Adaptive Deep Reinforcement Learning Approach for Service Migration in MEC-Enabled Vehicular Networks

Sabri Khamari, Abdennour Rachedi, Toufik Ahmed, Mohamed Mosbah

Adaptive Deep Reinforcement Learning Approach for Service Migration in MEC-enabled Vehicular Networks

Sabri Khamari
Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 -Talence, France
sabri.khamari@u-bordeaux.fr

Abdennour Rachedi
Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 -Talence, France
abrachedi@u-bordeaux.fr

Toufik Ahmed
Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 -Talence, France
tad@labri.fr

Mohamed Mosbah
Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 -Talence, France
mohamed.mosbah@u-bordeaux.fr

***Abstract*— Multiaccess edge computing (MEC) has emerged as a promising technology for time-sensitive and computation-intensive tasks. However, user mobility, particularly in vehicular networks, and limited coverage of Edge Server result in service interruptions and a decrease in Quality of Service (QoS). Service migration has the potential to effectively resolve this issue. In this paper, we investigate the problem of service migration in a MEC-enabled vehicular network to minimize the total service latency and migration cost. To this end, we formulate the service migration problem as a Markov decision process (MDP). We present novel contributions by providing optimal adaptive migration strategies which consider vehicle mobility, server load, and different service profiles. We solve the problem using the Double Deep Q-network algorithm (DDQN). Simulation results show that the proposed DDQN scheme achieves a better tradeoff between latency and migration cost compared with other approaches.**

Keywords*— Multi-access edge computing, vehicular network, service migration, deep reinforcement learning.

I. INTRODUCTION

The integration of connected vehicles and autonomous driving technology is poised to fundamentally transform transportation systems, yielding incalculable advantages for our society [1][2]. Intelligent Transportation Systems (ITS) have the potential to enhance transportation safety through effective vehicle coordination and resource management [3]. Along with safety, ITS will offer entertainment services such as video streaming and gaming and may even extend to in-vehicle augmented reality [4][5]. For these capabilities to be realized, vehicles need to have the ability to communicate and access services with minimal latency. Thus, the ITS environment must meet these requirements. Therefore, novel strategies that are sensitive to mobility and resource constraints are required to diminish latency for applications and optimize the utilization of network and computing resources [6].

To satisfy such rigorous quality of service (QoS) requirements, the Multi-access Edge Computing (MEC) paradigm has been proposed by deploying servers at the edge of the network. Efficient deployment of edge servers provides robust computational capabilities to vehicles while ensuring a low latency [7][8]. As a result, applications that demand high computational power such as object detection, video stream analytics, and path navigation can be executed with edge servers in vehicular networks [9]. Although MEC-enabled vehicular networks offer numerous advantages, they present new challenges in task offloading and computing. A crucial challenge is the high mobility of vehicles which leads to a highly dynamic communication topology in vehicular networks resulting in unreliable communication links [10]. Vehicles may move beyond the coverage area of an edge server resulting in increased latency and the interruption of a service session between the vehicle and the edge server hosting this service.

To address these challenges, we investigate the service migration problem in MEC-enabled vehicular networks. To maintain the advantages of MEC, the user's service may require migration as they move between various geographical locations. The question is when and where to migrate the service. However, frequent service migrations can result in high migration costs, including increased backhaul load [11]. Therefore, it is crucial to explore solutions to the service migration problem that strikes a tradeoff between maintaining QoS and minimizing migration costs.

In this paper, we formulate the service migration problem as a Markov decision process (MDP). We present novel contributions by providing optimal migration strategies for different service profiles. Based on the service's performance requirements, its required computing capacity, and its size, an optimal tradeoff between the migration cost and the latency is determined. In order to solve this problem efficiently, we use deep reinforcement learning (DRL) techniques, specifically, deep Q networks (DQN). The proposed DRL-based migration scheme ensures service continuity under high mobility constraints and offers optimal latency for each service profile and a reduced migration cost. The proposed scheme executes proactive service migration while considering the mobility of vehicles, the necessary amounts of computational, and the service profile.

To summarize, the main contributions of this paper are synthesized as follows:

- We approach the problem of service migration by formulating it as a Markov decision process (MDP),

where we meticulously define the states, actions, and reward function. By doing so, we can effectively determine the optimal time and location for migrating the service while the vehicles are in motion. Different from the previous approaches to service migration, our model considers the type of service. A customized migration strategy is adopted for each service profile.

- We employ DRL techniques to provide an efficient MDP model solution. Specifically, we propose a deep Q learning (DQL)-based solution that utilizes a double Q network and a replay buffer to enhance the learning outcomes.
- We evaluate the performance of the proposed DRL-based scheme and compare it to other service migration approaches.

The remainder of this paper is organized as follows. In Section II, recent research works on service migration are reviewed. The system model is presented in Section III. Then, the problem formulation is elaborated in Section IV. After that, Section V presents the simulation and evaluation of the solution. Finally, the paper is concluded in Section VI.

II. RELATED WORK

In recent years, researchers have proposed many strategies for service migration. Authors in [12] propose the "always migration" scheme so that the service always migrates to the nearest MEC server. However, the migration costs are neglected. Work in [13] proposes an MDP-based model that considers the distances between mobile users and edge servers as the states, associates each state with an action (migrate or not), and defines the transition probabilities between two states with a specific action and the rewards. Thus, a model based on MDP is proposed to address the service migration issue. However, this work does not consider other parameters besides the distance between the user and the service to decide the optimal migration. Peng et al. [14] studied the dynamic migration decision in the vehicular network, which jointly considers the QoS and migration cost. However, this migration strategy determines whether to migrate a service or not without considering where to migrate the service, so the service is migrated to the nearest edge server. Abouaomar et al. [15] studied the problem of service migration in a MEC-enabled vehicular network to minimize the total service latency and migration cost. They modeled the problem as a multi-agent Markov decision process and solved it by leveraging deep Q learning (DQL). Nevertheless, this paper considers some context conditions as vehicle positions, and their velocities; but it didn't consider the load of edge servers.

In this paper, we present novel contributions by providing optimal migration strategies while considering the available capacity on each edge server. Our model provides a personalized migration strategy for each service profile. To the best of our knowledge, no work considered the difference between the service's profile in the definition of the service migration strategy.

III. SYSTEM MODEL

We consider a vehicular MEC-enabled architecture covered with a set of Roadside units (RSU), each equipped with an Edge server $n \in \mathcal{N} := \{1, 2, \dots, N\}$ as illustrated in Figure 1. There are K mobile users (or vehicles, interchangeably) requesting services from the Edge servers. Each vehicle requests some service to fulfill its requirements. Services are identified with a service profile that describes the service class, the required capacity (computing and memory), and the service image size.

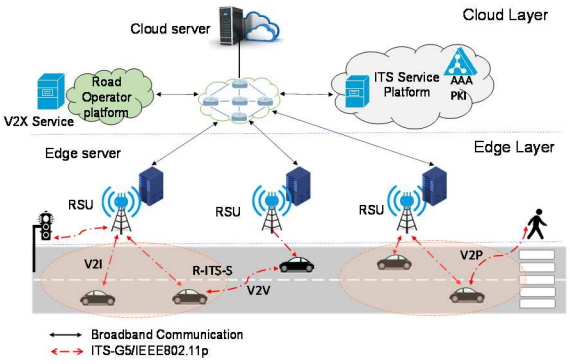

Figure 1. Illustration of the system architecture [16]

We adopted the service classification presented in [17] to identify three classes of services:

- Class 0 (used in Service Profile SP0): regroup highest priority services that need stringent requirements of low latency.
- Class 1 (used in Service Profile SP1): regroup second priority services with less stringent requirements than the first class.
- Class 2 (used in Service Profile SP2): regroup the lower priority services with nonspecific stringent requirements of latency.

IV. PROBLEM FORMULATION

In this section, we formulate the service migration problem as a Markov Decision Process (MDP). Then we present Double Deep Q Network (DDQN) method as an effective solution to the problem.

A. The MDP formulation

In reinforcement learning, the environment is formulated as an MDP which is represented as (S, A, P, R, γ), where S and A are state space and system actions, respectively. P represents the probability of each state transition when choosing an action $a \in A$ as $P(s' \mid s, a) : S \times A \times S \rightarrow [0,1]$, where $s \in S$ and $s' \in S$ is the current state and the next state respectively. An immediate reward R is obtained from this state transition as $R(s, a, s')$: $S \times A \times S \rightarrow \mathbb{R}$. In addition, $\gamma \in [0,1]$ is the discount factor used to balance immediate and long-term rewards. At each time slot, the service provider is responsible for obtaining the state from vehicles and the MEC servers. Then, it transmits this accumulated state to the DQL and receives feedback regarding the optimal action for determining whether to migrate at this time and to which edge server.

The action is then carried out, and the system shifts to the next state. The key elements of the MDP are defined as follows:

1. State space:

We define the state space $S = \{s_t \mid t = 1, 2, \ldots, t_{Max}\}$ where a state at time slot t s_t is a 4-tuple given by:

$$s_t = \{d_t[K], H_t, C_t[K], SP_t\} \quad (1)$$

Where:
d_t[K]: The distance between the vehicle and the K edge Servers at time slot t where K is the number of the edge servers.
H_t: The Edge server hosting the vehicle's service at time slot t.
C_t[K]: Available capacity on the K edge server (CPU, memory, disk).
SP_t: Service Profile:

- Service class (class 0, class 1, or class 2).
- Required capacity by the service.
- Service Image Size.

This definition of state space enables our model to function as a multi-criteria migration strategy. It takes into account the distance between the vehicle and each edge server in each time interval, and thus the vehicle's velocity implicitly. Also, it considers the load of edge servers and the required capacity of migrated service. And more importantly, our model adapts its migration strategy according to the service class.

2. Action space:

An action represents the decision to migrate a service to an Edge server e_i at time slot t. therefore, an action $a_t = \{A_1, A_2, \ldots, A_n\}$, where A_i is the action of migrating the service to Edge server e_i at time slot t, $A_i \in \{0,1\}$ and $\sum A_i = 1$. No migration will occur during time slot t if Hj is the hosting edge server and Aj=1. This action space representation allows our model to decide where and when a service should be migrated.

3. Reward function:

A MEC agent chooses an action a_t at time slot t and receives a reward R_t. Since we seek to reach optimal latency for each service profile and a reduced migration cost, we define the reward function as a combination of latency and migration cost as follows:

$$R_{a_t}^{s_t} = -\big((1 - \omega)D(t) + \omega Cost(t)\big) \quad (2)$$

Where D(t) is the sum of communication delay, backhaul delay, and computing delay. And Cost(t) is the migration cost C_v of service v at time slot t. ω is the weight factor, and $\omega \in [0,1]$.

B. The proposed DDQN

DQN is a sample and efficient Deep Reinforcement learning (DRL) algorithm. DQN approximates the Q-values *Q(s, a, θ)* of each state-action pair *(s, a)* using a Deep Neural Network (DNN), where θ represents the parameters of the Q-network. The training process of the DNN uses the experience replay memory mechanism by periodically storing MEC agent experience in a replay buffer. This experience consists primarily of the current state, the next state, the selected action, and the resulting reward. The experience replay memory mechanism provides uncorrelated data as inputs, thereby eliminating undesirable temporal correlations. The DDQN is used to make the training process faster and more reliable by using two DNNs [18]. The first DNN is called the main Q-network which is used to calculate the Q-values. And the second DNN is called the target Q-network which is used to provide the target Q-values $Q(s, a, \theta^-)$ to train the parameter θ of the main Q-network. The training phase of our proposed DDQN is presented in Algorithm 1.

In each episode, the training continues for several time slots (or steps). Each MEC agent observes the current state of its environment and selects an action a_t from its action space at each step. The MEC agent utilizes the ϵ-greedy policy to select an action. This policy selects an action randomly with probability ϵ. The MEC agent receives its reward and moves to the next state. The obtained experience is stored in the replay buffer which is used to create a training dataset. The latter is used to perform the training process in order to minimize the loss function given by:

$$Loss(\theta) = (1/j) \sum [y_j - Q(s_j, a_j, \theta)]^2 \quad (3)$$

Where $Q(s_j, a_j, \theta)$ is the Q-value of action a_j given in the state s_j which is calculated using the main Q-network with parameter θ. And y_j is the target Q-value which is calculated using the target Q-network with parameter θ^-.

Algorithm 1: Deep Q-learning algorithm for service migration

```
Initialize main DQN Q(s, a, θ) with random weights θ ;
Initialize Target DQN Q(s, a, θ̄) with weights θ̄ ← θ ;
Initialize replay memory D to capacity N ;
for each episode do
    Observe the initial state s0 stat : Servers states, Vehicle-server distance ...;
    for each time slot t do
        The controller acquires information about vehicle, Service, servers;
        if random number < ϵ then
            Select action a_t = argmax_a Q( s_t, a_t, θ) ;
        else
            Randomly select a_t ;
        end
        Execute action a_t , observe reward r_t and next state s_t + 1 ;
        Store the tuple < s_t, a_t, r_t, s_{t+1} > in D ;
        Randomly Sample a minibatch of tuple {< s_j, a_j, r_j, s_{j+1} >}_{j=1}^{J} from D ;
        for each tuple j=1..J do
            if episode terminate at step j+1 then
                y_j = r_j ;
            else
                y_j = r_j + λ argmax_a Q(s_{j+1}, a_{j+1}, θ̄) ;
            end
        end
        Perform a gradient decent step on Loss(θ) with respect to network parameters θ ,
          where loss function is Loss(θ) = (1/j) Σ[y_j − Q(s_j, a_j, θ)]^2 ;
        Update θ̄ = θ after K steps ;
        Terminate when the vehicle is out of simulation region;
    end
end
```

To update the parameter θ of the main Q-network, the agent performs a gradient descent step. Finally, the parameter θ^- is updated after each K steps by copying the parameter θ of the main Q-network.

The second phase is the inference phase of the DQL. Once the trained DDQN is obtained, the agent uses its optimal parameters to find the optimal migration strategy. The MEC

agent observes the current state at each step and chooses the action that maximizes the Q-value based on the trained DDQN. Thus, we find the solution to the migration problem, and we can determine when and where to migrate a service to achieve an optimal balance between latency and cost.

V. Simulation And Evaluation

A. Simulation Envirenment

The simulation experiments are based on OMNeT++ [19] and the Artery framework [20] to simulate the simulate movement of the vehicles and the communication between the vehicles and the RSUs, on the one hand, and the communication between the RSUs and the edge servers, on the other. The implementation of the DDQN agent is performed using Python and Keras.

We consider a MEC-enabled vehicular network where five Edge servers are deployed over a highway as depicted in Figure 2. The distance between each two neighbor servers is assumed the same. The RSUs are deployed in every 800m and each RSU is attached to the nearest Edge server. The length of the highway is 30 km. The vehicles move with a randomly-chosen fixed speed from the range of [60, 110] km/h.

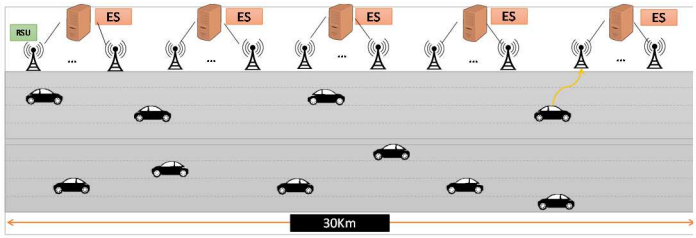

Figure 2. Simulation scenario

The DDQN and vehicular network parameters are presented in Table 1.

Table 1. Simulation parameters

Parameter	Value
Number of Edge servers	5
RSU Transmission power	47,9 mW
Vehicles transmission power	200 mW
Access Technology	IEEE 802.11p
Learning rate	5e -4
Learning rate	0,99
Replay memory size	1000000
Mini-batch size	64
Target update interval	100
Optimizer	Adam
RNN hidden layers	Two hidden layers of 256 neurons each.
Activation function	ReLU

B. Simulation Results and Analysis

Figure 3 illustrates the average reward per episode. It is clear that the reward improves with the training episodes. This shows the effectiveness of our proposed DQL algorithm. We notice that the DDQN algorithm converges at approximately 190 episodes. In other words, the corresponding MEC agent converges to a good learning outcome, which implies that it will explore better actions.

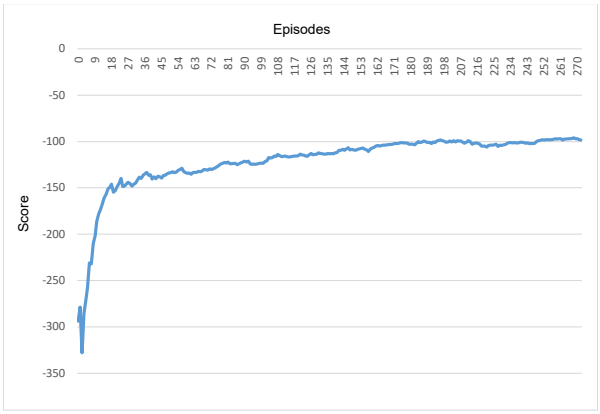

Figure 3. The training rewards for MEC agent

Our proposed service migration strategy is compared with the composite of the following approaches:

- Random migration (RM): the service is migrated randomly at each time slot t.
- Never migrate (NM): the service is hosted in one of the edge servers and does not migrate during the simulation.
- Always migration (AM): the migration schema proposed in [12], considers the migration of the service to the nearest edge server. We considered also the load of the edge server, if the nearest server is at 100% of capacity the migration doesn't occur.
- Peng et al. approach [14]: a state-of-the-art DRL-based service migration strategy. In this work, the different service profiles are not considered.

The simulations were carried out under different configurations of service profiles (class of service). As explained earlier, we consider three service profiles: SP0, SP1, and SP2 where SP0 represents the highest priority services that need stringent requirements of low latency.

Figure 4 represents the migration cost results for a service size of 5. (unit). The NM method is excluded from this comparison (Migration cost equal to zero). The results indicate that our DDQN method has the lowest migration cost compared to other methods. In SP0, the service's latency requirements are stringent, so our model migrates the service more frequently to a more effective edge server near the vehicle. The approach of Peng et al. provides marginally lower costs than our model in SP0. In SP1 and SP2, however, our model's migration costs outperform those of other approaches. The cost of migration is reduced in SP1 and SP2 because services in these service profiles are less latency-sensitive. So we can reduce migration while maintaining the QoS of these services.

Figure 5 illustrates the average service latency under the same configuration. The results show that the latency of our model increases in SP1 and SP2 compared to SP0. This is justified by the fact that SP1 and SP2 services are tolerant of higher latencies compared with SP0 which has a stringent requirement of low latency. However, as depicted in Figure 4, this increased latency allowed us to minimize the migration cost. In other words, our model can be adapted to the requirement of each service in order to reach an optimal balance

between migration cost and latency. Compared to the other approaches, our DDQN model outperforms RM, NM, and Peng et al. when the service is under SP0. AM offers the lowest latency because the service is always migrated to the nearest edge server, but the migration cost is significantly higher.

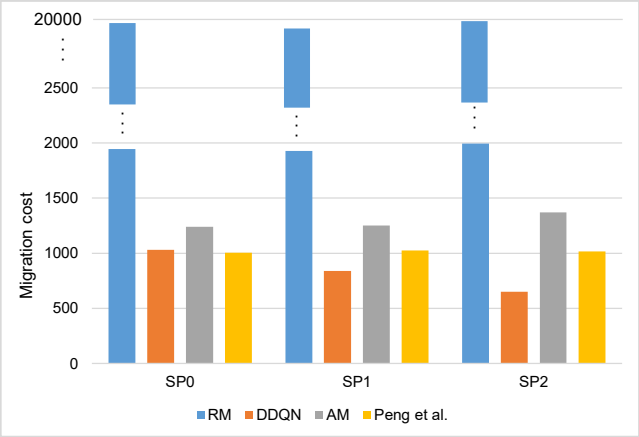

Figure 4. Migration cost results

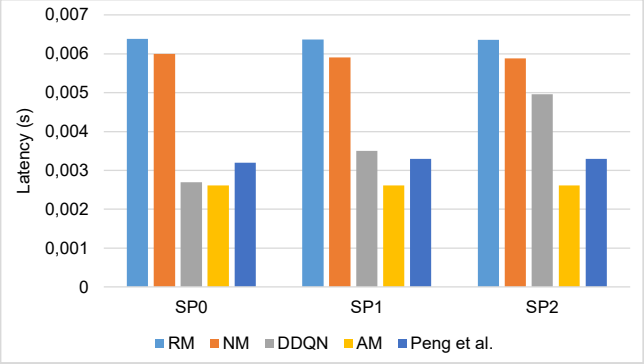

Figure 5. Latency results

VI. CONCLUSION

In this paper, we proposed a DRL-based scheme to solve the problem of vehicular service migration. The problem was initially formulated as a Markov Decision Process (MDP). This system takes into account various constraints, including vehicles' mobility, servers' capacity, and most importantly, the service profile. Considering this service profile enables the model to provide an adaptable migration strategy according to the service's requirements. We propose a deep Q learning (DQL)-based solution that utilizes a double Q network and a replay buffer to enhance the learning outcomes. Finally, we have demonstrated through extensive simulations that the proposed DQL algorithm, depending on the requirements of the service profile, achieves the best tradeoff between latency and migration cost compared to other strategies. In future work, we will expand our solution to a more realistic 2D scenario and transform the model into a multi-agent MDP to improve model training.

REFERENCES

[1] Lu, H., Liu, Q., Tian, D., Li, Y., Kim, H., and Serikawa, S. (2019). The Cognitive Internet of Vehicles for Autonomous Driving. IEEE Network, 33(3):65–73.

[2] Coutinho, R.W. and Boukerche, A. (2019). Guidelines for the Design of Vehicular Cloud Infrastructures for Connected Autonomous Vehicles. IEEE Wireless Communications, 26(4):6–11.

[3] Lin, Y., Wang, P., & Ma, M. (2017, May). Intelligent transportation system (ITS): Concept, challenge and opportunity. In 2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids) (pp. 167-172). IEEE.

[4] S.Wang, V. Charissis, J. Campbell,W. Chan, D. Moore, and D. Harrison, "An Investigation Into the Use of Virtual Reality Technology for Passenger Infotainment in a Vehicular Environment," in Proc. IEEE Int. Conf. Adv. Mater. Sci. Eng. (ICAMSE), 2016, pp. 404–407.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[5] H. Khan, S. Samarakoon, and M. Bennis, "Enhancing Video Streaming in Vehicular Networks via Resource Slicing," IEEE Trans. Veh. Technol., vol. 69, no. 4, pp. 3513–3522, 2020

[6] Aljeri, N. and Boukerche, A. (2019). Fog-enabled vehicular networks: A new challenge for mobility management. Internet Technology Letters.

[7] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," IEEE Internet Things J., vol. 6, no. 5, pp. 7635–7647, Oct. 2019.

[8] Khamari, S., Ahmed, T., & Mosbah, M. (2022, December). Efficient Edge Server Placement under Latency and Load Balancing Constraints for Vehicular Networks. In GLOBECOM 2022-2022 IEEE Global Communications Conference (pp. 4437-4442). IEEE.

[9] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," Proc. IEEE, vol. 108, no. 2, pp. 246–261, 2020.

[10] Lopes, D., & Sargento, S. (2014, June). Network mobility for vehicular networks. In 2014 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-7). IEEE.

[11] Peng, Y., Liu, L., Zhou, Y., Shi, J., & Li, J. (2019, December). Deep reinforcement learning-based dynamic service migration in vehicular networks. In 2019 IEEE Global communications conference (GLOBECOM) (pp. 1-6). IEEE.

[12] W. Bao, D. Yuan, Z. Yang, S. Wang, W. Li, B. Zhou and A. Zomaya, "Follow Me Fog: Toward Seamless Handover Timing Schemes in a Fog Computing Environment," IEEE Communications Magazine, pp. 72-78, 2017.

[13] A. Ksentini, T. Taleb, and M. Chen, "A markov decision processbased service migration procedure for follow me cloud," in Proceedings of the IEEE International Conference on Communications (ICC), pp. 1350–1354, IEEE, 2014.

[14] Peng, Y., Liu, L., Zhou, Y., Shi, J., & Li, J. (2019, December). Deep reinforcement learning-based dynamic service migration in vehicular networks. In *2019 IEEE Global communications conference (GLOBECOM)* (pp. 1-6). IEEE.

[15] Abouaomar, A., Mlika, Z., Filali, A., Cherkaoui, S., & Kobbane, A. (2021, October). A deep reinforcement learning approach for service migration in mec-enabled vehicular networks. In *2021 IEEE 46th conference on local computer networks (LCN)* (pp. 273-280). IEEE.

[16] Khamari, S., Ahmed, T., & Mosbah, M. (2022, December). Efficient Edge Server Placement under Latency and Load Balancing Constraints for Vehicular Networks. In *GLOBECOM 2022-2022 IEEE Global Communications Conference* (pp. 4437-4442). IEEE.

[17] Maaloul, S., Aniss, H., Kassab, M., & Berbineau, M. (2021). Classification of C-ITS services in vehicular environments. *IEEE Access*, *9*, 117868-117879.

[18] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in Proc. AAAI Conf. Artif. Intell., vol. 30, no. 1, 2016.

[19] VARGA, OMNeT++ Discrete Event Simulation System, Release 5.6, 2020.

[20] Riebl, Raphael, Obermaier, Christina, et Gunther, Hendrik-Jörn. Artery: Large Scale Simulation Environment for ITS Applications. In: Recent Advances in Network Simulation. Springer, Cham, 2019. p. 365-406