in the time {t}.

$$reward(time = t) = \begin{cases} 200 & If\ the\ destination\ is\ inside\ the\ grid; \\ -40 & Action = Carry\ the\ packet\ by\ the\ source; \\ -35 * \frac{NoV_{next\ grid,t}}{NoV_{Total,t}} & Action = Forward\ the\ packet; \\ -1000 & If\ there\ is\ no\ vehicle\ in\ the\ next\ grid. \end{cases} \tag{5.4}$$

In this way, we can avoid dropping a packet as long as possible, and we will increase the chance of finding the destination by the agent by limiting the number of possible paths to the most crowded ones. Furthermore, rewarding the agent for forwarding the packet through most populated grids to find a destination is a reasonable assumption; In reality, we want to forward a packet to another destination in the road network and following the crowd implicitly means following the roads which are usually all connected.

Moreover, it seems to use the help of the infrastructure is unavoidable to maintain an acceptable performance in real-size cities; On the other hand, ut we tried to minimize the number of RSUs.

Another observation which worth mentioning is as depicted in Fig.5.4 and Fig.5.5, after a

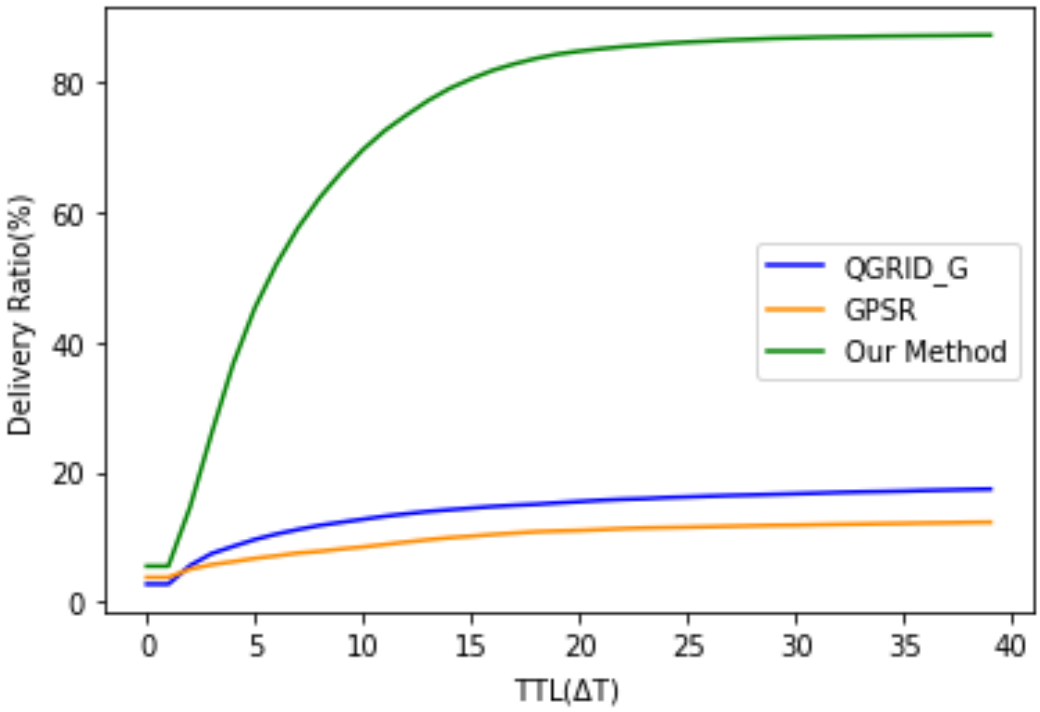


Figure 5.4: Delivery ratio of packets for different TTLs

while, even increasing acceptable TTL does not cause a significant change in the delivery ratio, and it is safe to assume that other packets have been lost.

The next evaluation metric for routing protocol is the number of vehicles that have carried the packet before it reaches the destination, which we call the number of hops. As we can see in Fig.5.6, most of the delivered message has been delivered with a small number of hops, which means transferring a packet between the bigger number of cars chance of packet delivery will decrease.

Another essential evaluation metric to discuss is a delay. Fig.5.7 shows the percentage of messages that have specific amounts of delay for the messages that have reached their destination. Based on the result, ReLeVR delivers 50.94% of all messages with delay smaller

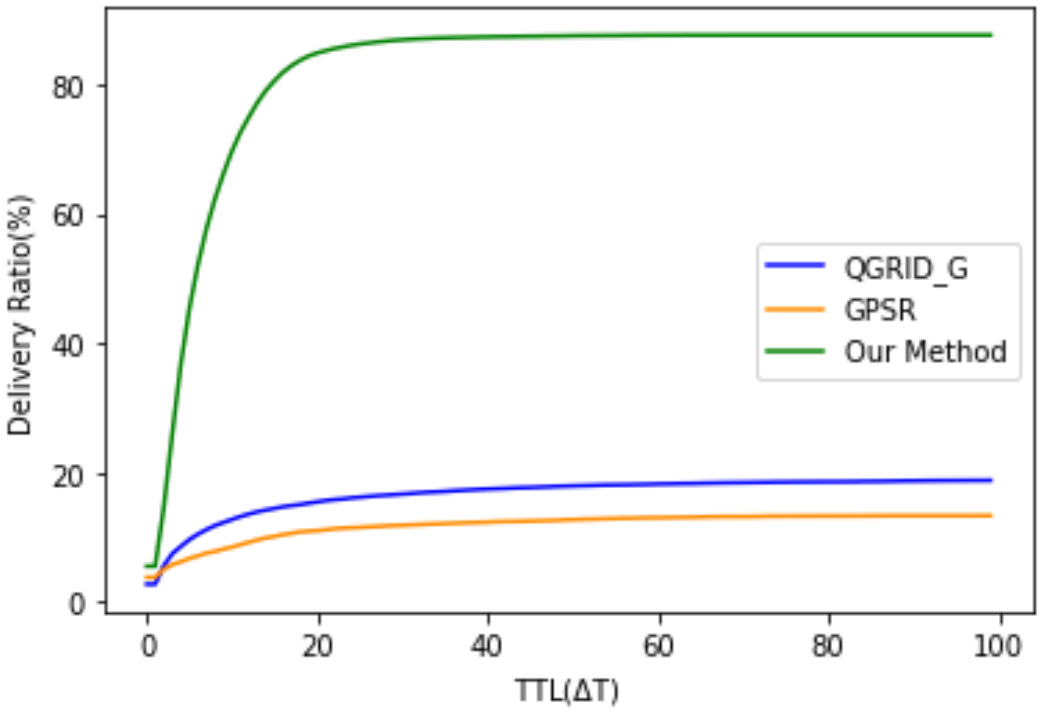


Figure 5.5: Delivery ratio of packets for different TTLs

than six time-step, while QGrid_G only delivers 10.45% of messages in less than six time-step and GPSR 7.11% of messages.

We put RSUs to use in forwarding the messages in long distances, so it was predictable that we are going to have more minor delays in long-distance packet forwarding in comparison with the other two methods, but since we deliver a more significant number of messages with less than one time-step delay, which means that the packet has not gone through the RSUs. We will discuss the delivery ratio by distance later with more details, but the delay charts are another evidence of our protocol performing well both long and short distances.
Another observation worth mentioning is QGrid_G did not perform as well as GPSR for delivering messages with less than one time-step delivery that can be explained by the fact

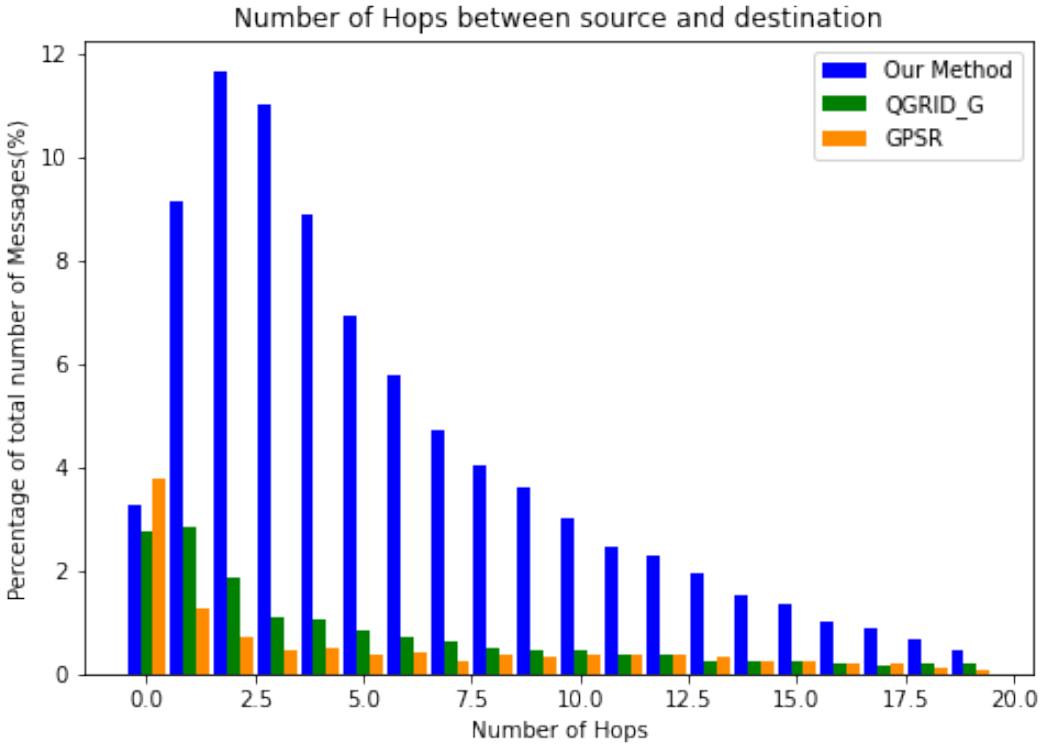


Figure 5.6: Histogram of number hops between source and destination

that QGrid_G does not define the keeping packet or forwarding it inside the grid as possible actions. It will add additional forwarding if the source and destination are less than one grid far from each other.

, On the other hand, we consider the carrying packet by the source or forwarding it inside the grid as a possible action for reinforcement learning agent, and GPSR does not force any restriction based on grid divisions. Consequently, GPSR and ReLeVR perform better for message forwarding in distances less than one grid width.

But this will not affect packet deliveries for longer distances and as a result QGrid_G gain a better performance in comparison with GPSR afterward. However, as we explained

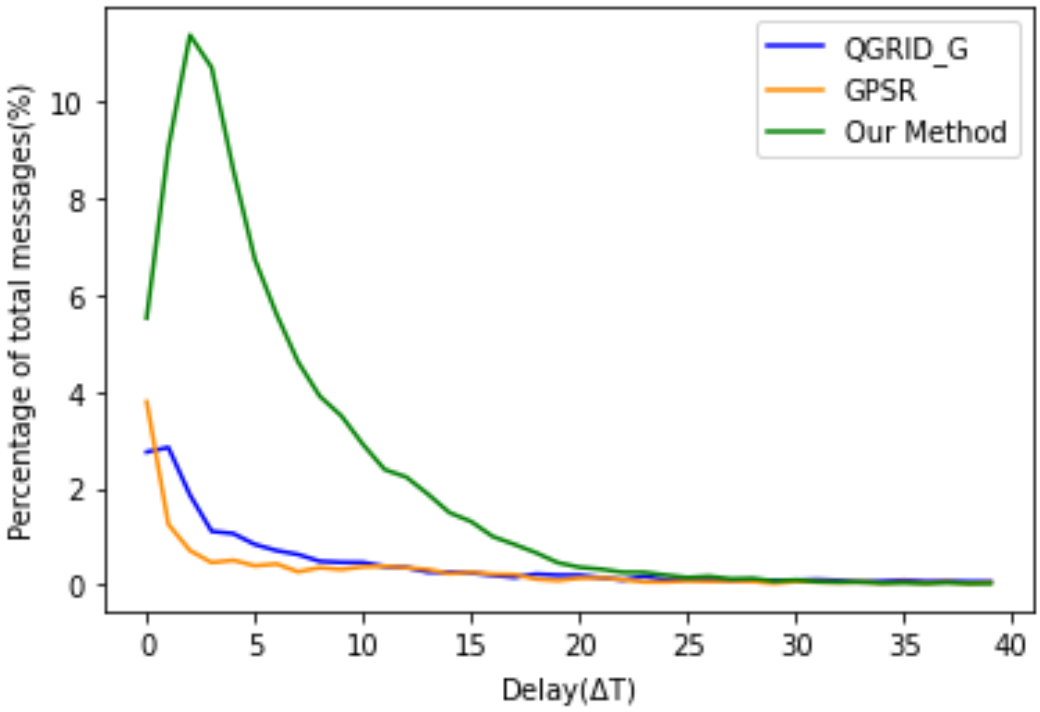


Figure 5.7: Percentage of message that have been received with a delay between 0-40 time-steps

before combination of QGrid_G's reward function and its lack of backbone of infrastructures like RSUs will cause its poorer performance in long distances in comparison with ReLeVR. Fig.5.8 also confirm our conclusion from Fig.5.6 and Fig.5.5 that packets if do not reach the destination in reasonable time most probably never will reach the destination.

Various Traffic Load Scenarios

Considering that we use vehicles to forward packets in vehicular networks, the number of vehicles in cities plays a crucial role in the performance of routing protocols. Fig.5.9 depicts the total delivery ratio of the routing methods, which we are investigating, for both states of

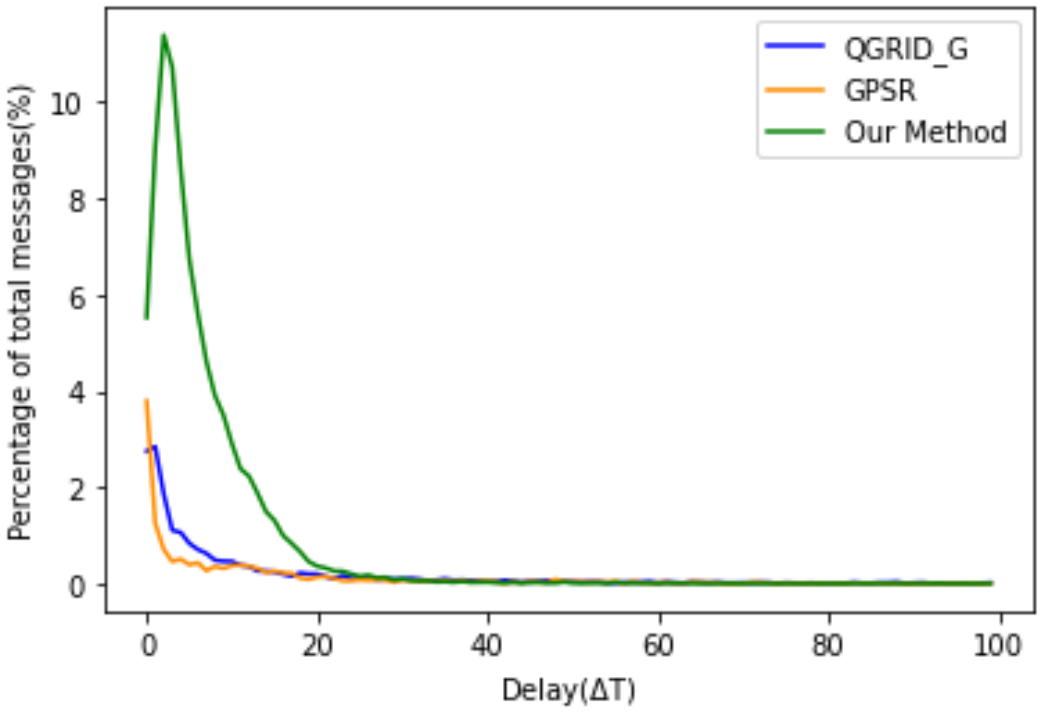


Figure 5.8: Percentage of message that have been received with a delay between 0-100 time-step

traffic loads.

Since in VANETs we are using vehicles to forward packets, any growth in the number of vehicles can improve the performance. The Q-learning-based methods in both states perform better than GPSR, and our proposed system outperformed QGrid_G as well.

However, all of the methods demonstrate a higher delivery ratio in heavy traffic loads in comparison with their average delivery ratio during the whole time of simulation; one of the critical features of a routing protocol is to eliminate the adverse effect of environmental changes such as traffic load as much as possible. The Table.5.1 shows how much each of the methods tolerates traffic load alteration. In this table, change is defined as 5.5. in which DR

refers to delivery ratio.

$$Change\ with\ traffic\ load = \frac{DR_{normal\ traffic} - DR_{rush\ hour}}{DR_{rush\ hour}} \tag{5.5}$$

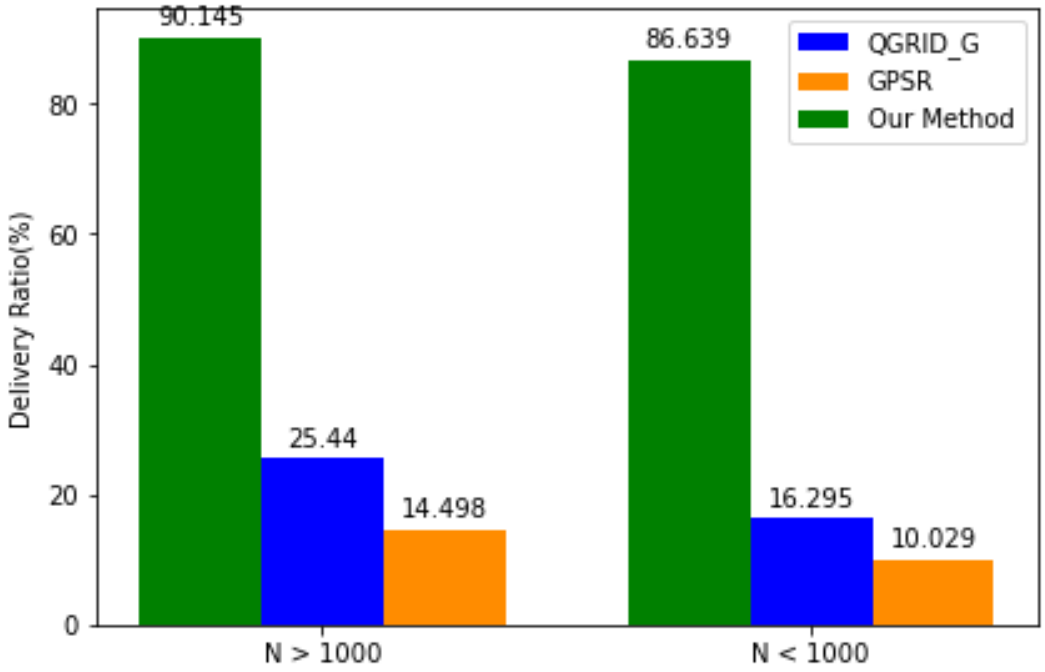


Figure 5.9: Delivery ratio for various traffic load

As it illustrated in 5.10 Q-Learning based methods generally have decreased the average delay for packets in comparison with GPSR, But ReLeVR even outperformed the QGrid_G method. On the other hand, since ReLeVRs benefits from backbone infrastructure in forwarding the packets which result in lower delays it is not fair comparison in some aspects. In order to make the comparison more reasonable we compared delay based on distance between source and destination in Fig.5.22 and also delay without getting help from RSUs(Forwarding in a same area) in Fig.5.24.

Method	Change with traffic load
GPSR	-30.82%
QGrid_G	-35.94%
ReLeVR	**-3.88%**

Table 5.1: Delivery ratio changes because of altering traffic load

However, we believe the considerable improvement in the delivery ratio, which is depicted in Fig.5.9 and a significant decrease in average delay, which is shown Fig.5.10 while using a much smaller number of RSUs in the network in comparison with well known RSU-based methods like QTAR justify the logic of comparison even without the scenarios in Fig.5.24 and Fig.5.22.

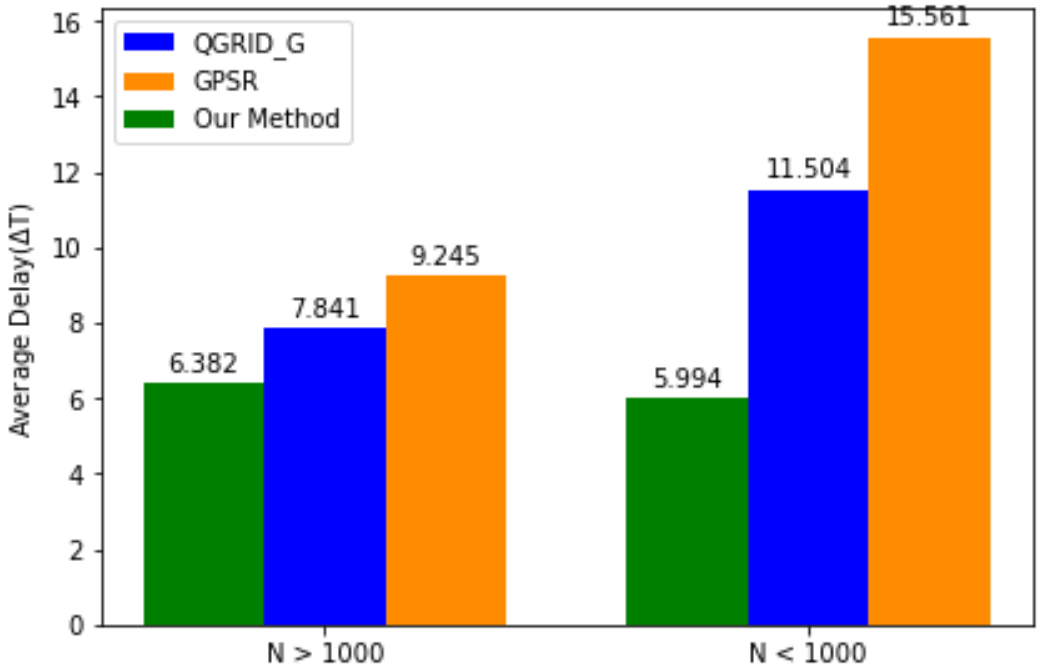


Figure 5.10: Average delay for various traffic load

Methods	Scenario	P-value	Result
QGrid_G,GPSR	N<1000	**0.247**	QGrid_G does not have significantly lower delay
ReLeVR,GPSR	N<1000	**0.0011**	ReLeVR has significantly lower delay
ReLeVR,QGrid_G	N<1000	**0.0067**	ReLeVR has significantly lower delay
QGrid_G,GPSR	N>1000	**0.0009**	QGrid_G has significantly lower delay
ReLeVR,GPSR	N>1000	**0.0002**	ReLeVR has significantly lower delay
ReLeVR,QGrid_G	N>1000	**0.0022**	ReLeVR has significantly lower delay

Table 5.2: Statistical analysis of affect of altering traffic load on delay

Scenario 1: Rush hours (more than 1000 cars in the city): The delivery ratio for the times that there are more than 1000 vehicles in city are as it has been shown in Fig.5.11.

By comparing Fig.5.11 and Fig.5.5 we can conclude that generally routing protocols provide better delivery ratio in crowded areas. So, all of the methods deliver higher percentage of messages during rush hours. Moreover, Fig.5.12 shows that the average delay is also lower during rush hours in comparison to hours with normal traffic.

As Fig.5.12 shows for GPSR and QGrid_G higher number of vehicles in city enhances the percentage of messages that have been delivered in their first forwarding. However, it does not have a significant impact on ReLeVR since our agent gives priority to sending the packets towards RSUs and it does not search its neighbor grids for the destination, unless the destination is in the same direction as the nearest RSU. Our proposed method has delivered a significantly larger number of messages with a lower average delay in total.

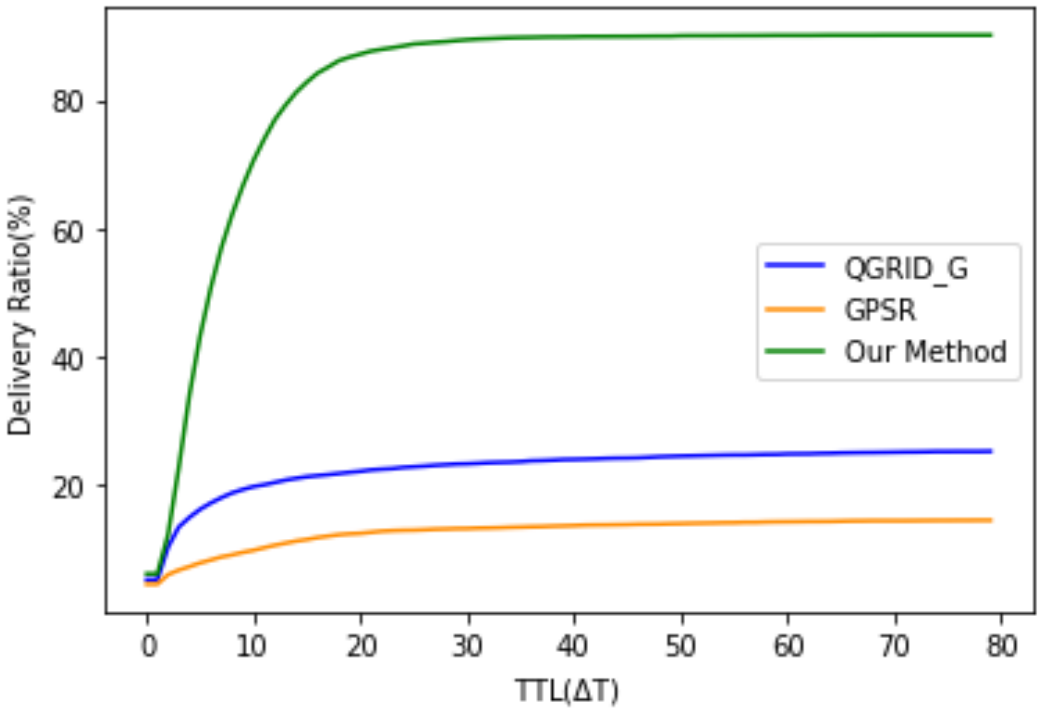


Figure 5.11: Delivery ratio of packets for different TTLs in rush hours

Scenario 2: less than 1000 cars in the city Since in more than 77% of a usual working day the city experiences this scenario, performance of routing methods in this density plays a key role in whole system's reliability. As you can see in in Fig.5.13, when we have small number of vehicles in the city, our proposed method shows significantly better delivery in comparison with GPSR and QGrid_G.

In this scenario, the difference between our method's delivery ratio and other methods' delivery ratio is bigger than other scenarios; In fact, we can say that the most distinguishing feature for this system design is its outstanding performance for low density areas.
This advantage mostly stems from the fact that our reward function gives density of cars in neighbor's grid a key role in establishing path for forwarding whereas QGrid_G grants the

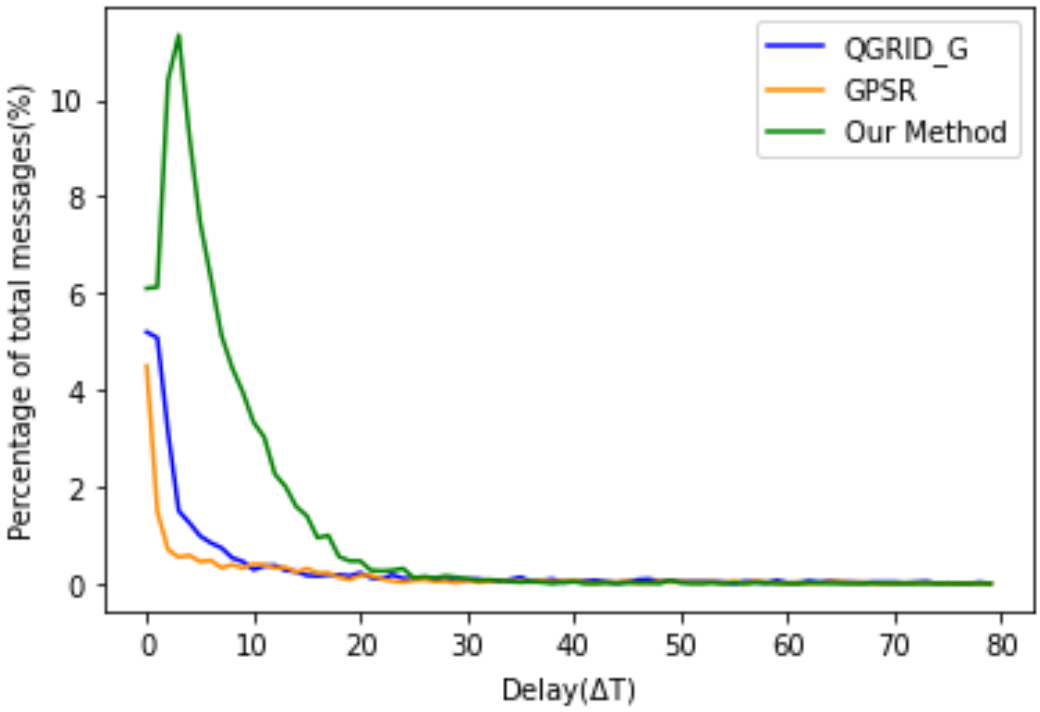


Figure 5.12: Percentage of message that have been received with a delay between 0-80 time-steps in rush hours

car density a small impact on updating coefficient of Q-value and GPSR does not consider the car density at all.

The whole concept of VANETs depends on existence of the vehicles to carry the packet consequently bringing density to consideration avoids losing the packet by forwarding it into empty grids or to the area's where cars are far from each other or they are getting far from each other. Finally, keeping the packet inside the network at the end will increase the chance of delivering it successfully.

Also, we should mention that changing the system design and bring limited number of RSUs in the network design facilitate the whole message forwarding while keeping the cost of

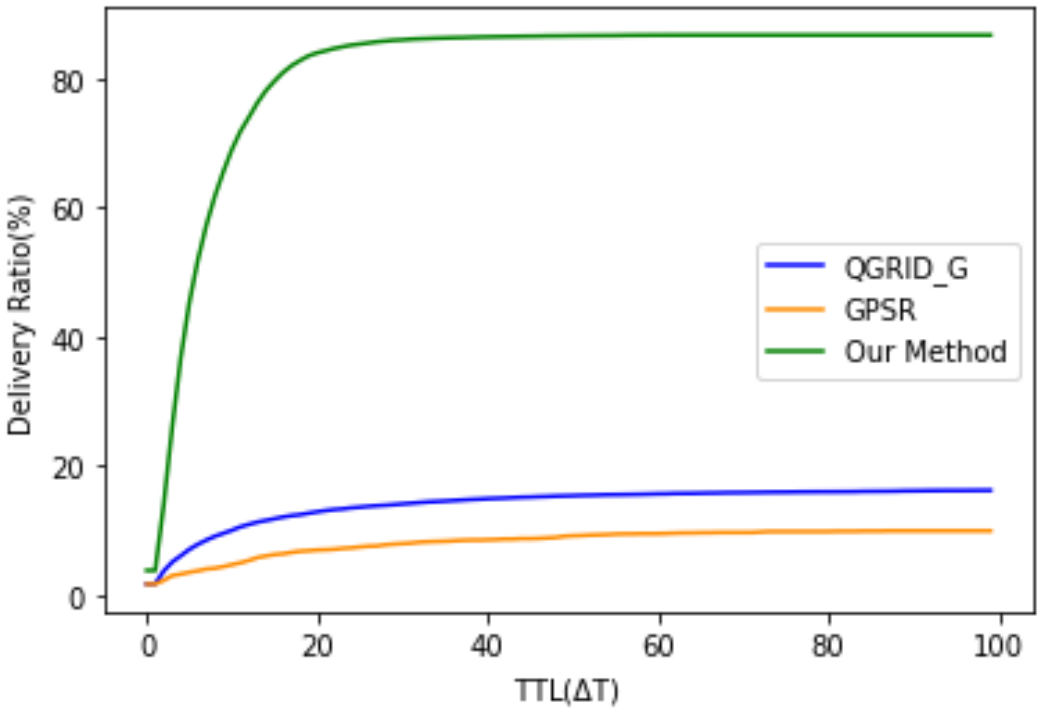


Figure 5.13: Delivery ratio for low traffic load

using back bone infrastructure as low as possible.

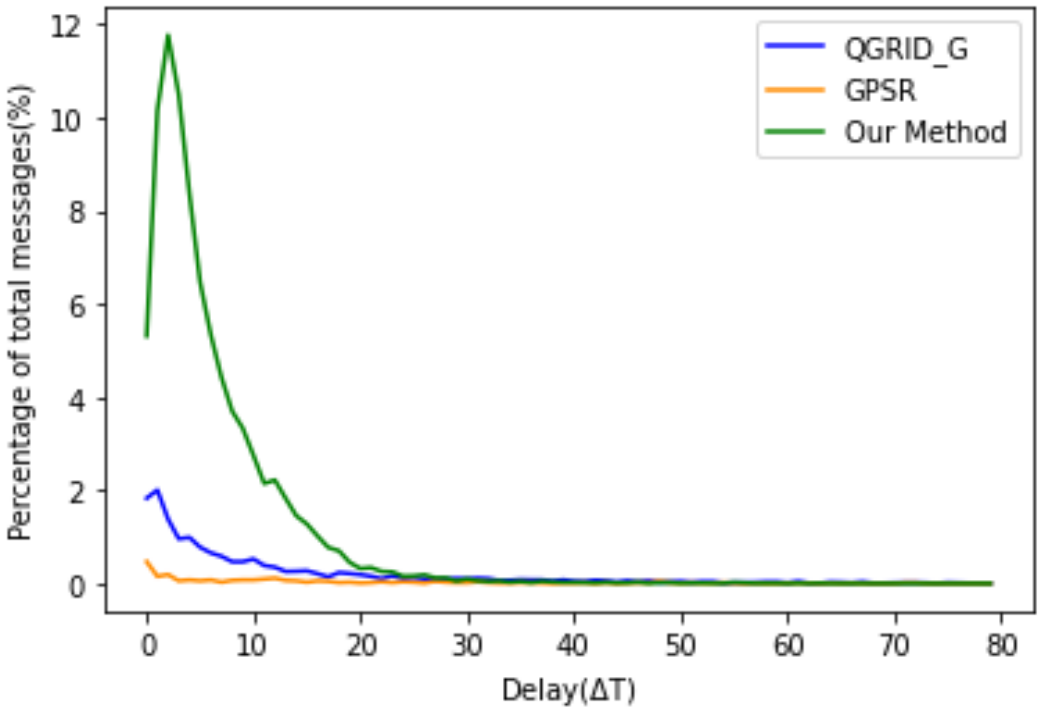


Figure 5.14: Percentage of messages that has been delivered with various delays in light traffic load

Various Velocity's Scenarios

One of effective factors in a routing problem is the degree of mobility in that network. In VANETs, mobility is inevitable and with increase of the average velocity of cars in the network the difficulty of successfully delivery of packets with rise.

In this scenario we evaluate performance of ReLeVR in comparison with other method while network experience various levels of dynamicity. In Fig.5.15 you can find delivery ratio for a highly dynamic scenario with an average speed higher than 65km/h and another scenario with lower degree of mobility in which average speed of cars does not exceed 65km/h. As

you can see however ReLeVR is affected by change able to keep an almost same performance regardless of changes in average velocity of vehicles.

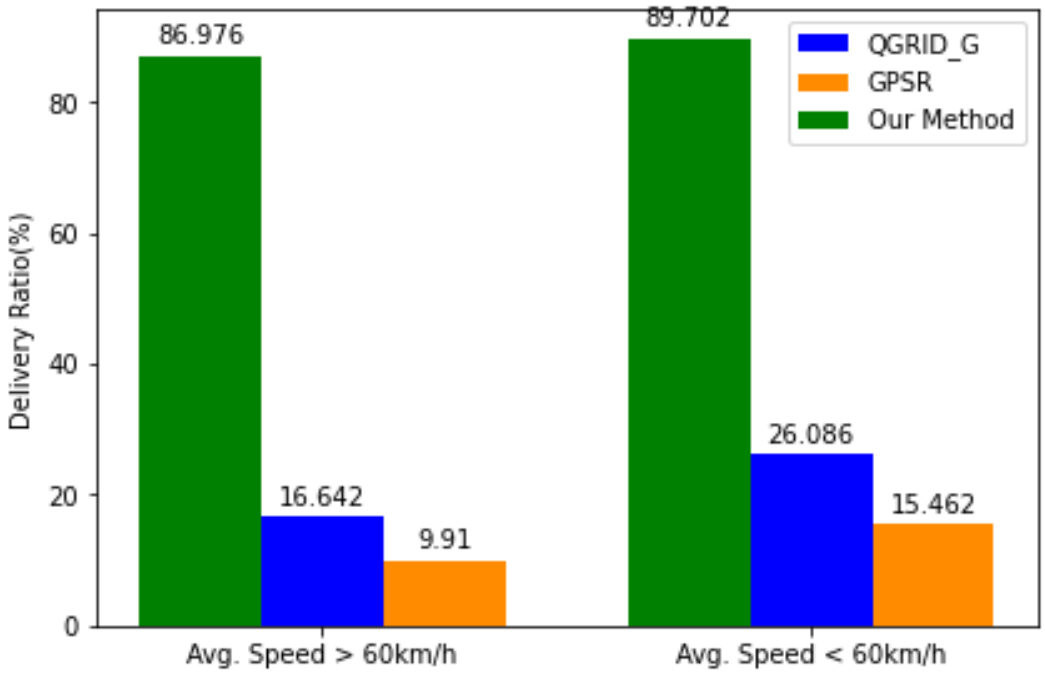


Figure 5.15: Delivery ratio for different average velocity scenarios

Fig.5.16 shows that the proposed method provides a lower average delay in both scenarios comparing to the QGrid_G and GPSR. Interestingly, we can also observe that while a change in velocity drastically influences the average delay in those two methods, it does not have a meaningful effect on our method's performance.

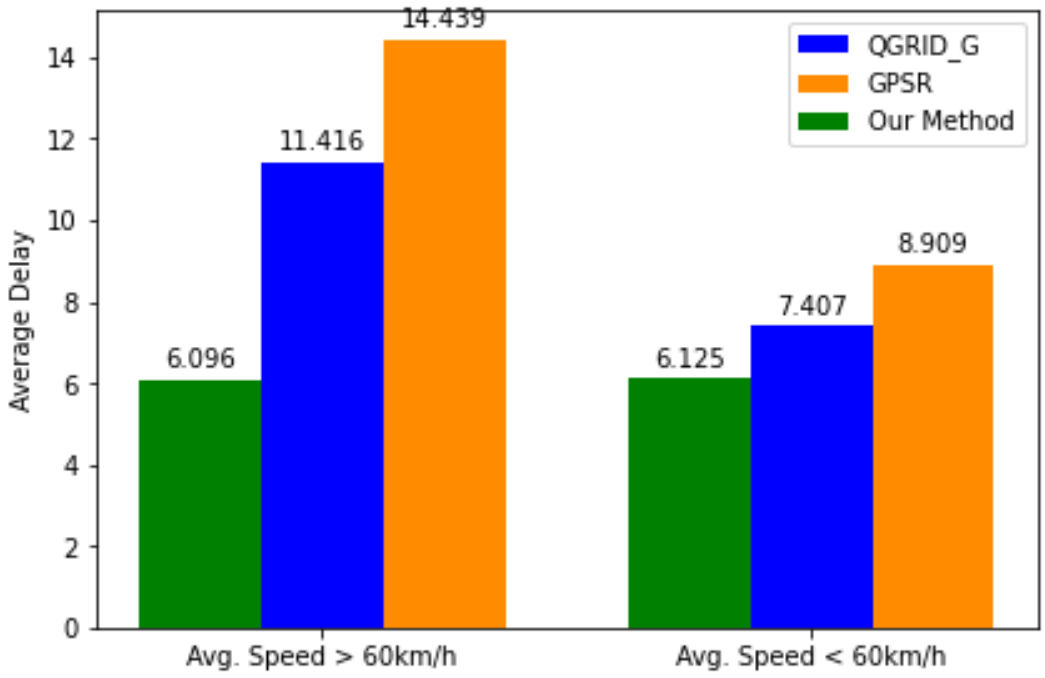


Figure 5.16: Delay average for different average velocity scenarios

Scenario 1: Highly dynamic system(Average speed in the city higher than 65 km/h): In a highly dynamic system, chance of losing packets because of consistent changes in network's topology rises. A routing protocol should be capable of handling the diverse alternations in environmental conditions including average velocity in the city.

Traditional methods does not have any specific consideration for handling these kinds of changes so as it is depicted in Fig.5.17 GPSR's delivery ratio has been drastically fallen down. QGrid_G's delivery ratio is also adversely affected by increasing the dynamicity in the network despite it is benefited from Q-learning. Reinforcement agent in QGrid_G is not applying any feedback from traffic parameters in its reward function and it seems their consideration of traffic flow in the coefficient of Q-value updater is not strong enough to deal

Methods	Scenario	P-value	Result
QGrid_G,GPSR	S > 60Km/h	**0.0004**	QGrid_G has significantly lower delay
ReLeVR,GPSR	S > 60Km/h	**0.0002**	ReLeVR has significantly lower delay
ReLeVR,QGrid_G	S > 60Km/h	**0.0020**	ReLeVR has significantly lower delay
QGrid_G,GPSR	S < 60Km/h	**0.294**	QGrid_G does not have significantly lower delay
ReLeVR,GPSR	S < 60Km/h	**0.0019**	ReLeVR has significantly lower delay
ReLeVR,QGrid_G	S < 60Km/h	**0.0099**	ReLeVR has significantly lower delay

Table 5.3: Statistical analysis of affect of average speed on delay, In which S represents average speed of vehicles in the city

with the changes in network.

On the contrary, our reinforcement agent has been capable of learning trends of traffic flow and establish the path based on it. Consequently, making the network more dynamic does not have a significant influence neither on the delivery ratio nor delay.

Fig.5.18 illustrates the delay in this scenario; The general shape of the chart are similar to other scenarios' delay, and the same as other scenarios ReLeVR beats both of other methods and delivers more messages with a lower average delay. Also another notable observation is that ReLeVR deliver most of the messages with a delay between 5 to 10 time-steps; Whereas the other method deliver major parts of messages with almost no delays but in general they have a higher delay average in comparison with ReLeVR. This trends stem from the fact that these method performance decline with increasing the distance between source and

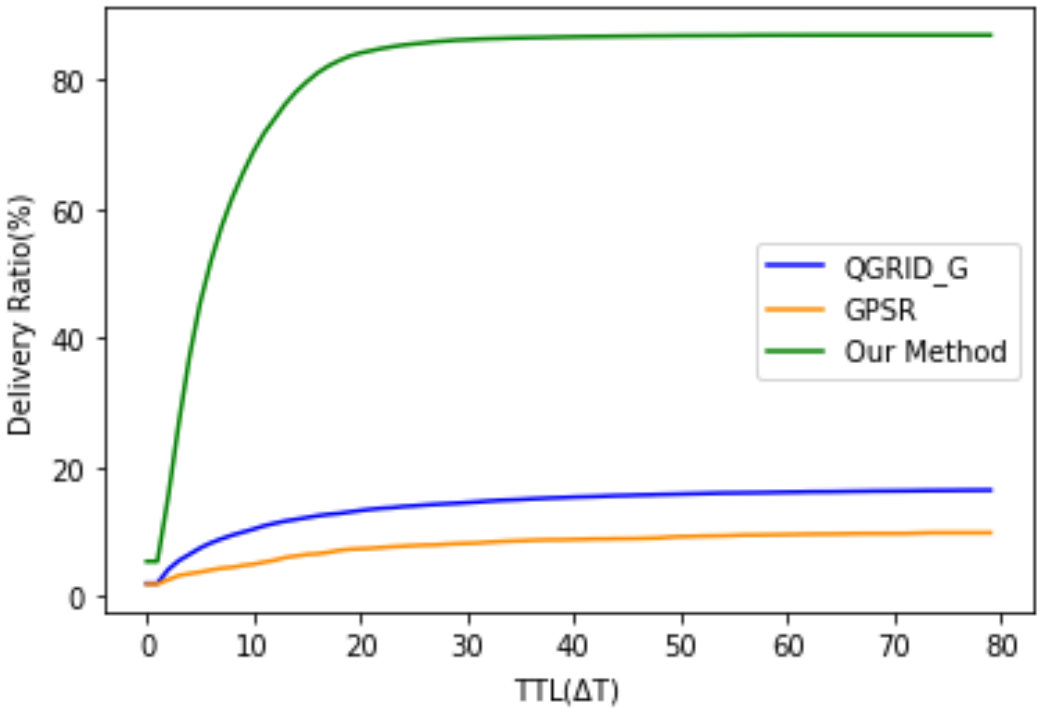


Figure 5.17: Percentage of messages that has been delivered with various delays in highly dynamic situations

destination.

Scenario 2: System with a regular dynamicity (Average speed in the city lower than 65 km/h): In reality this scenario is the regular level of dynamicity in a typical city, In which average speed of vehicles is under 65 km/h. In this scenario, all of three method perform better than highly dynamic scenario and even the difference

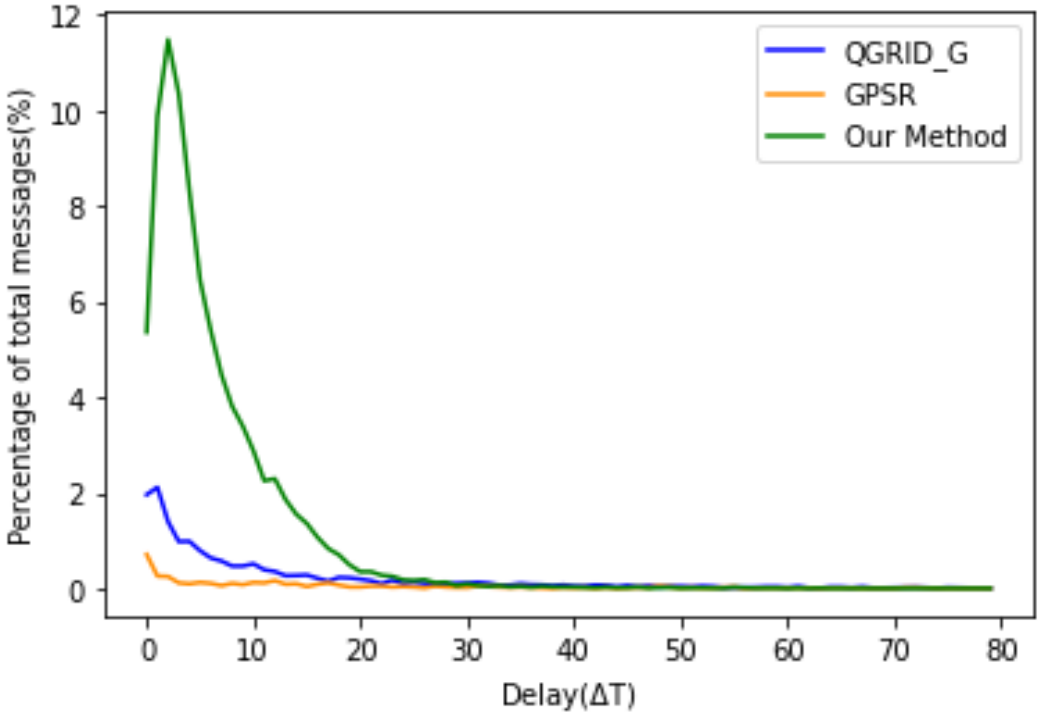


Figure 5.18: Percentage of messages that has been delivered with various delays in highly dynamic situations

between delivery ratio of our proposed method and the other method became smaller but still our method outperforms the others.

Fig.5.19 and Fig.5.20 elaborate more on the delay and delivery ratio that each of these three methods yield in a regular dynamicity situation in the city of our case study. You can see the significant improvement in the results of GPSR and QGrid_G in comparison with highly dynamic scenario that we went through it in the previous section.

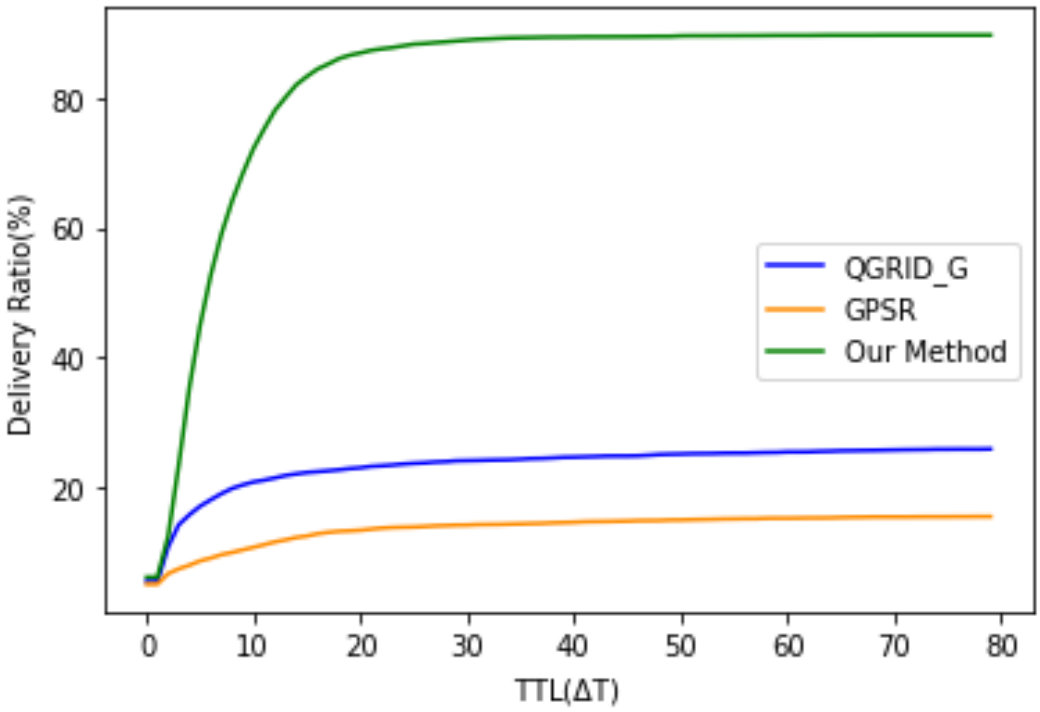


Figure 5.19: Delivery ratio by TTL in more statistic situations

Performance evaluation based on distance between source and destination

One of the most significant withdraws of the traditional methods in routing problem in VANETs is that their not scalable. Most of the machine learning-based methods also has been designed to address the other problems in VANET routing problem such as enhancing the delivery ratio and throughput or decreasing the delay. As a result, they inherited the shortage of scalability. On the contrary, one of key goals of our hybrid system is to address this program by Benefiting from RSUs while keeping the use of backbone infrastructure as less as possible.

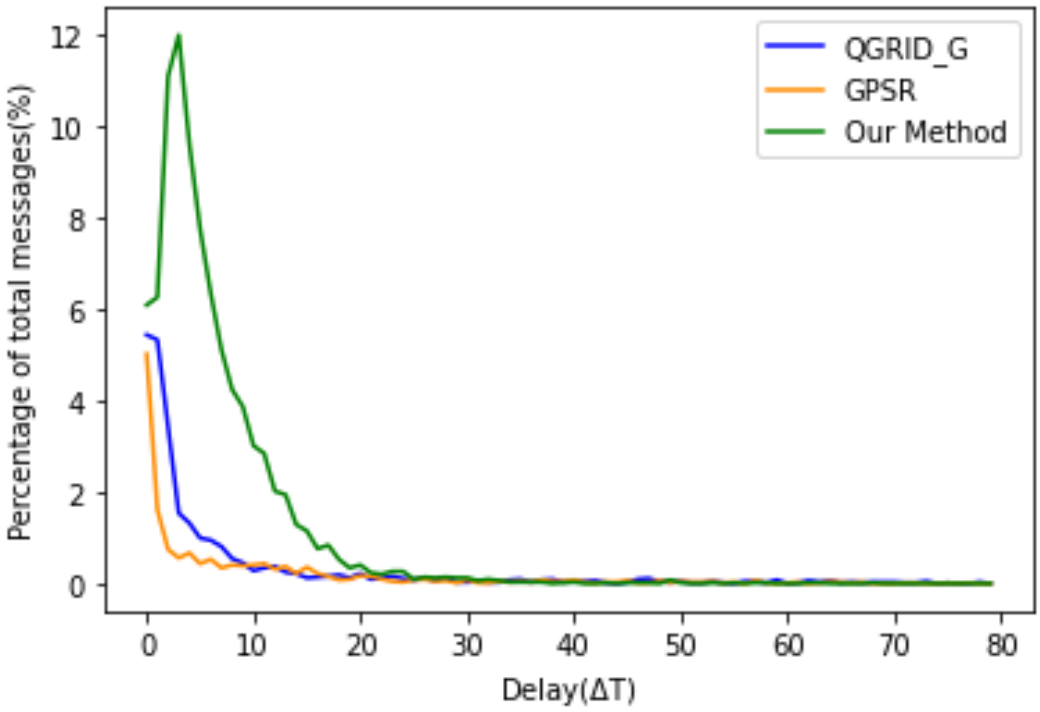


Figure 5.20: Percentage of messages that has been delivered with various delays in more statistic situations

Fig.5.21 confirms the scalability problem in both GPSR and QGrid_G. The delivery ratio for GPSR and QGrid_G has fallen down sharply right after the distance passes two grids which is a small distance in the scale of a city. On the other hand, our proposed method has managed to keep a almost same performance independent of the distance between the source and destination.

The Fig.5.22 demonstrates that our proposed method keep the delay in a small and relatively constant amount while QGrid_G shows a even higher delay in comparison with conventional methods in long distances.

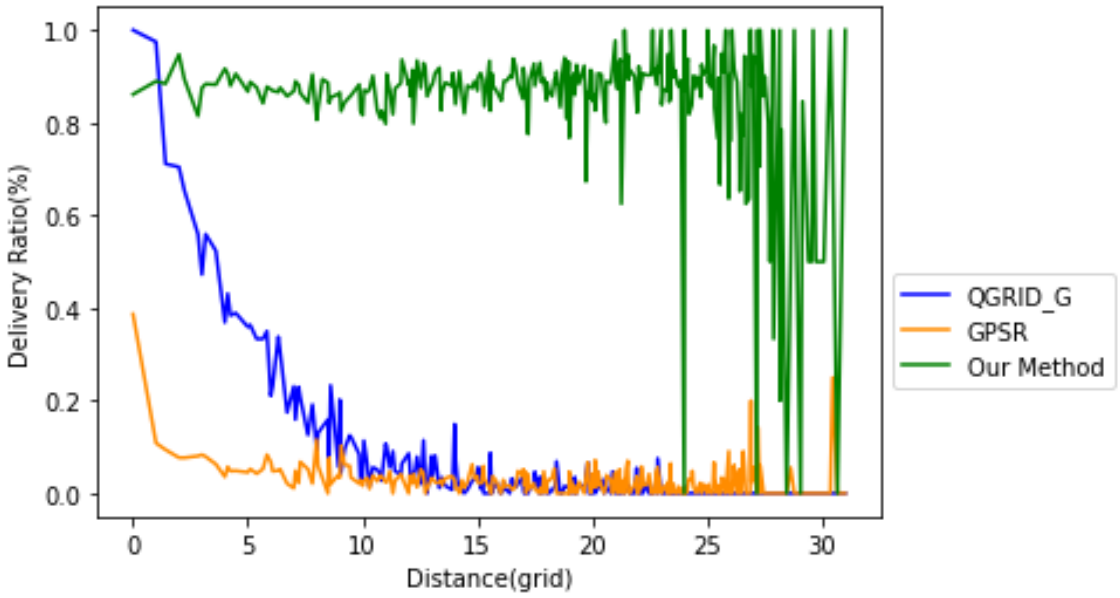


Figure 5.21: Delivery ratio by distance between source and destination

Performance evaluation without using RSUs

Finally, to make the comparison fair and prove that our designed reward function is routing the packets better even without the help of the backbone network in our hybrid system. We selected a multiple area all around the city and tried forwarding the packets inside neighbourhoods consisted of 6*6 grids without using the any RSUs just by the vehicles based on the policy that our reinforcement learning agent has learned from the traffic data. 6*6 grids that

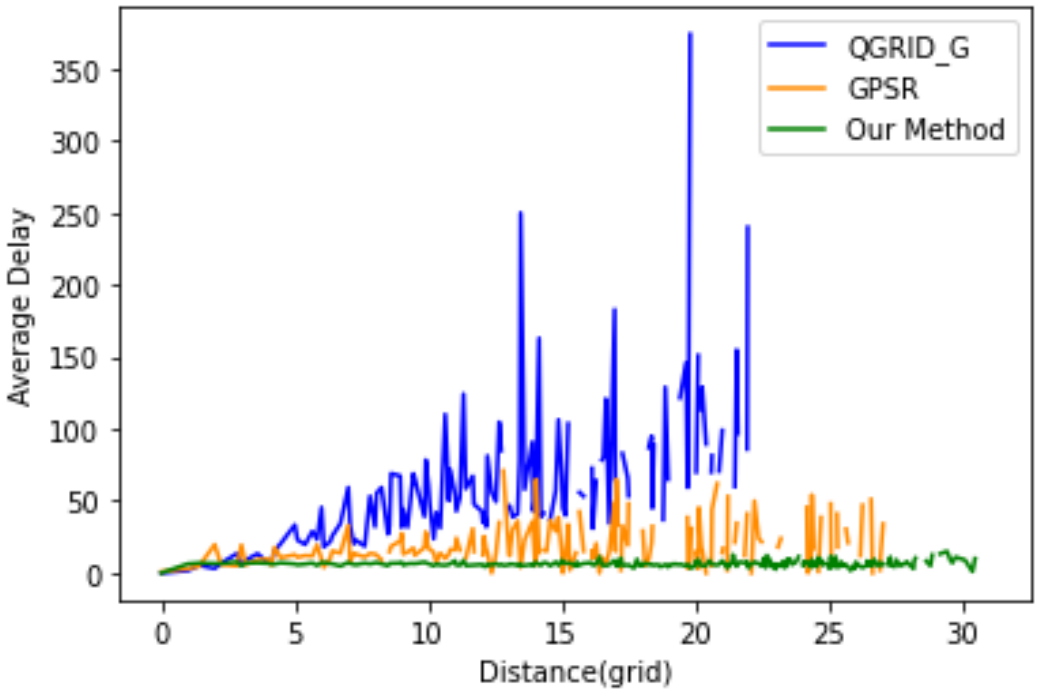


Figure 5.22: Delay by distance between source and destination

each grid area is equal to 500m*500m is the setting that authors of QGrid_G have used to evaluate their method.

Fig.5.23 shows that even in the absence of RSUs our designed reward function will lead the agent to establish more reliable and faster path between the source and destination which results in higher delivery ratio. In the [11], they published an almost 25% delivery ratio for this simulation area at this size at the TTL of 50 time-steps.

Fig.5.24 depicts the delay for all the method and as you can see ReLeVR has delivered much more messages with a lower average of delay at the end.

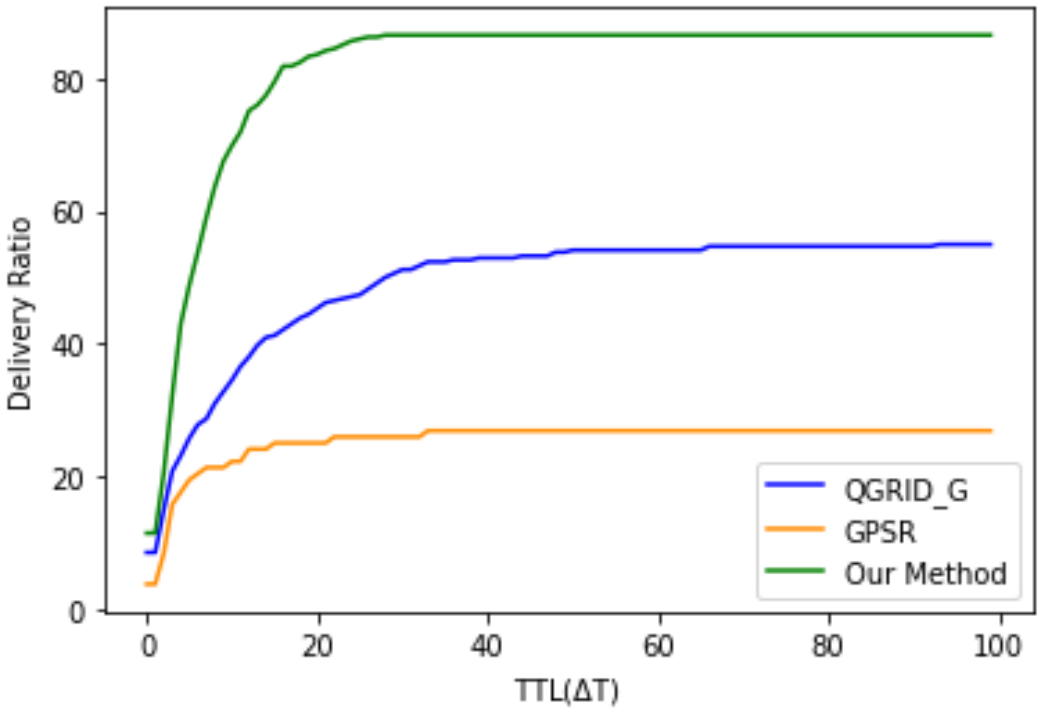


Figure 5.23: Delivery ratio by TTL without using RSUs

Simulation limitations

However, we tried to make the simulation near to real-life, like any simulation we have avoided some details to decrease the complexity of implementation. In this work, we have ignored the impact of obstacles in the environment and assumed that any message can be successfully sent and received as far as the source and destination are in each other radio range in that time step.

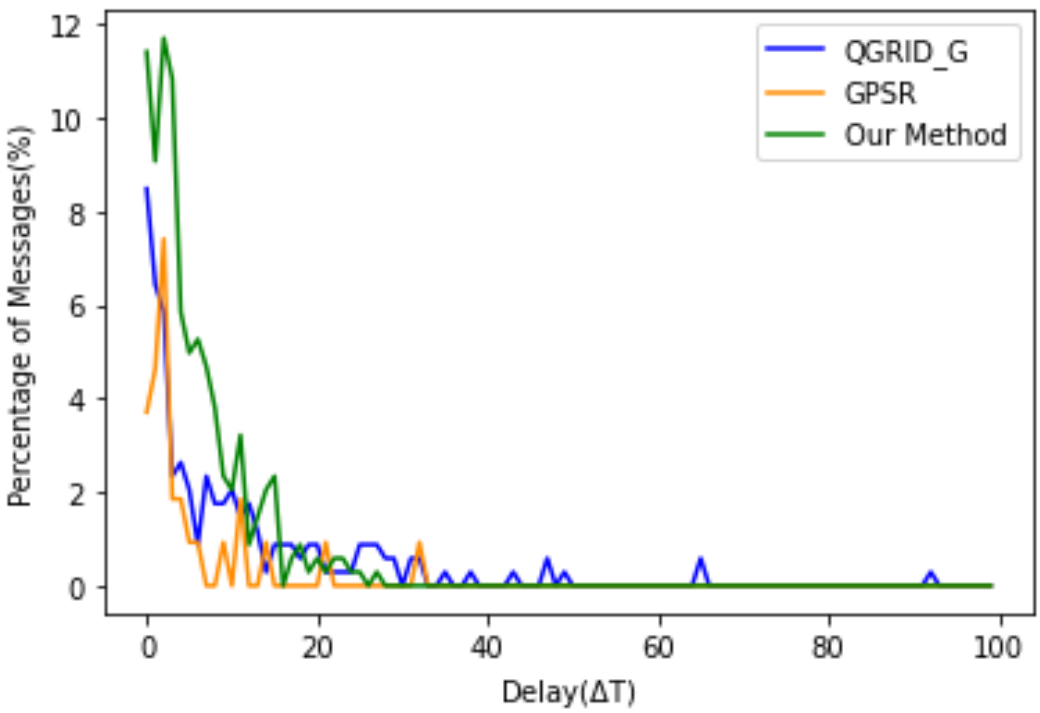


Figure 5.24: Percentage of messages that has been delivered with various delays without using RSUs

Chapter 6

Conclusion

In this work, we briefly reviewed the fundamental concepts of Vehicular Ad-hoc Networks (VANETs). Besides the increasing demand for multimedia contents in almost every network, there are some specific needs for vehicular networks such as emergency signals or traffic information; considering the scale of the potential network and the number of users, it is evident that putting all of the load on cellular networks would not be the best approach. Even if cellular networks are able to handle the communication load, it would be extremely expensive for most users since cellular communications are often costly. By allocating a portion of these communication loads to the vehicles on the roads, VANETs can be the answer to these demands while keeping the cost at reasonable rates.

In this work, we first divided the map of Antwerp city into several grids. Using the grid-map, we then proposed a traffic-aware RSU allocation method with the objective of minimizing the number of RSUs in the system and thus the realization of such a network.

Working with actual traffic data in a city as vast as Antwerp, we observe that one reason behind the packet loss in the network is the considerable distance between the source and the destination. In order to overcome this problem, we approached a hybrid routing scheme that allocates the communication load to both vehicles and the backbone network.

There are usually some patterns in the traffic flow of every city. This attribute makes machine learning algorithms to be a potentially promising tool to work with. We have chosen Q-learning, a reinforcement learning algorithm, to extract these patterns and provide a routing policy for the vehicles. This policy determines which grid the message should be sent to in the next step along the way for each location in the city. After the Q-learning algorithm learns the policy, it is broadcast to all of the city's vehicles. In the next step, we choose the vehicle which is supposed to receive the packet, and we do that by greedily choosing the one closest to the destination or an RSU.

The result of our extensive experiments shows that our proposed method improves the message delivery rate and decreases the delay in comparison to a well-known conventional method, GPSR, and a Q-Learning-based method, QGRID_G. Moreover, we have minimized the performance's sensitivity to VANETs' attributes such as velocity or vehicle density on the roads. Furthermore, compared to one of the latest related work, which also considers hybrid communication, we managed to deploy a far fewer number of roadside units to achieve the same performance.

For future works, one can think of deploying Deep Q-Learning to include more features in the state space. E.g. the vehicles' moving direction can also be regarded as a feature

along with other time-based data such as the week of the year or the month of the year. It is also possible to include a city's public transformation system as a part of the VANET and benefit from its extensive, organized infrastructure. Having a reliable periodic pattern of public transportation can significantly mitigate the difficulties of addressing the routing problem; however, it is not easy to access such data.

Bibliography

[1] Altayeb, M., and Mahgoub, I. A survey of vehicular ad hoc networks routing protocols. *International Journal of Innovation and Applied Studies 3*, 3 (2013), 829–846.

[2] Ayaida, M., Barhoumi, M., Fouchal, H., Ghamri-Doudane, Y., and Afilal, L. HHLS: A hybrid routing technique for VANETs. In *2012 IEEE Global Communications Conference (GLOBECOM)* (2012), IEEE, pp. 44–48.

[3] Chen, K.-H., Dow, C.-R., Chen, S.-C., Lee, Y.-S., and Hwang, S.-F. Harpiagrid: A geography-aware grid-based routing protocol for vehicular ad hoc networks. *J. Inf. Sci. Eng. 26*, 3 (2010), 817–832.

[4] Cheng, J., Cheng, J., Zhou, M., Liu, F., Gao, S., and Liu, C. Routing in internet of vehicles: A review. *IEEE Transactions on Intelligent Transportation Systems 16*, 5 (2015), 2339–2352.

[5] De Rango, F., Veltri, F., Fazio, P., and Marano, S. Two-level trajectory-based routing protocol for vehicular ad hoc networks in freeway and manhattan environments. *Journal of Networks 4*, 9 (2009), 866–880.

[6] Fonseca, A., and Vazão, T. Applicability of position-based routing for VANET in highways and urban environment. *Journal of Network and Computer Applications 36*, 3 (2013), 961–973.

[7] Goel, N., Sharma, G., and Dhyani, I. A study of position based VANET routing protocols. In *2016 international conference on computing, communication and automation (ICCCA)* (2016), IEEE, pp. 655–660.

[8] Hu, T., Liwang, M., Huang, L., and Tang, Y. An enhanced GPSR routing protocol based on the buffer length of nodes for the congestion problem in VANETs. In *2015 10th International Conference on Computer Science & Education (ICCSE)* (2015), IEEE, pp. 416–419.

[9] Karp, B., and Kung, H.-T. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking* (2000), pp. 243–254.

[10] Kiess, W., Füssler, H., Widmer, J., and Mauve, M. Hierarchical location service for mobile ad-hoc networks. *ACM SIGMOBILE mobile computing and communications review 8*, 4 (2004), 47–58.

[11] Li, F., Song, X., Chen, H., Li, X., and Wang, Y. Hierarchical routing for vehicular ad hoc networks via reinforcement learning. *IEEE Transactions on Vehicular Technology 68*, 2 (2018), 1852–1865.

[12] Luo, Y., Zhang, W., and Hu, Y. A new cluster based routing protocol for VANET. In *2010 second international conference on networks security, wireless communications and trusted computing* (2010), vol. 1, IEEE, pp. 176–180.

[13] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., and Kim, D. I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials 21*, 4 (2019), 3133–3174.

[14] Makridis Michail, Mattas Konstantinos, C. B. Traffic simulation data for antwerp, 2017.

[15] Seet, B.-C., Liu, G., Lee, B.-S., Foh, C.-H., Wong, K.-J., and Lee, K.-K. A-STAR: A mobile ad hoc routing strategy for metropolis vehicular communications. In *International conference on research in networking* (2004), Springer, pp. 989–999.

[16] Shah, S. S., Malik, A. W., Rahman, A. U., Iqbal, S., and Khan, S. U. Time barrier-based emergency message dissemination in vehicular ad-hoc networks. *IEEE Access 7* (2019), 16494–16503.

[17] Sun, W., Yamaguchi, H., Yukimasa, K., and Kusumoto, S. Gvgrid: A QoS routing protocol for vehicular ad hoc networks. In *2004 14th IEEE International Workshop on Quality of Service* (2006), IEEE, pp. 130–139.

[18] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction.* MIT press, 2018.

[19] Ullah, A., Yao, X., Shaheen, S., and Ning, H. Advances in position based routing towards ITS enabled FoG-oriented VANET-a survey. *IEEE Transactions on Intelligent Transportation Systems* (2019).

[20] Ullah, A., Yao, X., Shaheen, S., and Ning, H. Advances in position based routing towards its enabled fog-oriented vanet-a survey. *IEEE Transactions on Intelligent Transportation Systems* (2019).

[21] Varshney, P. K., Agrawal, G., and Sharma, S. K. Relative performance analysis of proactive routing protocols in wireless ad hoc networks using varying node density. *Invertis Journal of Science & Technology 9*, 3 (2016), 161–169.

[22] Watkins, C. J., and Dayan, P. Q-learning. *Machine learning 8*, 3-4 (1992), 279–292.

[23] Wikipedia contributors. Markov decision process — Wikipedia, the free encyclopedia, 2021. [Online; accessed 12-April-2021].

[24] Wu, J., Fang, M., Li, H., and Li, X. RSU-assisted traffic-aware routing based on reinforcement learning for urban vanets. *IEEE Access 8* (2020), 5733–5748.

[25] Yang, X., Li, M., Qian, Z., and Di, T. Improvement of GPSR protocol in vehicular ad hoc network. *IEEE Access 6* (2018), 39515–39524.

[26] Zhang, F., Liu, H., Leung, Y.-W., Chu, X., and Jin, B. Community-based bus system as routing backbone for vehicular ad hoc networks. In *2015 IEEE 35th International Conference on Distributed Computing Systems* (2015), IEEE, pp. 73–82.

[27] Zhang, F., Liu, H., Leung, Y.-W., Chu, X., and Jin, B. CBS: Community-based bus system as routing backbone for vehicular ad hoc networks. *IEEE Transactions on Mobile Computing 16*, 8 (2016), 2132–2146.