

# Reinforcement Learning-based Dynamic Service Placement in Vehicular Networks

Anum Talpur and Mohan Gurusamy

Department of Electrical & Computer Engineering

National University of Singapore, Singapore

Email: anum.talpur@u.nus.edu, gmohan@nus.edu.sg

**Abstract**—The emergence of technologies such as 5G and mobile edge computing has enabled provisioning of different types of services with different resource and service requirements to the vehicles in a vehicular network. The growing complexity of traffic mobility patterns and dynamics in the requests for different types of services has made service placement a challenging task. A typical static placement solution is not effective as it does not consider the traffic mobility and service dynamics. In this paper, we propose a reinforcement learning-based dynamic (RL-Dynamic) service placement framework to find the optimal placement of services at the edge servers while considering the vehicle's mobility and dynamics in the requests for different types of services. We use SUMO and MATLAB to carry out simulation experiments. In our learning framework, for the decision module, we consider two alternative objective functions - minimizing delay and minimizing edge server utilization. We developed an ILP based problem formulation for the two objective functions. The experimental results show that 1) compared to static service placement, RL-based dynamic service placement achieves fair utilization of edge server resources and low service delay; and 2) compared to delay-optimized placement, server utilization optimized placement utilizes resources more effectively, achieving higher fairness with lower edge-server utilization.

**Index Terms**—Service placement, reinforcement learning, vehicular networks.

## I. INTRODUCTION

THE platform of fifth-generation network (5G) brings tremendous benefits in vehicular communications, including transportation efficiency, improved safety, high reliability, low latency, and large communication coverage. 5G networks are highly-flexible and programmable end-to-end networks that provide enhanced performance while meeting various requirements from multiple services. The International Telecommunications Union (ITU) has categorized the diverse 5G services into three major use-cases, namely, (i) enhanced Mobile Broadband (eMBB) supporting very high data rate of  $\approx 10$  Gbps, (ii) ultra-Reliable and Low Latency Communications (URLLC) with high reliability and very low end-to-end latency of about 1 ms, and, (iii) massive Machine Type Communications (mMTC) supporting a density of  $\approx 106$  devices/km<sup>2</sup> [1]. To support vertical applications of different performance requirements, the next-generation mobile network (NGMN) alliance has introduced the concept of network slicing [2]. Network slices are virtual entities (or virtual network functions) that are deployed on a common-physical infrastructure to satisfy the diverse requirements of the use-cases in terms of functionalities and performance. A

mobile operator is allowed deploy different slices in parallel while guaranteeing isolation so that services of one slice do not affect services in another slice.

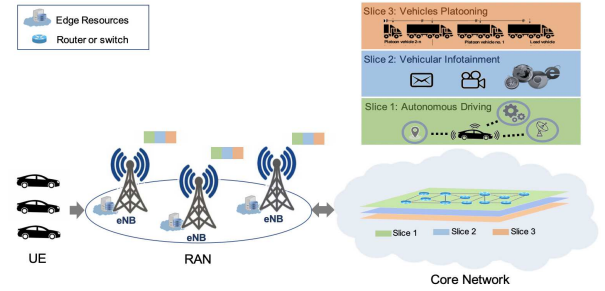


Fig. 1: An example 5G slicing system for vehicular applications

In a 5G-based vehicular network, a slice could be instantiated for a specific application, for which each slice is capable of providing multiple services. Vehicles communicate with each other or surrounding infrastructure for availing coordination in driving, convenience, road safety, and many other applications. Fig. 1 shows an example model of a 5G slicing system for three different vehicular applications including, autonomous driving, vehicular infotainment, and vehicle platooning. To support these applications, several computational operations need to be performed within a network. The European Telecommunications Standards Institute (ETSI) outlines the use of mobile edge computing (MEC) along with vehicular networks to satisfy various vehicle-to-everything (V2X) service requirements including, low delay, low computational cost, security, reliability, and so on [3]. The closer the application server is to a vehicle, the faster the communication and better the coverage is for the vehicle [4].

As shown in Fig. 1, the 5G slices support three kinds of applications in a vehicular network. The slices share the resources of radio access network (RAN), edge and core networks. Services can be deployed at the servers making use of compute, network and storage resources. Service placement is the problem of mapping services to the edge servers in vehicular networks to satisfy the requirements for the requested services while utilizing the resources efficiently. From the perspective of vehicles, the delay perceived by a vehicle is an important metric which should be minimum. From the service

provider perspective, it is important to minimize the server resource utilization while keeping the resource utilization across the servers as balanced as possible. This will enable servers to scale up the resources to handle the events of congestion, failures and changing service demands.

The growing complexity of traffic patterns and dynamics in the requests for different types of services has made service placement more challenging. It is necessary to adopt continual learning of the environment for providing better services. Therefore, embedding intelligence using machine learning (ML) has drawn research interests recently. Different ML techniques have received significant attention and reinforcement learning (RL) is an attractive approach for various problems in the area of vehicular communications [5].

In [6], the authors address the problem of V2X service placement using delay as the objective function. It performs fixed placement based on the optimization formulation considering a static service placement problem that does not take into account the dynamicity of service requests. A static solution in [6] that fixes servers for hosting services is not effective for a mobile and dynamic scenario of a vehicular network. It is therefore imperative that the real-time environment be taken into consideration while mapping service to an edge server. While our work considers a delay-based optimization similar to [6], we use it as one of the two possible objective functions to be used by the decision module in our learning framework. Different from [6], we consider the dynamic service placement problem with the proposal of a new solution approach based on reinforcement learning to continuously learn and adapt to the dynamics of the system.

In this paper, we propose a reinforcement learning-based dynamic (RL-Dynamic) service placement framework to find the optimal placement of services at the edge servers while considering the vehicle's mobility and dynamics in the requests for different types of services. We use SUMO and MATLAB to carry out simulation experiments. In our learning framework, for the decision module, we consider two alternative objective functions - minimizing delay and minimizing edge server utilization. We developed an ILP based problem formulation for the two objective functions. The experimental results show that 1) compared to static service placement, RL-based dynamic service placement achieves fair utilization of edge server resources and low service delay; and 2) compared to delay-optimized placement, server utilization-optimized placement utilizes resources more effectively, achieving higher fairness with lower average edge-server utilization.

The remaining sections are organized as follows. Section II provides an overview of the related work in the literature. Section III describes the network model, request model and problem description. Section IV presents the proposed method. Section V discusses the experimental setup and results, and section VI concludes the paper.

## II. RELATED WORK

In the literature, there are few recent works focusing on the problem of optimal service placement for vehicular networks.

In [6], the authors consider the problem of V2X service placement. They propose an ILP model for minimizing the average service delay where the scope of the environment is limited to the highway with constant speed, fixed distance between vehicles, and movement of vehicles in one direction. The delay experienced by the vehicles for V2X communication is also based on randomly assigned values from a given range. In [7], the authors consider a more realistic scenario of the highway environment for V2X service placement. They consider 5 different V2X applications for minimization of communication and download link delay using binary ILP model. Some work on V2X applications are carried out in the context of cloud computing and fog-computing [8]–[11]. One common aspect in most of the previous works is the consideration of latency or delay as the objective. Some works consider priority [11] and cost [9], [10] as additional factors for service migration. The above works basically consider a static service placement problem which do not consider the dynamicity of service requests. Different from the existing works, our work makes contributions in consideration of the dynamic service placement problem and development of a reinforcement learning-based dynamic (RL-Dynamic) solution to provide services with low delay while keeping the server resource utilization low. In our learning framework, for the decision module, we consider two alternative objective functions - minimizing delay and minimizing edge server utilization.

## III. PROBLEM DESCRIPTION

In this section, we describe the problem and system model of the proposed approach.

### A. Assumptions

In this paper, we assume a city road environment with multiple lanes in different directions. The vehicles are moving along the road by randomly choosing a source, destination, and speed to start and end their journey at different times. The speed limit regulations specified in the SUMO simulator, for the type of vehicle and environment are followed. We assume the city environment is under 5G coverage using evolved NodeB (eNB) stations. 5G is a perfect enabler for V2X applications [12]. The deployment of eNB follows urban-macro 5G regulations for which inter-site distance (ISD) is 500m [13]. It is also assumed that eNBs are equipped with MEC hosts to form the network edge with limited capacity servers. Additionally, the network edge connects the core cloud data center with large capacity servers via a backbone network. Further, we assume adequate links between different nodes and servers to enable the communication.

### B. Network Model

We use  $E$  to denote a set of edge servers with  $i \in E$  as an edge node. For each edge node  $i$ , the residual computing resources (available resources) is denoted by  $C_i$ . Let  $V$  and  $S$  denote a set of vehicles (UEs) and service types (services), respectively. A vehicle  $v \in V$  requires a service  $s \in S$  which is to be hosted at an edge node. The amount of resources

consumed by deploying service  $s$  at edge node  $E$  is denoted by  $R_s$ , and the delay/latency requirement for each service  $s$  is denoted as  $D_s$ . The notations are summarized in Table. I.

TABLE I: Summary of Notations

Notation	Description
$E$	Set of edge servers
$S$	Set of services
$V$	Set of vehicles
$R_s$	Resources consumed by service $s$
$C_i$	Available resources at edge node $i$
$x_i^s$	Assignment of service $s$ at the edge node $i$
$\phi_i$	Server utilization of edge node $i$
$\beta$	Balancing factor
$d_{i,v}^s$	The time delay experienced by vehicle $v$ when services $s$ is deployed at node $i$
$D_s$	Delay threshold or maximum allowed delay for service $s$
$U_s$	Service demand (number of UEs requesting service $s$ )
$N_i$	Number of UEs edge node can handle

### C. Service Request Model

A service request is specified as a 4-tuple  $(v, loc, t, s)$ , wherein  $v$ ,  $loc$ ,  $t$ , and  $s$  represent the vehicle ID, vehicle location, time of request and type of service, respectively. We assume each entity (i.e. vehicle/UE) is equipped with a clock and GPS, which enables it to specify time  $t$  and location  $loc$  in its service request message. Associated with each service  $s$ , there are delay and resource requirements. In response to the request, the location of the edge server where the requested service is deployed, will be returned.

### D. Problem and Proposed Approach

We consider a service placement problem in vehicular networks with edge servers having limited resources. Given a set of services belonging to different network slices with their resource and delay requirements, the problem is to find the optimal placement of services at the edge servers while considering the vehicle's mobility and dynamics in the requests for different types of services. The number of vehicles requesting service  $s$  and their distance from different edge servers is very dynamic. A static solution which fixes servers for hosting services is not effective for a mobile and dynamic scenario of a vehicular network. It is therefore imperative that the real-time environment be taken into consideration while mapping a service to an edge server. With this goal, we propose a RL-based approach and develop an RL-Dynamic service placement algorithm which facilitates remapping of services to edge servers in accordance with the changing vehicular environment. Our solution framework uses a classic model-free Q-learning algorithm which finds an optimal set of actions  $a$  which optimize a certain objective such as minimizing resource utilization or minimizing the delay while satisfying the service requirements.

## IV. PROPOSED RL-BASED DYNAMIC (RL-DYNAMIC) SERVICE PLACEMENT FRAMEWORK

In this section, we present the proposed RL-Dynamic service placement framework, for the problem described above.

We first summarize the state space, reward function, the action space and state transition policy used in our RL framework.

- **State Space  $\omega_t(s)$ :** The state space set describes the network environment, formed from the request messages  $([v_1, loc_1, t, s], [v_2, loc_2, t, s], \dots, [v_n, loc_n, t, s])$  for service  $s$  at time  $t$ , where  $n$  is number of vehicles requesting for service  $s$ .
- **Reward Function  $r(\omega_t(s), a_t)$ :** The reward is measured from the average delay feedback from vehicles in accessing their service from the associated edge server.
- **Action Space and State Transition Policy  $a_t$ :** The action space describes the action taken by the decision module for placement of service  $s$  on the edge node  $i$ . On the other hand, the state transition policy defines the condition, upon which re-optimization of the decision matrix for the new service placement will take place. It aims to maximize the q-value for the action space.

Fig. 2 depicts the framework of the RL-Dynamic service placement algorithm which consists of three modules (decision module, learning module, and data repository). The *decision module* has direct interaction with the UEs. The request for service  $s$  by UE  $v$  is specified in the form of a 4-tuple  $(v, loc, t, s)$ , as discussed in Section III. In return, considering the demand for service  $s$  at time  $t$  and location  $loc$  of vehicles (UEs) requesting for service  $s$ , the decision module selects the servers for the services to place based on the Q-table maintained in the data repository by the learning module. The *learning module* optimizes the objective function subject to different constraints. We use two alternative objective functions in the framework in our study. First, we consider delay/latency as the objective function for providing a RL-Dynamic solution. Second, we consider edge-server utilization as the objective function.

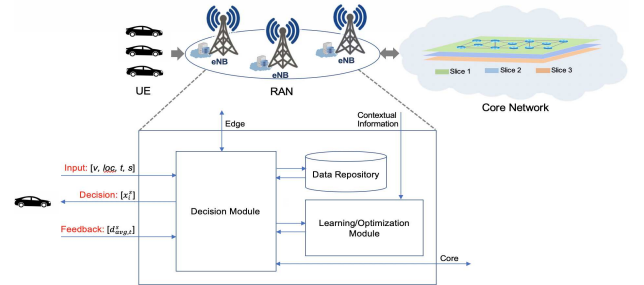


Fig. 2: The proposed RL-based dynamic (RL-Dynamic) service placement framework

**Delay:** In this work, the delay problem is formulated as, Minimize

$$\sum_{s \in S_u} \sum_{i \in E} \left( \frac{1}{|V|} \sum_{v \in V} d_{i,v}^s \right) x_i^s \quad (1)$$

Subject to:

$$\sum_{i \in E} x_i^s = 1; \forall s \in S \quad (2)$$

$$\sum_{s \in S_u} x_i^s \leq 1; \forall i \in E \quad (3)$$

$$\sum_{s \in S_u} x_i^s \left( \frac{1}{|V|} \sum_{v \in V} d_{i,v}^s \right) \leq D_s; \forall i \in E \quad (4)$$

$$\sum_{i \in E} x_i^s R_s \leq C_i; \forall s \in S \quad (5)$$

$$\sum_{i \in E} x_i^s U_s \leq N_i; \forall s \in S \quad (6)$$

$$x_i^s \in \{0, 1\}; \forall s \in S, \forall i \in E \quad (7)$$

$x_i^s$  is a binary variable used to indicate the placement of service  $s$  at node  $i$ . If edge node  $i$  deploys service  $s$ ,  $x_i^s$  is 1. Otherwise, it is 0. Constraint (2) ensures that each service should be deployed onto exactly one edge server. Constraint (3) guarantees each edge server node hosts a different and unique service. This condition ensures no repetition of the same service at different edge servers. Constraint (4) ensures that the average delay experienced by vehicles requesting service  $s$  should be less than the service's maximum delay threshold. Constraint (5) ensures that the available resources at the edge node is not exhausted while deploying service  $s$ . Constraint (6) guarantees the edge node has the capacity to handle the number of UEs requesting service  $s$ . Finally, condition (7) defines the decision variable  $x_i^s$  as a binary integer decision variable.

*Server-Utilization:* In this case, we consider minimizing the total server resource utilization. The objective function is formulated as,

Minimize

$$\sum_{s \in S} \sum_{i \in E} \varphi_i (U_s + \beta) x_i^s \quad (8)$$

The rationale for minimizing the server utilization is to decrease the possibility of congestion so that a server has enough room for resource scale-up. Our proposed objective function aims to minimize the total edge server utilization<sup>1</sup> weighted by service demand<sup>2</sup>. In our objective function, we use a small offset value  $\beta$  (i.e. balancing factor) to account for a situation of zero requests for any service as it would otherwise result in multiplication by zero in the objective function. The difference between the two ILP formulations is in the objective function. On the other hand, the constraints are the same.

The decision matrix calculated by the learning module is tabulated for a quick reference by the decision module. It is stored in the *data repository* in the form of a Q-table. A high Q-value means a high-quality decision. The decision module will look-up the Q-table to forward the new decision to a vehicle (UE), and updates the Q-value and reward based on the feedback for the previous decision. In our model, the iterative maintenance of the lookup table (i.e. Q-table) uses the Bellman

equation to observe a given state of the environment. The Q-value of a state at a given time is calculated as,

$$Q^{new}(\omega_t(s), a_t) = \alpha(r(\omega_t(s), a_t)) + \gamma(\max Q(\omega_{t+1}(s), a)) + (1 - \alpha)Q(\omega_t(s), a_t) \quad (9)$$

where,  $(\omega_t(s), a_t)$  is a state-action pair at time  $t$  and  $r(\omega_t(s), a_t)$  is the reward of applying action  $a_t$  at state  $\omega_t(s)$ . The value  $\alpha$  and  $\gamma$  is the learning rate and discount factor, respectively. The greater the discount factor  $\gamma$  is, the more the effect of future reward is applied to the new Q-value. In our case,  $\omega_{t+1}(s)$  is not deterministic and we only consider the historic performance and current feedback/reward to update Q-table, therefore we set  $\gamma = 0$ . The larger value of  $\alpha$  implies more significance of the current feedback reward. The decision module continuously receives feedback from vehicles about average delay  $d_{avg,t}^s$  experienced by vehicles requesting for service  $s$  at time  $t$ . In this work, we assume the reward is either 1, 0.5 or  $-\infty$ . The reward represents the average delay is either increased (0.5), decreased (1), or violating delay threshold ( $-\infty$ ), as shown in Equation 10. Here,  $\infty$  means a very large value which converts q-value into a negative value to trigger the violation of constraints. This may vary with the size and scope of a network. In our model, we use the value -10.

$$r(\omega_t(s), a_t) = \begin{cases} 0.5 & d_{avg,t-1}^s \leq d_{avg,t}^s < D_s \\ +1 & d_{avg,t-1}^s \geq d_{avg,t}^s < D_s \\ -\infty & \text{else} \end{cases} \quad (10)$$

We present the proposed RL-Dynamic service placement technique in Algorithm 1. In line 5-10, the learning module solves ILP to calculate optimal decision matrix  $x_i^s$  at time  $t = 1$  for the given set of requests. The decision module performs action  $a_t$  (i.e. forwarding information of corresponding service and its deployment location to the vehicles) for all services. It also stores the decision matrix in the form of a lookup table at the data repository. The iterative re-optimization of the decision matrix will take place after every fixed period to capture the dynamicity of vehicular networks (Line 11-18). The decision module monitors the state of the Q-value for the stored action set. According to the state transition policy used in this work, if the Q-value is decreased twice consequently for any service  $s$  or the constraint of delay is violated then the learning module performs re-optimization of  $x_i^s$  over the newly-collected set of data at the given time. This is updated in the Q-table as well. The state transition policy may vary depending on the size of a network and the periodicity of monitoring q-values.

## V. EXPERIMENTAL SETUP AND RESULTS DISCUSSION

To evaluate the performance of the proposed RL-Dynamic service placement mechanism, we use SUMO and MATLAB to set up the simulation environment. SUMO is an open-source simulator, used to simulate a virtual traffic scenario of a realistic vehicular network. In this work, we extract the area of  $3km^2$  around the National University of Singapore using Openstreetmaps. The choice of the area is significant

<sup>1</sup> Server utilization is defined as the ratio between the resources that any service type  $s$  will consume and the available resources at the edge node

<sup>2</sup> Service demand is defined as the total number of UEs requesting for service  $s$

**Algorithm 1: RL-Dynamic Service Placement**


---

**Input:**  $[v, loc, t, s]$   
**Output:** State-Action pair  $(\omega(s), a)$

- 1 **Initialization:**  $r(\omega_t(s), a_t) = 0$ ,  $Q(\omega_t(s), a_t) = 0$
- 2 Set  $d_{avg,t}^s = \frac{1}{|V|} \sum_{v \in V} d_{i,v}^s$
- 3 **for**  $t=1,2,3,\dots$  **do**
- 4   **for all**  $s \in S$  **do**
- 5     Collect  $\omega_t(s)$
- 6     **if**  $t=1$  **then**
- 7       calculate  $x_i^s$  using ILP (learning-module)
- 8       set  $Q(\omega_t(s), a_t) = x_i^s$
- 9       create Q-table from  $x_i^s$ ,  $Q(\omega_t(s), a_t)$ , and  $r(\omega_t(s), a_t)$
- 10      perform action  $a_t$
- 11    **else**
- 12      perform action  $a_t$  using Q-table
- 13      calculate feedback  $d_{avg,t}^s$  and store
- 14      calculate reward  $r(\omega_t(s), a_t)$  using (10)
- 15      Update  $Q(\omega_t(s), a_t)$  using (9)
- 16      **if**  $Consecutive\_Decrements(Q(\omega_t(s), a_t)) == true \parallel Constraint\_Violated == true$  **then**
- 17        Reoptimize  $x_i^s$  using ILP for the new set of data
- 18        Update Q-table

---

as it is present in the center of the city with high traffic densities (Urban environment). Furthermore, the randomTrip application of the SUMO package is used to automatically generate the trips for the vehicles with mobility over the given area of the map. We collect traces of data which helps to generate a 4-tuple message dynamically for our algorithm. The implementation of RL-based optimization is carried out using MATLAB. All experiments are evaluated on a system with Intel Corei5 2GHz and 8GB RAM. The implementation of the proposed RL-Dynamic mechanism for a given set of vehicles has a significantly low run time of  $\approx 0.01$ -0.02sec.

Table II lists the parameters used in the simulation. Different sets of values are chosen for performing multiple experiments. Small delay threshold is chosen to enforce strict delay constraints. Whereas, the selection of resource unit for  $R_s$  and  $C_i$  is random. Experiments were performed for different sets (by choosing lowest values as well as the highest values) of  $R_s$  and a similar performance trends are observed. We use average service delay, percentage of server utilization, and fairness in edge server utilization as our evaluation metrics. We evaluate our algorithm termed as RL-Dynamic for the two objective functions, delay optimization (D-Optimization) and server utilization optimization (SU-Optimization). We compare our algorithms with the static solution (termed Static) which has a fixed placement based on the optimization solution. We note that the varying distance between a vehicle and the server

hosting the service due to the mobility has an impact on the delay and service placement. We carry out experiments (trials) five times with different random seeds. We present the results for different trials. We also present the average of five trials to ensure low confidence intervals.

TABLE II: Simulation Parameters

Parameters	Value
$S$	6
$V$	100
$E$	6
$R_s(Unit)$	[60 20 60 40 50 70]
$C_i(Unit)$	[60 60 70 80 90 100]
$D_s(ms)$	[5 4 4.5 5 5 5.5]
$N_i$	100
$\alpha$	0.75
$\beta$	0.1
$\gamma$	0
$t(sec)$	1 to 500

**A. Average Service Delay**

Fig. 3 shows the average delay experienced for different services by the vehicles. It compares the average service delay of the static placement and RL-Dynamic placement scenarios (for both optimization models). Here, the static placement refers to one-time solution of an ILP formulation. It can be observed that the average delay observed by vehicles for the static service placement is higher than the RL-Dynamic service placement. This is due to the fact that the vehicular environment is not stationary. The high mobility of nodes and constantly changing topology requires continual learning of the environment (as in the case with our RL-Dynamic service placement) to provide a better average delay for each service. It can be observed that the static placement is not able to satisfy the delay requirement for some services. In contrast, the delay for RL-Dynamic placement is always well below its threshold for all trials. Moreover, with the SU-optimization model, although the delay is not the lowest when compared with the D-optimization model, it is always under the required threshold value. This shown the effectiveness of adopting a learning method by the dynamic approach.

**B. Edge Server Utilization**

In this section, we consider edge server utilization and evaluate the performance in terms of fairness and average percentage of utilization. The fairness of server utilization are a representation of fair and efficient resource consumption. We use Jain's index as a fairness measure in this work [14]. The server utilization is fairer when Jain's index is closer to 1.

TABLE III: Fairness (Jain's index)

Trial#	D-Optimization		SU-Optimization	
	Static	RL-Dynamic	Static	RL-Dynamic
1	0.876	0.898	0.954	0.989
2	0.899	0.972	0.921	0.982
3	0.879	0.932	0.940	0.974
4	0.850	0.926	0.942	0.999
5	0.877	0.903	0.917	0.961

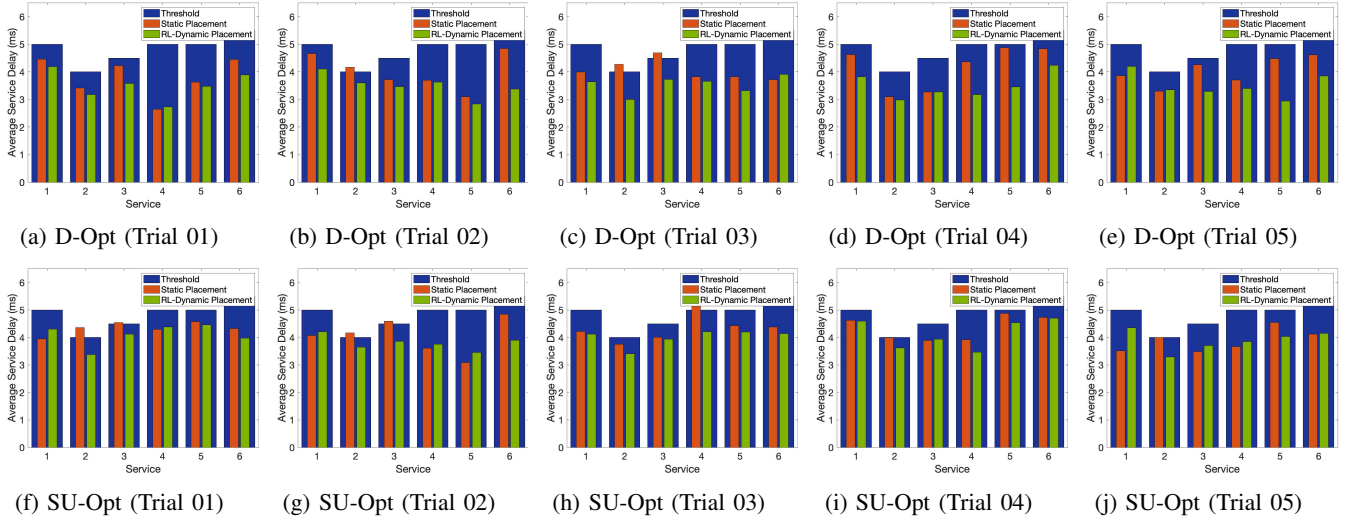


Fig. 3: Average service delay for the two optimization algorithms.

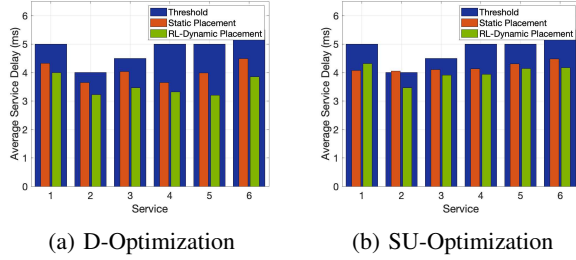


Fig. 4: Average service delay for the two optimization algorithms (Average of five trials).

From Table. III, we can observe that the proposed scheme of RL-Dynamic service placement achieves higher fair utilization of edge servers compared to the static one. When compared to the D-optimized, the SU-optimized model exhibits substantially higher fairness in server utilization. In addition, as shown in Fig. 5, the proposed SU-optimized model mitigates the load imbalance problem and spreads the load more evenly across the edge nodes, while satisfying the delay requirements. The balanced spread of service resources among different edge nodes will also help to prevent the saturation/congestion at any single server given the limited resources at the servers. Inefficient usage of resources not only results in wastage but also forces future service demands to be accessed from the network core that will incur higher delay leading to lower performance.

TABLE IV: Average edge server utilization (%)

Trial#	D-Optimization		SU-Optimization	
	Static	RL-Dynamic	Static	RL-Dynamic
1	66.83	67.89	63.85	65.08
2	65.59	66.72	64.98	65.48
3	67.45	68.48	64.27	65.29
4	70.09	68.26	64.25	65.21
5	68.37	69.20	66.23	66.82

Now, we compare the server utilization for different algorithms and optimizations, as shown in Table. IV. It is the average of the percentage of the available capacity of all servers consumed by a service. As can be observed from the table, our proposed SU-optimized placement intends to utilize edge-server resources more effectively accommodating the same demand (as carried out by D-optimized), but with lower percentage of edge server utilization. It is also worth noting that we performed multiple trials to show the confidence level of the performance.

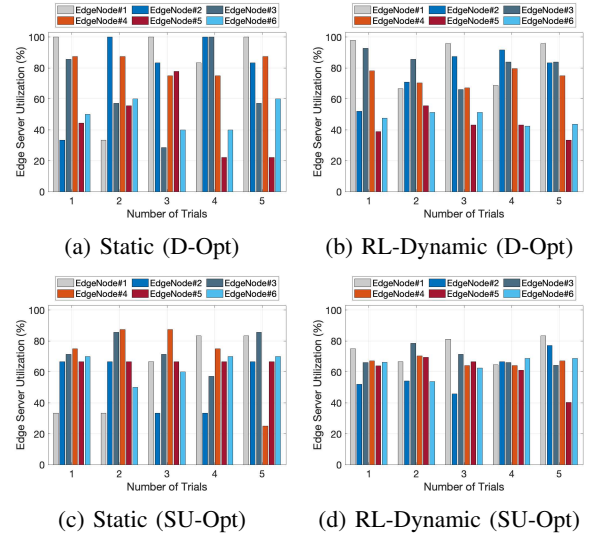


Fig. 5: Edge server utilization for the two optimization algorithms.

## VI. CONCLUSION

In this work, we addressed the problem of dynamic service placement in vehicular networks. We developed a reinforcement learning-based algorithm for continual learning of the

environment to capture the dynamicity of vehicles and varying service demands and request-types. For the decision module in our learning framework, we explored two different objective functions- minimizing the delay and minimizing the server resource utilization. We developed ILP problem formulations for the two objective functions. We evaluated our framework by simulating a virtual traffic scenario of a realistic vehicular network using SUMO. Our performance study shows that our proposed RL-based dynamic service placement achieves higher fairness in utilization of edge server nodes and low service delay compared to the static one. When compared to D-optimized decision, the SU-optimized decision utilizes resources more effectively balancing the load across edge servers, and achieving higher fairness with lower edge server utilization. As a future work, we plan to extend this work considering attack scenarios and a more complex architecture of edge networks.

- [14] R. Jain, D. Chiu, and W. Hawe, *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System*. Eastern Research Laboratory, Digital Equipment Corporation, 1984. [Online]. Available: <https://books.google.com.sg/books?id=M2QLGwAACAAJ>

## REFERENCES

- [1] M. Series, *IMT Vision—Framework and overall objectives of the future development of IMT for 2020 and beyond*. Recommendation ITU, 2015, vol. 2083.
- [2] 3GPP, *Study on management and orchestration of network slicing for next generation network (Release 15)*. Document TR 28.801, 2018.
- [3] ETSI, *Multi-Access Edge Computing (MEC); Study on MEC Support for V2X Use Cases*. European Telecommunications Standards Institute, ETSI GR MEC 022 V2.1.1, Sept 2018. [Online]. Available: <http://www.etsi.org>
- [4] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, “Survey on multi-access edge computing for internet of things realization,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.
- [5] A. Haydari and Y. Yilmaz, “Deep reinforcement learning for intelligent transportation systems: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–22, 2020.
- [6] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, “Edge-enabled V2X service placement for intelligent transportation systems,” *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.
- [7] I. Shaer, A. Haque, and A. Shami, “Multi-component V2X applications placement in edge computing environment,” *arXiv*, 2020.
- [8] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, “Toward cloud-based vehicular networks with efficient resource management,” *IEEE Network*, vol. 27, no. 5, pp. 48–55, 2013.
- [9] Y. Lin, J. Hu, B. Kar, and L. Yen, “Cost minimization with offloading to vehicles in two-tier federated edge and vehicular-fog systems,” in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–6.
- [10] H. Yao, C. Bai, D. Zeng, Q. Liang, and Y. Fan, “Migrate or not? exploring virtual machine migration in roadside cloudlet-based vehicular cloud,” *Concurrency and Computation: Practice and Experience*, vol. 27, no. 18, pp. 5780–5792, 2015. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3642>
- [11] X. Yu, M. Guan, M. Liao, and X. Fan, “Pre-migration of vehicle to network services based on priority in mobile edge computing,” *IEEE Access*, vol. 7, pp. 3722–3730, 2019.
- [12] C. Bockelmann, N. K. Pratas, G. Wunder, S. Saur, M. Navarro, D. Gregoratti, G. Vivier, E. De Carvalho, Y. Ji, C. Stefanovic, P. Popovski, Q. Wang, M. Schellmann, E. Kosmatos, P. Demestichas, M. Raceala-Motoc, P. Jung, S. Stanczak, and A. Dekorsy, “Towards massive connectivity support for scalable mMTC communications in 5G networks,” *IEEE Access*, vol. 6, pp. 28 969–28 992, 2018.
- [13] ETSI, *5G; Study on scenarios and requirements for next generation access technologies*. European Telecommunications Standards Institute, ETSI TR 138 913 V15.0.0, 2018. [Online]. Available: <http://www.etsi.org>