Received 15 July 2024; accepted 11 August 2024. Date of publication 14 August 2024; date of current version 23 August 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3443514

DRL-Based Availability-Aware Migration of a MEC Service

ANNISA SARAH, GIANFRANCO NENCIONI, AND MD MUHIDUL ISLAM KHAN

Department of Electrical Engineering and Computer Science, University of Stavanger, 4036 Stavanger, Norway

CORRESPONDING AUTHOR: A. SARAH (e-mail: annisa.sarah@uis.no)

This work was supported by the Norwegian Research Council through the 5G-MODaNeI Project under Grant 308909.

ABSTRACT Multi-access Edge Computing (MEC) allows a mobile user to access a service on a computing device called MEC Host (MEH), enabling lower latency by running the service closer to the users. When the user moves away from the serving MEH, the latency increases, which may cause a disruption of the user experience and of the service continuity. Moreover, the serving MEH may also fail, making the service unavailable. We propose a solution to a service migration problem that maximizes the MEC service availability by jointly deciding (i) migration timing and (ii) target MEH based on latency constraint, resource constraint, and availability status of a MEH. We solve the problem by using Deep Reinforcement Learning (DRL). The experiment shows that our proposed solution can successfully maintain a high service availability (more than 94%) in the presence of different failure probabilities, while another algorithm gives unstable service availability.

INDEX TERMS Edge computing, 5G, service migration, reinforcement learning.

I. INTRODUCTION

MULTI-ACCESS Edge Computing (MEC) is a technology that provides computing and storage resources close to the users. When integrated with the fifth generation (5G) of mobile networks, MEC enables the delivery of Ultra-Reliable Low-Latency Communication (URLLC) services. The integration presents two advantages: (i) Users can access greater computing power than their limited User Equipment (UE), while (ii) they experience reduced end-to-end latency compared to cloud-based services. One of the use cases that benefit from MEC is the Intelligent Transportation Systems (ITS). Some examples of ITS application are: video analytics for traffic management [1], autonomous driving [2], and smart ambulance [3].

A MEC system includes multiple computing platforms at the network edge known as *MEC Hosts* (MEHs), which provide computing, storage, and network resources to run *services* [4], [5]. A service is a MEC application instance hosted by a MEH in response to a request from a UE [4]. Unlike cloud systems that rely on centralized data centers, MEHs are geographically distributed to reduce service response times by being closer to the UEs [6]. A MEC also includes a MEC Orchestrator (MEO), which manages the overall MEC system, including deciding which MEH should provide a service to a UE. In the 5G-MEC context, a UE requests a service with specific requirements, such as maximum latency or required resources [4]. Intuitively, a UE's service should be allocated to the nearest MEH, as described in [7]. The nearest MEH is usually chosen due to the following reasons. When the UE moves away from the Serving-MEH (S-MEH), the service may become unavailable because the increased end-to-end latency may violate the latency requirement. To ensure service continuity, it is crucial to migrate the service to an appropriate MEH at the right moment. This decision is influenced by UE mobility, which may impact, together with other causes, the service performance. An appropriate MEH means a MEH that can fulfil the latency requirement and resource requirements of the requested service. During MEH operation, a MEH can experience failures due to different causes, such as hardware failures, security attacks, and system overload. When the MEH that is serving a UE fails, the UE's service may become unavailable. Given that typical MEC applications are latency sensitive, we assume that the failure of the S-MEH leads in service unavailability, as the MEH failure would surely lead to the violation of the latency constraints. In summary, service is down due to the violation of service requirements, mainly caused by UE mobility or MEH failure.

In this context, the MEO is responsible for maintaining service continuity by deciding (i) the best time to migrate the service and (ii) the MEH where the service should be migrated. To make these decisions, the MEO monitors the end-to-end latency between the UE and the S-MEH, the spare resources capacity of each MEH, and the MEH status (up or down) [4], [8].

Current works on the migration of a MEC service have very different assumptions, different objectives, and take different decisions [9], [10], [11], [12]. For example, some works focus on deciding the Target-MEH (T-MEH) selection [9], while others focus on deciding the migration time, i.e., when to migrate [11]. Both works assume that the other decision is taken using a simple strategy, e.g., location-aware threshold-based policy. Moreover, the various works consider only network resources (such as latency) [13] or computing resources (such as processing capacity and storage) [14]. Finally, only a few works have a target for the maximization of service availability, often expressed as minimization of service downtime [15]. Many of these works have a simple definition of service downtime and a very simple model of the MEH failure.

This work addresses the problem of service migration in a 5G-MEC system. We formulate a problem that jointly decides the time to perform the migration and the T-MEH, and this decision is based on the information the MEO has according to ETSI on the network computing resources and MEH status. We aim to minimize service downtime, defined as when the service requirements of MEC service are not satisfied.

To address this problem, we use Reinforcement Learning (RL) due to its capability to handle dynamic scenarios, which are characterized by informational uncertainty. We propose the Availability-aware Service Migration (ASM) algorithm based on Deep Reinforcement Learning (DRL), specifically Deep Q-learning. In contrast to the previous approaches [11], we define the attributes of the Deep Q-learning to analyze the service unavailability, which may be caused by the latency requirement violation or the MEH failure. We have selected Deep Q-Learning for its reduced complexity, simplicity, and ease of implementation relative to actor-critic models. In evaluating our proposed algorithm, we incorporate a model for MEH failure, where the probability of failure depends on the usage of MEH over time.

In summary, we aim to minimize the long-term average service downtime, subject to the service requirements and capacity constraints. Compared to recent studies, our contributions are as follows.

1) We formulate a service migration problem that:
   a) jointly decides *migration timing* and *T-MEH* based on the service performance monitored by the MEO;
   b) *maximizes the service availability* by minimizing the service downtime, which is caused by a service requirement violation under the uncertainty of the UE mobility, network condition, and MEH status.
2) We propose ASM, a DRL-based algorithm, to solve the problem.
3) We provide an extensive evaluation of the proposed solution.

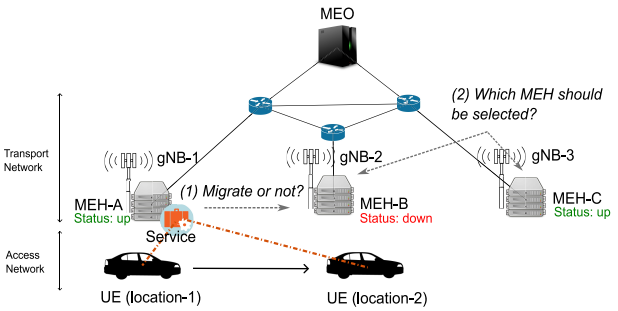


FIGURE 1. Example of MEC service migration problem.

The paper is organized as follows. We describe the addressed problem and the underlying assumptions in Section II. This is followed by presenting the related works in Section III. Then, we present the actual formulation of the problem in Section IV. Next, we introduce our proposed solution to the problem in Section V. The system model for the simulation setting is described in Section VI. We then outline the evaluation scenario and discuss the results in Section VII. Finally, we draw our conclusions in Section VIII.

II. PROBLEM DESCRIPTION

Fig. 1 illustrates an example of a service migration problem in an integrated 5G and MEC system. We consider that the UE is connected to a gNodeB (gNB) and requests a service that runs on a MEH. A MEO manages the overall MEC system and makes two decisions: (1) when to migrate the service and (2) which T-MEH should be selected. The decisions are based on the observed parameters, such as end-to-end latency and MEH failure status.

A. 5G NETWORK & MEC SYSTEM

In ETSI MEC, the MEO can access all the information on the MEC system, including the MEH status (up or down), resources capacity, workload, running services and network performance of connectivity between MEHs [4]. The information is securely exchanged between the MEO and MEHs [16]. The MEH can be down when one or some of the MEH components fail, or when the MEH is attacked by adversaries [17], [4]. A discrete *timestep* scenario is considered, meaning that the MEO observes and decides only on each timestep.

1) MEH RESOURCES

A MEH has *computing power* (Unit of Measure - UoM: virtual Core Processor Unit, vCPU) and *memory* (UoM: Gigabyte, GB) to serve the service request. We focus only on computing power and memory because they are the main attributes considered in the application descriptor from ETSI [18]. Each MEH has a fixed capacity for each resource throughout the entire period. Nevertheless, the actual available resources can fluctuate over time because the MEH resources are also utilized by other services. We call the available resources as *remaining processing capacity* and *remaining memory capacity.*

2) TRANSPORT AND ACCESS NETWORK

The MEO and MEHs are interconnected via the transport network. A path in the transport network has a certain data rate. Each MEH is co-located with a 5G gNB [19]. In a 5G-MEC system, two events may occur and are usually caused by the UE mobility and the variability of the network conditions: *handover* and *service migration*. The handover means that a UE changes the gNB to which it is connected. The service migration means that the service requested by a UE is transferred from the S-MEH to the T-MEH. We assume that a UE is always connected to the nearest gNB. The MEO does not decide the handover between gNBs, as the handover is usually taken care of by the 5G system. We assume that the MEC system has the intra-operator MEC application mobility support, where a user can move to another gNB that is not co-located with the S-MEH but still is served by the S-MEH [8].

3) END-TO-END LATENCY

The end-to-end latency (UoM: ms) refers to the total latency a UE perceives when accessing a service hosted on the S-MEH. The end-to-end latency is composed of the following three types of delay:

- *Network delay*: Amount of time needed to send a certain packet from the source to the destination in the access and transport network.
- *Computing delay*: Time needed to run a service request on a computing platform, i.e., MEH. The computing delay includes *processing delay* and *queue delay*. The processing delay is the time needed to execute a service, whereas the queue delay is the time between the service arriving at MEH and the service being processed by the MEH.
- *Migration delay*: Time needed to migrate a service from S-MEH to T-MEH. This component is present only in case of migration.

B. SERVICE REQUIREMENTS

A service can be hosted on a MEH in response to a user request [8]. Since we are focusing on the service migration, we assume the service has already been initially deployed. When scheduling a service, a MEO must guarantee to fulfil the service requirements over the service period to maintain the service continuity. The service requirements comprise:

- End-to-end latency
- Computing power
- Memory

We assume that each UE has a dedicated service instance; thus, the total number of UEs equals the total number of service instances. In the rest of the paper, we use the term service also when referring to a service instance for simplicity and brevity.

C. SERVICE AND MEH AVAILABILITY

The major challenge for a MEC service is to maintain the service continuity under uncertain behavior of UE, network condition, and MEC system condition. A service is considered down when it does not meet the service requirements. The violation of the service requirements may have multiple causes.

First, when a UE moves from location-1 to location-2 (see Fig. 1), the UE may perceive a higher end-to-end latency because the path between the UE and the S-MEH is getting longer, and this may increase the network delay. Second, the transport network is commonly shared with other traffic and is used to deliver other services. Therefore, the traffic and the latency on each link cannot be predicted. When a packet has been sent through a network path, one of the links may be congested, increasing the end-to-end latency. In both cases, the increased latency may violate the service latency requirement, resulting in the service being down. Third, the S-MEH or the selected T-MEH can fail. In these cases, the failed MEH cannot serve the service, and the service needs to be migrated to another MEH. This migration process increases the end-to-end latency, which may violate the service latency requirement. In this case, the service is down in the timesteps when the UE is assigned to a failed MEH and the successive timestep(s) where the migration may cause a violation of the latency requirement.

In summary, the MEO is responsible for deciding *when* to migrate a service and to *which MEH* the service needs to be migrated under the presence of uncertainty of UE mobility, network condition, and MEH status. The objective of our problem is to minimize the long-term service downtime subjected to capacity constraints.

Remarks: In this work, we focus on a single user, as we aim to investigate the service availability and the adaptive migration strategy under user mobility and the MEH failure. This approach has been taken in [15] to focus on understanding the impact of MEH failure from a user perspective. Concentrating on a single user enables a detailed analysis of the adaptive migration process and its effectiveness in maintaining service availability during MEH disruptions, providing precise insights without the added complexity of multiple users. This controlled scenario ensures that the fundamental mechanisms and challenges are well understood, forming a solid foundation for future

research. Furthermore, a migration downtime of a single service has been tested in [20], demonstrating the critical importance of minimizing service disruption to maintain user satisfaction and service continuity. In future work, we will consider the multi-user setting scenario to investigate the impact on the whole 5G-MEC system.

III. RELATED WORKS

Efficient service migration is crucial for sustaining service performance within the MEC system, particularly considering mobile users, while ensuring low latency and highly reliable connectivity, as in URLLC services. This section emphasizes the current landscape of service migration challenges and the approaches that researchers adopt, especially in considering the service availability [11], [13], [14], [21], [22] and in using RL to solve the service migration problem [11], [15], [23], [24], [25].

A. RESOURCES

As the migration decision maker, a MEO must be aware of the important attributes of the MEC system that affect the service migration strategy. Some of the migration problems consider only the computing resources of MEHs [11]. Still, most studies consider computing and network resources (e.g., latency, data-rate) [9], [10]. As discussed in the previous studies, MEO must consider computing and network resources when deciding on a service migration process. Our paper considers both resources as attributes for making the service migration decision.

B. SERVICE AVAILABILITY

In addition to the type of the resource, focusing on the service availability during the migration process is important. Service availability is influenced by factors such as the latency requirement violation [13], [21], unmet resource demands [14], or the unavailability of a MEH [11], [22]. In URLLC services, it is crucial that messages are delivered correctly and within an acceptable time to ensure minimum delay or loss. The MEC systems have improved the service availability of URLLC services, compared to only using cloud systems, as demonstrated in [26]. Studies in paper [13] and paper [21] model service reliability, which is computed as the likelihood that messages are delivered within a specific delay bound. If the overall time required to complete the offloading service exceeds a prescribed value, the service has failed. Paper [14] investigates how unmet requirements of resources, like computation and data rate, can lead to service unavailability in the MEC system. Meanwhile, paper [22] focuses on maximizing the service availability by considering failures in both physical and virtual machines. The paper [11] states that a MEH can become unavailable if its computing capacity overloads, leading to a failure of the MEH These studies highlight the prominent role of the MEO as the main controller of the MEC system to ensure service availability by overseeing the maximum latency requirement, the resource demand fulfilment, and the MEH availability. Papers [13], [21] only focus on considering the latency violation but not considering the MEH failures. Paper [14] takes latency and resource constraints as a consideration, but not MEH unavailability. Paper [22] considers MEH failure but not latency and resource requirements. Paper [11] considers all latency, resource demand, and MEH unavailability. However, the paper assumes that the unavailability of MEH is due to the exceeding capacity limits. Such failure should not have happened within the presence of MEOs that have a holistic view of the MEC system. Although MEO has the overview of the system, MEH can still be failed due to unobservable reasons such as (1) component faults or (2) security threats [17], [27]. In [15], the authors study a failure-aware migration in a MEC system by assuming a service migration to the failed MEHs as a rare event. However, the authors model the failure as a static failure probability without any causality relation. In practice, the failure rate may also depend on the usage period [28], [29]. The usage period indicates that the longer the MEH is utilized for processing services, the more the probability of failure increases, similar to the wear-out phase concept in the bathtub failure model or Weibull distribution [30]. In this paper, we address service unavailability caused by latency violation, insufficient resources, and MEH failures dependent on the usage period of service allocated to the MEH.

Among those previously discussed papers, only some papers [9], [10], [11] explicitly mentioned the service migration process, while other papers [13], [14], [21], [22] consider a dynamic service placement scenario. Dynamic service placement and service migration are closely intertwined in the context of the MEC services. However, the difference between dynamic service placement and service migration is the consideration of migration triggers and the migration process. The goal of dynamic service placement is usually to have a balance load and resource usage [22], [26], [27], while the service migration problem focused on maximizing service quality [11], [12]. In the service migration context, MEO needs to take two distinct decisions: (i) migration trigger and (ii) T-MEH selection [12]. The migration trigger involves determining the appropriate time for a service migration and the circumstances under which migration should occur. In a mobile UE scenario, most studies consider the migration timing based on the threshold or change of gNB. Threshold-based decisions for migration timing can give a ping-pong effect, meaning that the service can be migrated back and forth between S-MEH and T-MEH. This condition degrades the user experience and wastes resources [8]. In [11], the authors trigger the migration request based on the cell ID change, meaning that the MEO starts the migration procedure whenever the UE moves out from the S-MEH coverage. However, deciding when to migrate based only on the location may not be efficient, as it may not reflect the service performance where the MEH and network status changes dynamically [31]. Next, selecting an appropriate T-MEH to improve the service

performance is important. Many works focus on T-MEH selection [9], [11] with few papers deciding both migration timing and T-MEH selection [31], [32]. Our problem decides the migration timing based on the observed service performance that is dynamically changing. We decide both (1) the migration timing and (2) T-MEH selection. Compared to [31], [32] where the authors make the migration timing decision solely based on the latency requirement, we take a more holistic service performance, i.e., latency, computing resource fulfilment, and the MEH availability.

The service availability during the service migration faces specific challenges such as the *service migration downtime* [9]. Service migration downtime is the time interval when a service is unavailable to users while migrating from S-MEH to T-MEH. Recent studies [9], [32], [33] consider the service migration downtime as an additional delay. In [9], the downtime is assumed as a constant value to see how different strategies will be impacted if the downtime changes. In [33], the downtime is composed of the time of migrating service from S-MEH to T-MEH and the synchronization time on T-MEH. In [32], the authors formulate a service migration problem that targets minimizing service migration downtime. However, the downtime in [32] is due to the time spent deploying a new container on the T-MEH.

Compared to the related works in the service migration problem [9], [10], [11], [27], which mainly focus on minimizing the latency, maximizing throughput, or load balancing, we focus on maximizing service availability while relaxing the service latency as the service requirement. As long as the latency and resource requirements are met, we will have more T-MEH options to select. As our current findings, none of the related works focus on maximizing service availability as they consider availability a constraint. Our paper aims to maximize the service availability directly by putting the minimization of service downtime in the objective.

C. MEH FAILURE

Few papers study a migration strategy considering the presence of MEH failure [15], [22]. Authors of [22] model by using a random probability for physical host and virtual host failures. Authors of [15] study a failure-aware migration in a MEC system by assuming a service migration to the failed MEHs as a rare event. However, both papers model the failure using a static probability representing the likelihood of a failure occurring at any given time, independent of environmental conditions or previous states. In fact, the failure rate may also depend on the usage period [28], [29]. The usage period indicates that the longer the MEH is utilized for processing services, the more the probability of failure increases, similar to the wear-out phase concept in the bathtub failure model or Weibull distribution [30]. In our problem, we are using a more holistic approach in modeling the service availability, which is comprised of both latency violations and the unavailability of MEH.

D. RL IN MEC

RL and Artificial Intelligence (AI) are envisioned to be an integral part of the 5G and Beyond (5GB) and the sixth generation (6G) of mobile networks [34]. One of the roles of AI is data-driven resource optimization, known as *AI for Edge*. Several studies show the benefit of using Machine Learning (ML) and RL for the resource optimization, e.g., optimizing radio resource management [35] and controlling the overall MEC system, which includes the service migration [11], [15], [23], [24], [25].

Paper [23] uses RL to solve a service allocation problem that aims to minimize both service delay and energy consumption. Paper [35] maximizes service throughput for massive-IoT use cases by using RL, constrained to latency and resource requirements. A classical RL algorithm needs the whole state-action computation in a dynamic environment with many interacting parameters requiring high memory and computation resources. A Deep Neural Network (DNN) helps reinforce learning strategies to approximate state-action values and predict optimal policies. The combination of DNN and RL is called DRL [36]. Authors of [11] use DRL to minimize service latency under the presence of network and MEH failures due to overcapacity. Paper [15] investigates a single-user service availability under the occurrence of MEH failure as a rare event. Paper [24], [25] uses RL to develop a cost-aware migration policy in order to minimize generic cost (latency and service deployment). These papers demonstrate the use of RL as a novel solution in resource optimization on a MEC system.

There are various versions of RL or DRL such as Q-learning [24], Deep Q-learning [10], [11], [23], [25], or actor-critic [10], [15], [35]. We present ASM, a novel solution based on Deep Q-learning. While existing research primarily targets enhancing latency, user experience, or general costs to incentivize RL agents [10], [11], [15], [24], [37], our approach diverges by defining specific attributes within DRL framework to monitor changes in service availability. These changes could result from latency violations or MEH unavailability, thus rewarding agents based on their effectiveness in minimizing service downtime. We opted for Deep Q-Learning due to its simpler complexity, straightforwardness, and ease of implementation compared to actor-critic models. Many improvements from the classic Deep Q-learning have also been proposed, such as a dynamic exploration rate [23] or a dynamic learning rate [11]. However, we use a static rate for exploration and learning rates to understand more the impact of MEH failure on service availability. As a MEH failure is an event that rarely occur so, might be insignificant to have an adaptive rate or not.

E. NOVELTIES

In summary, we are considering both network and data resources to jointly decide the migration timing and the T-MEH. While recent works focus on minimizing service latency in the migration problem, we focus on minimizing

TABLE 1. Parameter definition.

Symbol [UoM]	Description
Sets	
$\mathcal{D}$	Set of MEHs
$\mathcal{N}$	Set of timesteps
Given variables	
δ [ms]	Latency demand
π [vCPU]	Processing demand
θ [GB]	Memory demand
$m_d^P(n)$ [vCPU]	Remaining processing capacity for $d \in \mathcal{D}$ at the timestep $n \in \mathcal{N}$
$m_d^Q(n)$ [GB]	Remaining memory capacity for $d \in \mathcal{D}$ at the timestep $n \in \mathcal{N}$.
$\zeta_d(n)$ [ms]	End-to-end latency between the MEH $d \in \mathcal{D}$ and the UE observed at the timestep $n \in \mathcal{N}$
$\hat{\zeta}_{d,d'}^{\text{MIG}}(n)$ [ms]	Expected migration delay between the S-MEH d' and the T-MEH d at the timestep $n \in \mathcal{N}$
$\beta(n)$	1 if the UE needs the service at timestep $n \in \mathcal{N}$
$\xi_d(n)$	1 if the MEH $d \in \mathcal{D}$ is up at timestep $n \in \mathcal{N}$, 0 otherwise
Unknown variables	
$x_d(n)$	1 if the service is allocated on MEH $d \in \mathcal{D}$ at time step n, 0 otherwise
$y(n)$	1 if the service is down at time step n, 0 otherwise

service downtime, which is comprised of latency violations and insufficient resources due to MEH unavailability. To the best of our knowledge, this is the first paper to focus on maximizing the service availability, which jointly considers the service latency and the presence of MEH failures. While DQN has been widely implemented to solve similar service migration problems, which is demonstrated in [11], our work is the first paper that uses DQN to be aware of service availability that is affected by both the user mobility and MEH availability status.

IV. PROBLEM FORMULATION

In this section, we introduce the problem formulation and the assumptions taken in our work.

Table 1 depicts all the variables that are used in the problem formulation. We consider that the UE connects to a gNB and requests a service that runs on a MEH $d \in \mathcal{D}$. The service has three requirements: (1) maximum latency δ (ms), (2) processing power π (vCPUs), and (3) memory θ (GB). Each MEH has a remaining computing power capacity $m_d^P(n)$ and remaining memory capacity $m_d^Q(n)$ to serve the service request, which may change at each timestep $n \in \mathcal{N}$.

The end-to-end latency $\zeta_d(n)$ between UE and the S-MEH may also change at each timestep $n \in \mathcal{N}$. The migration delay occurs if the service is migrated from the S-MEH to a T-MEH. Lastly, the computing delay depends on two factors: the CPU requirement π and the remaining computing capacity of each MEH $m_d^P(n)$.

Over the period, a UE may request a service at different time with different duration. We use the given variable $\beta(n) \in \{0, 1\}$ to denote whether the service is needed on timestep n ($\beta(n) = 1$) or not. Moreover, the status of MEHs changes. We use the given variable $\xi_d(n) \in \{0, 1\}$ to denote whether the MEH is up ($\xi_d(n) = 1$) or down. When the MEO decides to migrate a service, the migration does not instantly happen. At the beginning of the timestep $n-1$, the MEO observes all the necessary parameters to take the migration decision and T-MEH selection [4], [38]. During the timestep $n-1$, the migration decision and the T-MEH selection are made. In timestep n, the migration happens. To make a decision, a MEO observes: the end-to-end latency $\zeta_d(n-1)$ between UE and each MEH d, the expected migration delay $\hat{\zeta}_{d,d'}^{\text{MIG}}(n)$ between the S-MEH d' and each different MEH d, and the status of each MEH $\xi_d(n-1)$. In the formulation, the unknown variable $x_d(n) \in \{0, 1\}$ is equal to 1 if the service is allocated in the MEH d at the timestep n. The service is migrated if $x_d(n) \neq x_d(n-1)$ for any MEH d (in case of migration this condition would be true for both S-MEH and T-MEH). Another unknown variable $y(n) \in \{0, 1\}$ is equal to 1 if the service is down at the timestep n.

The objective is to minimize the long-term service downtime, as follows.

$$\min \sum_{n \in \mathcal{N}} y(n) \cdot \beta(n) \tag{1}$$

subject to:

$$\sum_{d \in \mathcal{D}} x_d(n) = \beta(n) \qquad \forall n \in \mathcal{N} \tag{2}$$

$$x_d(n) \leq \xi_d(n-1) \qquad \forall d \in \mathcal{D}, \forall n \in \mathcal{N} \tag{3}$$

$$y(n) = \begin{cases} 1, \text{ if } x_d(n) = 1 \ \& \ \xi_d(n) = 0 \\ 1, \text{ if } x_d(n) = 1 \ \& \ \zeta_d(n) > \delta \ \forall n \in N \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

$$\delta \geq \begin{cases} \zeta_d(n-1) + \hat{\zeta}_{d,d'}^{\text{MIG}}(n), & \text{if } d \neq d' \ \& \ x_d(n) = 1 \ \& \\ & \quad x_{d'}(n-1) = 1 \\ \zeta_d(n-1), & \text{if } x_d(n) = x_d(n-1) = 1 \end{cases}$$
$$\forall d, d' \in \mathcal{D}, \forall n \in \mathcal{N} \tag{5}$$

$$\pi \cdot x_d(n) \leq m_d^P(n) \cdot \xi_d(n-1) \qquad \forall d \in \mathcal{D}, \forall n \in \mathcal{N} \tag{6}$$

$$\theta \cdot x_d(n) \leq m_d^Q(n) \cdot \xi_d(n-1) \qquad \forall d \in \mathcal{D}, \forall n \in \mathcal{N} \tag{7}$$

Eq. (2) to Eq. (4) are the conditional assumptions that must be met in each step n. Eq. (2) describes that a service must be allocated in one MEH when the service is needed, i.e., $\beta(n) = 1$. Eq. (3) describes that the service can only be allocated to a MEH $d \in \mathcal{D}$ that has a status up at the previous step $n-1$ (information known by the MEO). Eq. (4) defines the condition of the service unavailability. The service is not available, i.e., $y(n) = 1$, when the selected MEH, i.e., $x_d(n) = 1$, is down. Regardless Eq. (3), this happens since the MEO observes the MEH status in step $n-1$, an up MEH may be down in the next step as the status is uncertain. Moreover, the service is down if the latency between the UE and the selected MEH does not

fulfill the service requirement. Eq. (5) to Eq. (7) are the constraints related to the resource requirements. Eq. (5) describes the latency constraint. The first condition applies when a migration occurs. If MEH d is selected as T-MEH ($x_d(n) = 1$), the observed latency at step $n-1$, $\zeta_d(n-1)$, plus the expected migration delay from the S-MEH d', $\hat{\zeta}^{\text{MIG}}_{d,d'}(n)$, must be less than or equal to the required latency δ. The second condition applies when no migration occurs. For both conditions, the latency considered is an estimation based on the value at the previous timestep. The actual end-to-end latency might be higher and may not meet the requirement. Eq. (6) and Eq. (7) describe the constraints related to the processing and memory capacities, respectively. The selected MEH ($x_d(n) = 1$) must have remaining processing capacity of MEH $m^P_d(n)$ and remaining memory capacity $m^Q_d(n)$ that are greater or equal to the capacity and memory demand, respectively. Note that if a MEH was down in the previous timestep ($\xi_d(n-1) = 0$), we consider that the MEH has no capacity left.

V. PROPOSED SOLUTION

This section presents the proposed method to solve our service migration problem. We first explain the RL technique, the problem representation in the Markov decision process model, and our detailed algorithm.

A. DRL

In the service migration problem, the decision at any given timestep depends on the current state of the network, which includes factors such as latency, available resources and MEH status. These factors can change rapidly and unpredictably, making the environment highly dynamic. Consequently, this idealized problem cannot be addressed purely by optimization techniques. Classical mathematical optimization techniques are typically designed to handle static problems where all variables and constraints are known in advance and remain fixed over time. These methods need a complete and fixed model of the problem, making them unsuitable for dynamic environments where the conditions change and the decisions must be adapted in real-time.

In contrast, RL is well-suited for sequential decision-making problems with changing conditions. RL learns to make decisions through interaction with the environment, adapting to new information and changing conditions in real-time [39]. To solve the problem by using an RL approach, the system must be modelled as a Markov Decision Processes (MDP) [10], [15], [37].

MDP model consists of a set of possible *states* S, a set of *actions* A, a *reward function* $R(S, A)$, and a transition probability that describes the environment, i.e., a probability associated to the transition between one state s to the next state s', given an action a. The state transition must satisfy the Markov property, meaning that it depends only on that state and not on the sequence of events that preceded it. The property can be formally written as $P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = P(s_{t+1}|s_t, a_t)$, meaning that the state transition probabilities are determined solely by the present state and action, making the process memoryless. To further simplify the mathematical expressions, we define s and a as the current state and action, s' and a' as the state and action of the next timestep.

A *Q-learning* algorithm is a classic RL algorithm that allows an *agent* to learn and find an optimal policy π based on the Q-values $Q(s, a)$ by interacting with an environment. A policy π is a strategy the agent follows to decide which action to take in each state, e.g., the Q-values. The Q-value $Q(s, a)$ is a state-action value that represents how good an action is, given a particular state. The optimal Q-values $Q^*(s, a)$ can be defined as the expected reward when taking an action a in a state s (see Eq. (8))

$$Q^*(s, a) = \mathbb{E}_{s,a}[R] \tag{8}$$

Q-learning algorithm is an off-policy algorithm that learns the optimal Q-values by interacting on with the environment, without the need to initially follow an optimal policy. The Q-value $Q^*(s, a)$ denotes the expected return (total reward), starting from state s, taking action a, which follows the optimal policy π^*. In the Q-learning context, optimal policy π^* maximize the expected cumulative reward from any state, formally depicted in Eq. (9).

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{9}$$

Based on the *Bellman Optimality Equation*, the $Q^*(s, a)$ can be computed as expressed in Eq. (10) [39], where R' is the immediate reward after taking action a in state s, s' is the next state, and γ is the discount factor.

$$Q^*(s, a) = \mathbb{E}\left[R' + \gamma \cdot \max_a Q^*\left(s', a'\right)\right] \tag{10}$$

Eq. (10) can be further derived as in Eq. (11).

$$Q^*(s, a) = \sum_{s'} P\left(s'|s, a\right) \cdot \left(\mathbb{E}\left[R'|s, a, s'\right] + \gamma \cdot \max_a Q^*\left(s', a'\right)\right) \tag{11}$$

$P(s'|s, a)$ is the probability of transition from the state s to the next state s' given the action a. The term $\mathbb{E}[R_{t+1}|s, a, s']$ represents the expected immediate reward when an action a is taken in state s and leads to the next state s'. Explicitly, the expected immediate reward can be written as $r(s, a, s')$, and consequently the Q-value can be computed as depicted in Eq. (12).

$$Q^*(s, a) = \sum_{s'} P\left(s'|s, a\right) \cdot \left(r\left(s, a, s'\right) + \gamma \cdot \max_a Q^*\left(s', a'\right)\right) \tag{12}$$

In the following, we refer to the immediate reward $r(s, a, s')$ as r for simplicity. In the Q-learning algorithm, the transition probabilities are taken care of by the sampling process, meaning that over many interactions between the agent and the environment, the sampled transition (s, a, r, s')

will reflect the underlying transition probabilities. The Q-learning update rule is based on the temporal-difference (TD) learning method, meaning the Q-value is updated using the TD error. The TD error is defined as in Eq. (13) [39]. Given the TD error equation, the update rule of $Q(s, a)$ can be derived as shown in Eq. (14). The learning rate α from Eq. (13) determines how aggressively the current Q-value is adjusted towards the new estimate derived from the new observation.

$$TD = r + \gamma \cdot \arg\max_{a'} Q(s', a') - Q(s, a) \tag{13}$$

$$\begin{aligned} Q(s, a) &\leftarrow Q(s, a) + \alpha \cdot TD \\ \therefore Q(s, a) &\leftarrow Q(s, a) + \alpha \cdot \left(r + \gamma \cdot \max_{a} Q(s', a') - Q(s, a)\right) \end{aligned} \tag{14}$$

A specific implementation of DRL is deep Q-learning, which uses a neural network to approximate the Q-value function $Q(s, a; \phi)$, where ϕ are the network weights. Deep Q-learning incorporates experience replay and the target network. The *experience replay* stores the agent experiences (s, a, r, s') in a replay buffer. This experience is randomly sampled during the training to update the network. Using a replay buffer prevents the network from only learning from immediate actions, which can reduce the correlations between each transition. A *target network* ϕ^T is used to compute the target Q-values and is periodically updated with the weights of the main network ϕ. During the training process, a target Q-value z is computed as in Eq. (15). Then, the target Q-value is used to calculate the loss function $\mathcal{L}(\phi)$ (as shown in Eq. (16)). The expectation operator in Eq. (16) means that we are averaging over a distribution of experiences (s, a, r, s'), which are sampled from the replay buffer. The loss function is further derived as the quadratic loss function, i.e., mean squared Bellman residual, as in Eq. (17).

$$z = r + \gamma \cdot \max Q\left(s, a|\phi^T\right) \tag{15}$$

$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s')}\left[(z - Q(s, a|\phi))^2\right] \tag{16}$$

$$\mathcal{L}(\phi) = \frac{1}{2} \cdot \Big(\underbrace{r + \gamma \cdot \max_{a} Q(s', a|\phi)}_{\text{target}} - \underbrace{Q(s, a|\phi)}_{\text{actual}}\Big)^2 \tag{17}$$

The loss function $\mathcal{L}(\phi)$ is then used to perform gradient descent to update the parameter ϕ of the main network. Given the learning rate α from the Q-learning algorithm, the weight of the neural network from the gradient descent is updated using Eq. (18).

$$\phi = \phi - \alpha \cdot \gamma \cdot L(\phi) \tag{18}$$

B. PROBLEM REPRESENTATION

The state, action and the reward models of the service migration problem are represented as follows:

1) *State*: The state represents the current condition of the environment, capturing the relevant information that affects the decision-making process. The states change at every timestep n. In our case, the state includes the end-to-end latency $\zeta_d(n-1)$, the MEH selection $x_d(n-1)$, MEH availability status $\xi_d(n-1)$, service downtime status $y(n-1)$, and service request status $\beta(n)$. Note that the first three parameters are for each MEH $d \in \mathcal{D}$, and the first four parameters are obtained from the previous timestep $n-1$. The state is therefore characterized by $3 \cdot |\mathcal{D}| + 2$ elements.

$$\begin{aligned} s(n) = \big\{&\{\zeta_d(n-1), x_d(n-1), \xi_d(n-1)\} \\ &\forall d \in \mathcal{D}, y(n-1), \beta(n)\big\} \end{aligned} \tag{19}$$

2) *Action*: The action defines the set of possible decisions that the DRL agent can make to transition from one state to another. At each timestep n, the agent evaluates the state condition to decide which MEH should be selected to allocate the service. If the agent selects the same MEH as the previous timestep ($x_d(n) = x_d(n-1)\ \forall d \in \mathcal{D}$), no migration occurs. If the agent selects a different MEH ($x_d(n) \neq x_d(n-1)$), this triggers service migration. Given that the service can be allocated only on one MEH (see Eq. (2)), there are $|\mathcal{D}|$ possible actions when the service is requested by the UE (i.e., $\beta(n) = 1$). Of course, if the service is not requested, no action needs to be taken ($x_d(n) = 0\ \forall d \in \mathcal{D}$). Based on the taken action, the state changes ($s \rightarrow s'$). The set of actions $\mathcal{A}$ consists solely of actions that satisfy the capacity constraints outlined in Eq. (6) and Eq. (7)). This means that each action $a \in \mathcal{A}$ meets the required capacity requirements.

$$a(n) = \{x_d(n)\ \forall d \in \mathcal{D}\} \tag{20}$$

3) *Reward*: The reward can be considered as feedback on the immediate benefit of an action taken by the agent. The goal of the DRL agent is to maximize the cumulative reward over time. The immediate reward in our problem is depicted in Eq. (21). The function $f(\cdot)$ depicts how favorable the situation is and depends on the service status on the current timestep $y(n)$, the service status on the previous timestep $y(n-1)$, and the MEH status $\xi_d(n)$. Table 2 shows the non-Boolean output of the function $f(\cdot)$ given the various Boolean inputs. The function $f(\cdot)$ gives different rewards to the learning agent, with the highest reward given when the service remains continuously operational. The most significant penalty occurs when a service fails due to MEH unavailability, as this reflects a failure in the MEO to oversee the MEC system. The output values of $f(\cdot)$ were chosen based on empirical studies from simulations, where different values were tested and adjusted to find the most effective setting, i.e., the setting that enables the agent to learn properly. Eq. (21) also include a factor that exhibits an exponential relationship with respect to the end-to-end latency $\zeta_d(n)$, which allows us to

TABLE 2. Definition of the $f(\cdot)$ function.

Input 1	Input 2 (& 3)	Output	Description
$y(n) = 0$	$y(n-1) = 0$	1000	Service continues to be up
	$y(n-1) = 1$	500	Service status changes from down to up
$y(n) = 1$	$y(n-1) = 0$ & $\xi_d(n) = 0$	-1000	Service goes down due to unavailable MEH
	$y(n-1) = 0$ & $\xi_d(n) = 1$	-500	Service goes down because of latency violation
	$y(n-1) = 1$ & $\xi_d(n) = 0$	-700	Service continues to be down due to unavailable MEH
	$y(n-1) = 1$ & $\xi_d(n) = 1$	-800	Service continues to be down because of latency violation

capture the significant impact and amplification of end-to-end latency $\zeta_d(n)$ on the resulting rewards. This exponential relationship contributes to the effectiveness of the learning algorithm to make significant reward differences between one condition to another, enabling it to focus on important regions at the state space and to adapt accordingly [11]. During the design phase, a linear relationship was initially considered for the reward equation but proved inadequate in providing a suitable reward structure for the agent to learn effectively.

$$r(n) = f(y(n), y(n-1), \xi_d(n)) \cdot 0.8^{\zeta_d(n)} \quad (21)$$

Constructing the states, actions, and rewards of a Markov decision process provides clarity regarding the dynamic of the problem and the potential impacts of different decisions. This problem representation forms a basis for using RL techniques to determine the optimal strategies.

C. AVAILABILITY-AWARE SERVICE MIGRATION (ASM)

We propose an algorithm called ASM to solve the formulated service migration problem. The pseudocode of ASM is shown in Algorithm 1. First, all the parameters and hyper-parameters are initialized. As previously described in Section V-A, DQN uses two networks: one for determining actions (policy network, Q^{π}) and another for evaluating Q-values (target network, Q^{T}) The policy network Q^{π} determines the actions, and the target network Q^{T} evaluates the Q-value. The replay memory size M determines the maximum number of stored experiences, affecting the diversity and richness of training data. The target update period N^{T} is the time interval between two consecutive updates of the target network Q^{T} to match the policy network Q^{π}. The mini-batch size B determines how many experiences are used per update, affecting the stability and efficiency of training.

For each episode, the algorithm is run on a consistent scenario, e.g., the same number of MEHs, a user moves following a specific mobility pattern and periodically visits different MEHs at designated timesteps. At each timestep, an action is taken based on the ϵ-greedy approach. The ϵ is a small value determining the trade-off between exploitation and exploration. The agent selects the action with the highest state-action value based on the current policy with probability $1-\epsilon$ and selects a random action with probability ϵ. Next, the agent receives the reward r_n and moves to the next state s_{n+1}. For clarity and brevity in the pseudocode, we use different notations for timesteps compared to the rest of the paper. For instance, r_n is used instead $r(n)$. Then we store each transition as a tuple (s_n, a_n, r_n, s_{n+1}) in replay memory M. Within the replay memory, we sample a random subset of transitions (s_j, a_j, r_j, s_{j+1}), where j is an index in a mini-batch of transitions. The Q-values and target values are computed, and the temporal difference error or loss $\mathcal{L}(\phi)$ is calculated by comparing the predicted Q-values to the target values by using the mean squared error. The neural network parameters are updated through gradient descent to minimize the loss. Additionally, the weights of target network π^{T} are periodically refreshed with the weights of policy network π to improve stability. This iterative process continues over multiple episodes, enabling the policy network to learn and enhance its decision-making capabilities over time.

Algorithm 1 ASM Algorithm

Parameters and initialization:
ϕ - weights for the policy network Q^{ϕ}
ϕ^{T} - weights for target network Q^{T}
M - replay memory size
N^{T} - period of target updates
$\mathcal{B}$ - mini-batch size
for each episode **do**
  reset the environment
  for each $n \in \mathcal{N}$ **do**
    Choose the action based on ϵ-greedy approach
    Execute action in simulator
    Observe reward r_n
    Store the transition (s_n, a_n, r_n, s_{n+1}) in M
    Sample a random mini-batch of transitions $\{(s_j, a_j, r_j, s_{j+1})\ \forall j \in \mathcal{B}\}$ from the memory M:
    for each $j \in \mathcal{B}$ **do**
      Set $z_j = \begin{cases} r_j, & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a' \mid \phi^{T}), & \text{otherwise} \end{cases}$
      Calculate the loss (perform a gradient descent)
      $\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s')}[z_j - Q(s_j, a_j \mid \phi)]^2$
      Update the NN parameters:
      $\phi = \phi - \alpha \cdot \gamma \cdot L(\phi)$
      if n mod $N^{T} = 0$ **then**
        Update the target network:
        $\phi^{T} \leftarrow \phi$

VI. SYSTEM MODEL

This section describes the system model, including the end-to-end latency model and MEH failure model, that will be

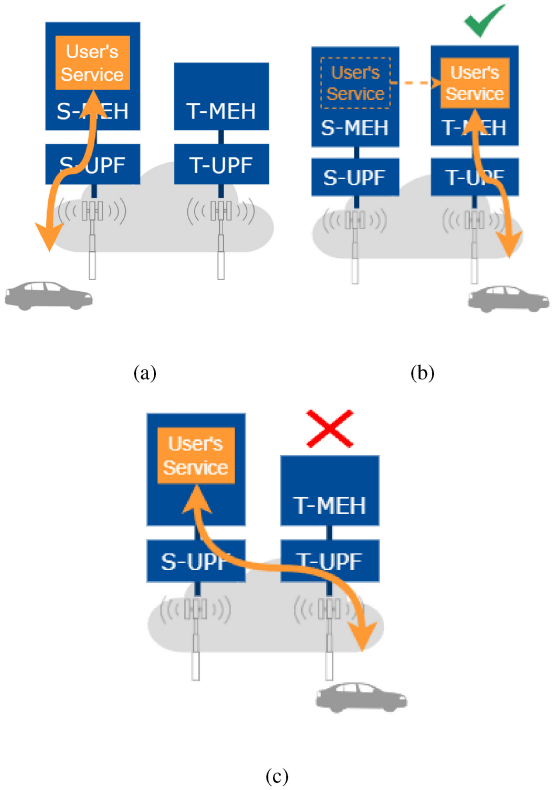


FIGURE 2. Scenario with mobility support enabled, where (a) is the initial condition, (b) is when the service is migrated and T-MEH is available, and (c) is when the service is not migrated due to the T-MEH cannot be used [8].

used in the evaluation. The system model- explained in this section is not used by the ASM. The system model creates an accurate simulation scenario that closely mirrors real-world conditions.

A. END-TO-END LATENCY

We assume that each MEH and local User Plane Function (UPF) of the 5G core is physically co-located with the gNBs [19]. The 5G UPF manages user plane operations such as packet routing and connectivity to MEHs and other data network. As illustrated in Fig. 2(a), a service initially runs on a S-MEH with traffic routed through the Serving-UPF (S-UPF) to gNB and finally the UE. In case of a handover, the gNB and, consequently, the UPF are changed. In such a case, the MEO should ideally transfer the service to the T-MEH that is co-located with the Target-UPF (T-UPF) (see Fig. 2(b)). However, the service transfer may not occur under two conditions: the T-MEH is unavailable, and the service instance on the S-MEH can still satisfy the delay requirement. When the service is not transferred, the service data is routed through the T-UPF, the S-UPF, and the S-MEH, as shown in Fig. 2(c).

The end-to-end latency on each timestep, $\zeta_d(n)$, comprises of network delay (which is composed of the *access network delay* between UE and the serving gNB $g \in \mathcal{G}$, $\eta_g^{\text{RAD}}(n)$, and *transport network delay* between the selected MEH d and the serving gNB g, $\eta_{g,d}^{\text{TNP}}(n)$), *migration delay* from MEH d' selected at the previous timestep to the selected MEH d (only id $d' \neq d$), $\eta_{d,d'}^{\text{MIG}}(n)$, and the *computing delay* at the selected MEH, $\eta_d^{\text{COMP}}(n)$.

$$\zeta_d(n) = \eta_g^{\text{RAD}}(n) + \eta_{g,d}^{\text{TNP}}(n) + \eta_{d,d'}^{\text{MIG}}(n) + \eta_d^{\text{COMP}}(n) \quad (22)$$

In an actual scenario, the end-to-end latency would measured by network monitoring tools used by the MEO. In our simulation, the various delays composing the end-to-end latency are computed using approximation equations from the literature. This simulation-based approach allows us to evaluate the impact of these factors under controlled conditions systematically. The access network delay $\eta_g^{\text{RAD}}(n)$ is primarily influenced by the radio transmission delay. This delay is calculated by dividing the *service payload* μ^{PL} by the radio transmission rate ω_g^{RAD} between UE and serving gNB g, as detailed in Eq. (23). The service payload μ^{PL} refers to the data volume exchanged in accessing a service (UoM: KB). The radio transmission rate ω^{RAD} is the data rate at which data can be transmitted over a radio communication channel or link (UoM: KB/s). The radio transmission rate ω^{RAD} depends on the number of Physical Resource Blocks (n_PRB), the bandwidth size of each PRB (PRB_bw), and the Signal-to-Noise Ratio (SNR), as in Eq. (24). The radio transmission rate calculation in the practical 5G channel is complex [40], [41]. We simplify the calculation and make it depend only on the variation of the number of PRB and SNR. A higher number of PRBs means more radio resources are allocated, and a higher SNR means better radio channel quality. Thus, a higher number of PRBs and a higher SNR mean a higher transmission rate.

The SNR can be calculated in (Eq. (25)). The signal is the sum of transmit power (Tx), transmitter and receiver antenna gain (Gt, Gr), and path loss between user and gNB g, $P_g(n)$ (UoM: dB). The noise is the sum of thermal noise Tm and additional losses from components of gNB, UE, and cables L^{ADD}. The thermal noise is uniformly distributed $Tm \sim U(0, 4)$. The pathloss $P_g(n)$ between UE and serving gNB g in Eq. (26) is derived from WINNER+ (C2 Urban LOS) [42]. The pathloss $P_g(n)$ depends on the distance between UE and serving gNB $\text{dist}_g(n)$ (UoM: m), and frequency carrier f (UoM: GHz).

$$\eta_g^{\text{RAD}}(n) = \frac{\mu^{PL}}{\omega_g^{\text{RAD}}(n)} \quad (23)$$

$$\omega_g^{\text{RAD}} = (\text{n\_PRB} \cdot \text{PRB\_bw}) \cdot \log_2\left(1 + \text{SNR}_g(n)\right) \quad (24)$$

$$SNR_g^{\text{dB}} = \left(Tx + Gt + Gr - P_g^{\text{dB}}(n)\right) - \left(Tm + L^{ADD}\right) \quad (25)$$

$$P_g^{\text{dB}}(n) = A \cdot log_{10}\left(\text{dist}_g(n)\right) + B + C \cdot log_{10}\left(\frac{f}{5.0}\right)$$
$$A = 26, B = 41, C = 20 \quad (26)$$

We assume that the MEC system has the intra-operator MEC application mobility support, where a user can move to another gNB that is not co-located with the S-MEH but still served by the S-MEH [8]. Therefore, an additional

transport network delay is calculated to accommodate the link connection between T-UPF and S-UPF. The transport network delay $\eta^{\text{TNP}}_{g,d}(n)$ can be computed as a transmission delay by using Eq. (27) and Eq. (28) (derived from [43]) that models the delay on a path between S-MEH and the serving gNB that actually includes the queuing and propagation delay on the path. The transport network delay depends on the service payload μ^{pl} and transport transmission rate $\omega^{\text{TNP}}_{g,d}(n)$, as in Eq. (27). If the MEH d is co-located with the serving gNB g, the transport network delay is assumed to be negligible. The model includes a maximum transport transmission rate ω^{MaxTNP}, which depicts a theoretical link transmission rate where there is no other traffic. The transmission rate between gNB g and MEH d, $\omega^{\text{TNP}}_{g,d}$, degrades when the distance between gNB g and MEH d increases. The model assumes a higher distance leads to a higher probability of network congestion and packet loss, consequently leading to a higher network delay [43].

$$\eta^{\text{TNP}}_{g,d}(n) = \frac{\mu^{PL}}{\omega^{\text{TNP}}_{g,d}(n)} \tag{27}$$

$$\omega^{\text{TNP}}_{g,d}(n) = \omega^{\text{MaxTNP}} \cdot \left(1 - \left(0.8 \cdot \frac{\text{dist}_{g,d}(n)}{\max(\text{dist}_{g,d}(n))}\right)\right) \tag{28}$$

The computing delay depends on the service requirement on the computing power π and the remaining computing capacity of the selected MEH $m^P_d(n)$, as shown in Eq. (29). We assume a maximum processing latency denoted as h (measured in ms) to simplify matters. We treat h as a static value that accounts for the varying computing power resulting from different CPU frequencies. In our scenario, we set h as 2 ms. Eq. (29) is derived from [10], [44], [45], [46], which describes the dependency of computing delay on CPU frequency and server resource usage.

$$\eta^{\text{COMP}}_{d}(n) = h \cdot \frac{\pi}{m^P_d(n)} \tag{29}$$

The migration delay occurs if the service is migrated from the S-MEH d' to a T-MEH d. The migration delay depends on the *service instance size* μ^{SI}, and the transport transmission rate between the two MEHs [10], [46]. The ω^{TNP} is decreased to the distance between S-MEH and T-MEH.

$$\eta^{\text{MIG}}_{d',d}(n) = \frac{\mu^{SI}}{\eta^{\text{TNP}}_{g,d}(n)} \tag{30}$$

B. MEH FAILURE MODEL

A usage-dependent failure model has a similar concept to the initial wear-out phase in the Weibull distribution [30]. We consider that the failure rate is increasing exponentially depending on the MEH usage:

$$\lambda(n) \ = \lambda_0 \cdot e^{b \cdot U(n)}, \tag{31}$$

where $\lambda(n)$ is the failure rate at the timestep n, $U(n)$ is the recorded usage at timestep n, λ_0 is the baseline failure rate, and b is the coefficient for the exponential increase in failure rate with usage.

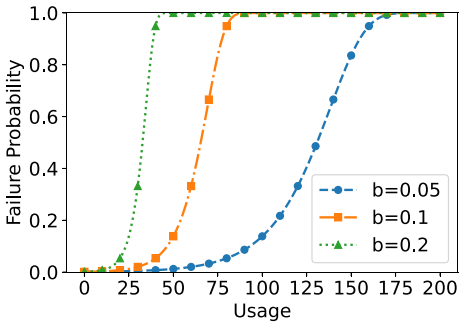


FIGURE 3. Usage-dependant failure model with different *b*.

The probability of a server failure then can be calculated as follows:

$$P(\text{MEH failure}) = e^{-\lambda(n)} \tag{32}$$

When a MEH is often to be selected, that MEH is more likely to fail sooner. The MEH availability of different b is shown in Fig. 3. The increase of b means that the impact of usage is more severe, e.g., the MEH becomes more prone to failure. When a MEH fails at timestep n, we assume a static recovery time of five timesteps, meaning that the particular MEH will fully recover at timestep $n + 5$.

VII. EVALUATION

In this section, we describe the simulation scenario to evaluate our solution. Then, we discuss the result, implications, and limitations of our solution.

A. EVALUATION SCENARIO

The simulation considered a dense urban area that is modelled as a grid 600 x 600m area, with a number of MEHs $|\mathcal{D}|$ equal to 9 and an inter-site distance of 100 m horizontally, 141 m diagonally (see Fig. 4). The mobility pattern mimics the road in a city, thus following a specific grid road path. The MEHs are co-located with the gNBs [19]. For the sake of simplicity, we assume a single-user, and all the MEHs have the same remaining computing and memory capacities, which do not change in time, i.e., they are the same for all the timesteps. However, MEO must incorporate all its users in a practical, actual condition. While our simulation accurately models the availability dynamics and responsiveness from the perspective of a single user, real-world scenarios often involve multiple users concurrently accessing and interacting with services.

Table 3 shows the parameters associated with the method, where the values are set based on references and empirical studies. The values for the MEC system are taken from the capacities of Intel NUC models, which are a solution for edge servers [47]. We consider a single user that needs one service in all timesteps and moves on the road. The service has a similar characteristic of an advanced driving application

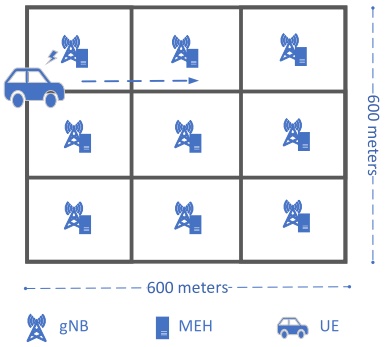


FIGURE 4. MEH-gNB location and UE trajectory.

TABLE 3. Simulation setting.

Parameter	Value
Service requirement	
δ, π, θ	10 ms, 4 vCPU, 4 GB
MEC system	
$m_d^P(n)$, $m_d^Q(n)$ $\forall$ $d \in \mathcal{D}$, $\forall$ $n \in \mathcal{N}$	16 vCPU, 64 GB
RL and NN hyper-parameters	
α, γ	0.001, 0.9
N^T, M, ϵ	20, 10000, [0.9,0.05]
Hidden layer, Batch size	3 layers (size = 128), 512
NN Optimizer	AdamW (PyTorch)

that requires a strict low latency and a medium amount of computing capability [48], [49]. The required latency is 10 ms [48], and the computing and memory requirements are 4 vCPUs and 4 GB, respectively [49]. The service payload μ^{PL} for transmitting and receiving the service is 32 KB [48]. The service instance size μ^{SI} is 40 MB, based on the size of the VideoLAN Client (VLC) application [50] studied in service migration experiment as a representation of a V2X application [51].

We assume that the gNBs operate on the 5.9 GHz spectrum, as the 5G Automotive Association (5GAA) expects it would be the standard for C-V2X direct radios in global advanced-driving scenario [52]. There is one agent which learns to maximize the service availability of a user and has the learning rate $\alpha = 0.001$ and the discount factor $\gamma = 0.9$. We use a replay experience with a memory size of 10000. We consider a NN with three layers and 128 nodes per layer. We use an Adaptive Moment Estimation (Adam) algorithm to optimize the NN weights and the Rectified Linear Unit (ReLU) for the activation function of a particular input. The evaluation was conducted using a laptop with 8 vCPUs, a 2.8 GHz processor, and 32 GB RAM. We use Python 3.9, TensorFlow v2.12.0, and Spyder 5.2.2. We consider 350 timesteps, 500 episodes, and 5 simulations per scenario.

We perform a simulative evaluation of ASM by comparing it with the Distance Focused Service Migration (DFSM) algorithm, which was previously proposed in [11]. DFSM is a state-of-the-art solution to the service migration problem with the presence of failures. DFSM aims to minimize the number of migration failures by measuring the user experience, which degrades to the distance between a user and a MEH. DFSM stands out due to its ability to adapt its learning rate depending on the presence of node or link failures. However, there is a lack of information provided in the paper about the deep learning optimizer implementation and how the learning rate is affected. To establish a fair comparison, we adopt an Adam optimizer algorithm for both ASM and DFSM that adaptively adjusts the learning rate based on changes in NN weight gradients, whether in the presence of failures or not [53]. Among all the investigated DRL settings for DFSM, we have chosen the one that leads to the highest availability.

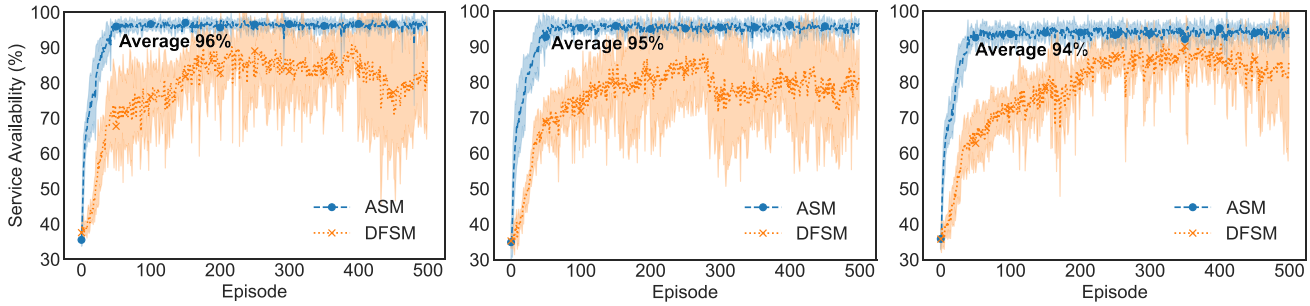
Compared with DFSM, ASM enhances problem representation by defining states and rewards that incorporate service availability into the agent's learning process. This enables the ASM agent to improve service availability directly, unlike the DFSM agent, which rewards based on the distance between the UE and the S-MEH. The DFSM's attributes are primarily based on the number of hops, disregarding other factors that may influence latency. Overall, our approach presents an innovative problem representation that aids in maximizing service availability, especially in the presence of MEH failures, compared to the DFSM approach.

B. RESULTS AND DISCUSSION

Fig. 5 shows the average service availability (expressed as the percentage of time that the service is up with respect to the total simulation time) at each episode. Each subfigure has a different b value. Given the same usage, a higher b implies a higher failure probability. When $b = 0.05$, there are only two to four MEH failures over an episode. When $b = 0.1$, six to eight MEH failures occur; when $b = 0.2$, ten to twelve failures occur. The figure shows that the ASM agent learns how to act to improve service availability over the episodes. ASM and DFSM have the same learning rate α, discount factor γ, same NN parameters (hidden layer, size) and replay memory size M. The findings of this study can be described as follows.

1) FAST CONVERGENCE

ASM learns faster than DFSM. In all scenarios, the ASM agent converges after 50 episodes, while the DFSM agent converges after 200 to 300 episodes. The ASM agent considers the service availability in the reward function and state features, allowing the agent to take an action that maximizes its return and improves the average service availability over the episodes. Fig. 6 depicts the average return per episode, showing that the agent takes the right action to improve its return and converges. Instead, the DFSM agent takes actions to minimize the end-to-end latency by minimizing the distance. With the presence of



(a) b = 0.05 (b) b = 0.1 (c) b = 0.2

FIGURE 5. Service availability by using ASM and DFSM with different MEH failure probabilities.

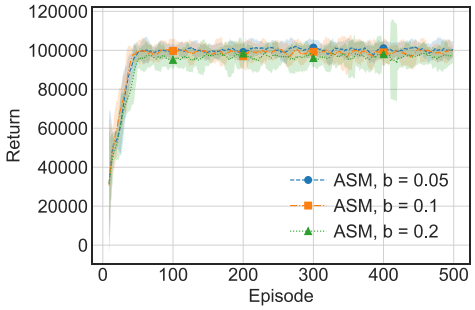


FIGURE 6. Average reward of ASM with different *b*.

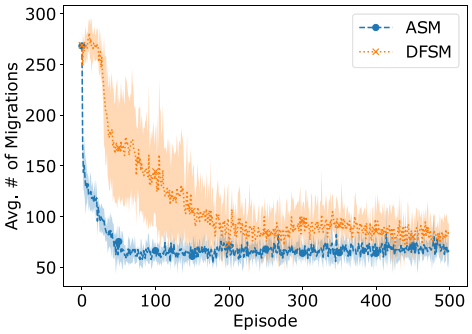


FIGURE 7. Number of migrations in each episode.

MEH failure, it is more difficult for DFSM to converge in the early stage.

2) ALWAYS MORE THAN 94% SERVICE AVAILABILITY

ASM provides stable average service availability. Fig. 5(a) shows the average service availability when the MEH rarely fails. ASM gives 96% average service availability when the MEH fails more often (Fig. 5(b) and Fig. 5(c)), the average service availability slightly decreases to 95% and 94% because more MEHs are unavailable to serve the user. Observing the average service availability of DFSM is difficult as convergence cannot be guaranteed in the early phase. After 100 episodes, DFSM provides around 75% to 90% service availability. However, ASM is slightly more computationally intensive, having a higher execution time than DFSM. The ASM needs approximately 27 ms to take decisions, while the DFSM takes approximately 22 ms.

3) LESS MIGRATION LEADS TO BETTER SERVICE AVAILABILITY

Fig. 7 shows the average migration frequency of all time steps on each episode. The ASM agent migrates the user less often than the DFSM agent. In the learning phase, the ASM agent migrates less frequently and achieves the convergence where the service is migrated 66.3 times on average over an episode. On the other hand, the DFSM agent migrates the service around 83.7 times. Thus, having a smaller number of migrations can lead to a stable and higher service availability.

TABLE 4. Reasons of service unavailability.

Avg. service failure	b = 0.05	b = 0.1	b = 0.2
ASM - Latency violation	90.7 %	76.3 %	61.3 %
ASM - MEH unavailability	9.3 %	23.8 %	38.7 %
DFSM - Latency violation	97.7 %	92.6 %	75.7 %
DFSM - MEH unavailability	2.3 %	7.4 %	24.3 %

4) LINEAR CORRELATION BETWEEN MEH AVAILABILITY AND SERVICE AVAILABILITY

We divide the causes of service failure into two categories: latency violation and unavailability of the selected MEH. Table 4 provides an overview of the causes of service unavailability, indicating that most service unavailability is due to latency violations (90.7% of service unavailability). However, as the MEH failures increase, the number of service unavailabilities caused by MEH unavailability also increases (specifically, from 9.3% to 38.7%). In contrast, DFSM shows almost similar behaviour, where most of the service failures are due to latency violations and a small percentage are due to MEH unavailability.

VIII. CONCLUSION

This paper formulates a service migration problem that aims to maximize service availability considering user mobility, resource constraints, service requirements, and MEH availability. Two decisions must be made: the migration timing and the T-MEH selection. This paper proposes ASM, a

DQN-based algorithm, to solve the formulated problem. The evaluation shows that the ASM maintains a higher and more stable service availability (94% to 96%) under the presence of different MEH failure rates, than DFSM, a reference solution. ASM maintains a high service availability by having fewer migrations than DFSM. Further refinements can be made to improve the ASM algorithm. In future, we will address scenarios where several users request different services that run in more complex and diverse environments. We can design a cooperative multi-agent RL that learns to coordinate the actions to maximize the overall service availability.

REFERENCES

[1] (Intel-Corp., Santa Clara, CA, USA). *Accelerate the Development, Deployment, and Time to Value of Video Analytics at the Edge*. 2023. [Online]. Available: https://www.intel.com/content/www/us/en/internet-of-things/computer-vision/intelligent-video/overview.html

[2] V. Yeruva. "Edge computing's applications in autonomous driving and business at large." 2023. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/09/27/edge-computings-applications-in-autonomous-driving-and-business-at-large/?sh=4eed401e96aa

[3] M. K. Pratt. "The cutting edge of healthcare: How edge computing will transform medicine." 2021. [Online]. Available: https://www.computerworld.com/article/3635589/cutting-edge-healthcare-how-edge-computing-will-transform-medicine.html

[4] "Multi-access edge computing (MEC); framework and reference architecture," ETSI, Sophia Antipolis, France, document GS MEC 003, Mar. 2022.

[5] A. Sarah, G. Nencioni, and M. M. I. Khan, "Resource allocation in multi-access edge computing for 5g-and-beyond networks," *Comput. Netw.*, vol. 227, May 2023, Art. no. 109720. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623001652

[6] "MEC federation: Deployment considerations," ETSI, Sophia Antipolis, France, White Paper 49, Jun. 2022.

[7] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[8] "Multi-access edge computing (MEC): MEC 5G integration," ETSI, Sophia Antipolis, France, document GR MEC 031, Dec. 2020.

[9] J. Li et al., "Service migration in fog computing enabled cellular networks to support real-time vehicular communications," *IEEE Access*, vol. 7, pp. 13704–13714, 2019.

[10] J. Wang, J. Hu, G. Min, Q. Ni, and T. El-Ghazawi, "Online service migration in MEC with incomplete system information: A deep recurrent actor-critic learning approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6663–6675, Nov. 2023.

[11] L. Rui, M. Zhang, Z. Gao, X. Qiu, Z. Wang, and A. Xiong, "Service migration in multi-access edge computing: A joint state adaptation and reinforcement learning mechanism," *J. Netw. Comput. Appl.*, vol. 183, Jun. 2021, Art. no. 103058.

[12] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[13] M. I. Ashraf, C.-F. Liu, M. Bennis, W. Saad, and C. S. Hong, "Dynamic resource allocation for optimized latency and reliability in vehicular networks," *IEEE Access*, vol. 6, pp. 63843–63858, 2018.

[14] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in MEC networks with storage, computation, and communication constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1047–1060, Jun. 2020.

[15] M. Siew, S. Sharma, and C. Joe-Wong, "ACRE: Actor critic RL for failure-aware edge computing migrations," in *Proc. CISS*, 2023, pp. 1–6.

[16] "MEC security: Status of standards support and future evolutions," ETSI, Sophia Antipolis, France, White paper 46, Sep. 2022.

[17] G. Nencioni, R. G. Garroppo, and R. F. Olimid, "5G multi-access edge computing: A survey on security, dependability, and performance," *IEEE Access*, vol. 11, pp. 63496–63533, 2023.

[18] "Mec management; part 2: Application lifecycle, rules and requirements management," ETSI, Sophia Antipolis, France, document GS MEC 010-2, Jun. 2023.

[19] S. Kekki, W. Featherstone, and Y. Fang, "MEC in 5G networks (first ed)," ETSI, Sophia Antipolis, France, White Paper, Jun. 2018.

[20] P. V. Wadatkar, R. G. Garroppo, and G. Nencioni, "MEC application migration by using advantEDGE," in *Proc. Int. Conf. Testbeds Res. Infrastruct.*, 2022, pp. 104–118.

[21] M. Merluzzi, P. Di Lorenzo, S. Barbarossa, and V. Frascolla, "Dynamic computation offloading in multi-access edge computing via ultra-reliable and low-latency communications," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 342–356, Mar. 2020, doi: 10.1109/TSIPN.2020.2981266.

[22] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven VNF placement in a MEC-NFV environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–7.

[23] Z. Zhao, H. Cheng, and X. Xu, "Improved DQN-based computation offloading algorithm in MEC environment," in *Proc. IEEE 28th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2023, pp. 25–32.

[24] Y. Wang et al., "Towards cost-effective service migration in mobile edge: A q-learning approach," *J. Parallel Distrib. Comput.*, vol. 146, pp. 175–188, Dec. 2020.

[25] Z. Gao, Q. Jiao, K. Xiao, Q. Wang, Z. Mo, and Y. Yang, "Deep reinforcement learning based service migration strategy for edge computing," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, 2019, pp. 116–1165.

[26] M. L. F. Sindjoung, M. Velempini, and A. B. Bomgni, "A MEC architecture for a better quality of service in an autonomous vehicular network," *Comput. Netw.*, vol. 219, Dec. 2022, Art. no. 109454.

[27] H. T. Malazi et al., "Dynamic service placement in multi-access edge computing: A systematic literature review," *IEEE Access*, vol. 10, pp. 32639–32688, 2022.

[28] F. P. Mathur, V. Profile, and O. M. A. Metrics, *Hardware Reliability: Encyclopedia of Computer Science*. New York, NY, USA: ACM, Jan. 2003.

[29] A. Fox, D. A. Patterson, and R. H. Chen, "FaaS: A cloud service for large-scale, online failure drills," Dept. EECS, Univ. California, Berkeley, CA, USA, Rep. EECS-2011-87, 2011.

[30] S. Nadarajah, "Bathtub-shaped failure rate functions," *Qual. Quant.*, vol. 43, no. 6, pp. 855–863, 2009.

[31] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "SEGUE: Quality of service aware edge cloud service migration," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, 2016, pp. 344–351.

[32] P. Wang, T. Ouyang, G. Liao, J. Gong, S. Yu, and X. Chen, "Edge intelligence in motion: Mobility-aware dynamic DNN inference service migration with downtime in mobile edge computing," *J. Syst. Archit.*, vol. 130, Sep. 2022, Art. no. 102664.

[33] Y. Chen, Y. Sun, C. Wang, and T. Taleb, "Dynamic task allocation and service migration in edge-cloud IoT system based on deep reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 16742–16757, Sep. 2022.

[34] E. Kartsakli et al., "Ai-powered edge computing evolution for beyond 5G communication networks," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, 2023, pp. 478–483.

[35] J. Huang et al., "Reinforcement learning based resource management for 6G-enabled mIoT with hypergraph interference model," *IEEE Trans. Commun.*, vol. 72, no. 7, pp. 4179–4192, Jul. 2024.

[36] D. Silver. "Deep reinforcement learning." Jun. 2016. [Online]. Available: https://deepmind.google/discover/blog/deep-reinforcement-learning/

[37] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in MEC: A RL approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.

[38] "Mobile edge computing (MEC); end to end mobility aspects," ETSI, ophia Antipolis, France, document GS MEC 018, Oct. 2017.

[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT press, 2018.

[40] A. K. Thyagarajan, P. Balasubramanian, D. Vydeki, and M. Karthik, "SNR-CQI mapping for 5G downlink network," in *Proc. IEEE Asia Pac. Conf. Wireless Mobile (APWiMob)*, 2021, pp. 173–177.

[41] M. A. T. Almahadeen and A. M. Matarneh, "Performance assesment of throughput in a 5G system," *Jordan. J. Comput. Inf. Technol.*, vol. 6, no. 3, pp. 1–14, 2020.

[42] P. Kyösti et al., *IST-4-027756 WINNER II D1.1.2 V1.2 WINNER II Channel Models*, CEPT Univ., Ahmedabad, India, document D1.1.2 V1.2, 2007.
[43] T. Lu, S. Chang, and W. Li, "Fog computing enabling geographic routing for urban area vehicular network," *Peer Netw. Appl.*, vol. 11, pp. 749–755, Jul. 2018.
[44] G. Li, J. Wang, J. Wu, and J. Song, "Data processing delay optimization in mobile edge computing," *Wireless Commun. Mobile Comput.*, vol. 2018, no. 1, Feb. 2018, Art. no. 6897523. [Online]. Available: https://www.hindawi.com/journals/wcmc/2018/6897523/
[45] B. Li, P. Hou, H. Wu, R. Qian, and H. Ding, "Placement of edge server based on task overhead in mobile edge computing environment," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 9, 2021, Art. no. e4196. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4196
[46] A. Abouaomar, Z. Mlika, A. Filali, S. Cherkaoui, and A. Kobbane, "A DRL approach for service migration in MEC-enabled vehicular networks," in *Proc. IEEE LCN*, 2021, pp. 273–280.
[47] (Intel, Santa Clara, CA, USA). *Embedded Edge Server*. Accessed: Dec. 1, 2023. [Online]. Available: https://simplynuc.com/embedded-edge-server/
[48] M. H. C. Garcia et al., "A tutorial on 5G NR V2X communications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1972–2026, 3rd Quart., 2021.
[49] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled V2X service placement for intelligent transportation systems," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1380–1392, Apr. 2021.
[50] (VideoLAN, Paris, France). *VLC: Official Site—Free Multimedia Solutions for all OS!*. Accessed: Jul. 15, 2024. [Online]. Available: https://www.videolan.org/
[51] R. G. Garroppo, M. Volpi, G. Nencioni, and P. V. Wadatkar, "Experimental evaluation of handover strategies in 5G-MEC scenario by using advantedge," in *Proc. IEEE Int. Mediterr. Conf. Commun. Netw.*, 2022, pp. 286–291.
[52] "A visionary roadmap for advanced driving use cases, connectivity technologies, and radio spectrum needs," 5GAA Autom. Assoc., Munich, Germany, White paper, 2022. Accessed: May 12, 2023.
[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optim," 2017, *arXiv:1412.6980*.

ANNISA SARAH received the B.Sc. degree in telecommunication engineering from Telkom University, Indonesia, in 2014, the M.Sc. degree in wireless systems from the KTH Royal Institute of Technology, Sweden, in 2017, and the Ph.D. degree from the University of Stavanger, Norway, in 2021. Previously, she worked as an Assistant Professor for electrical engineering program, responsible to teach and research in telecommunications, with the Atma Jaya Catholic University of Indonesia from 2018 to 2021. Her current research activity regards resource allocation in 5G-multi-access edge computing ecosystem. Her past works were mainly for wireless network, network optimization, and rural network.

GIANFRANCO NENCIONI received the M.Sc. degree in telecommunication engineering and the Ph.D. degree in information engineering from the University of Pisa, Italy, in 2008 and 2012, respectively. He is an Associate Professor with the University of Stavanger, Norway, from 2018. In 2011, he was a Visiting Ph.D. Student with the Computer Laboratory, University of Cambridge, U.K. He was a Postdoctoral Fellow with the University of Pisa from 2012 to 2015 and the Norwegian University of Science and Technology, Norway, from 2015 to 2018. He is currently the Head of the Computer Networks Research Group and the Leader of the 5G-MODaNeI Project funded by Norwegian Research Council. His research activity regards modeling and optimization in emerging networking technologies (e.g., SDN, NFV, 5G, network slicing, and multi-access edge computing). His past research activity has been focused on energy-aware routing and design in both wired and wireless networks and on dependability of SDN and NFV.

MD MUHIDUL ISLAM KHAN received the master's degree from the Bangladesh University of Engineering and Technology in 2009, the Ph.D. degree in interactive and cognitive environment under the Erasmus Mundus Grant from European Commission working with Klagenfurt University, Austria, and the University of Genova, Italy, from January 2011 to September 2014, and the Joint Doctorate degree in September 2014. He is a Postdoctoral Fellow with the University of Stavanger, Norway, from 2021. He has participated in the "eLINK"-project at Corvinus University of Budapest, Hungary, from September 2009 to July 2010 (funded by the European Union). He joined as an Assistant Professor with BRAC University, Bangladesh, and served there for one year. After that, he completed his one-year Postdoctoral from the Hebei University of Technology, Tianjin, China. He worked as a Research Scientist with the Electronics Department, Tallinn University of Technology, Estonia. He worked as a Senior Lecturer with the School of Information Technologies, Tallinn University of Technology, Tallinn, Estonia. His specialization lies in the fields of wireless sensor networks, networked embedded systems, and pervasive computing.