Federated Deep Reinforcement Learning for Prediction-Based Network Slice Mobility in 6G Mobile Networks

Zhao Ming, *Student Member, IEEE,* Hao Yu, *Member, IEEE,* and Tarik Taleb, *Senior Member, IEEE*

Abstract—**Network slices are generally coupled with services and face service continuity/unavailability concerns due to the high mobility and dynamic requests from users. Network slice mobility (NSM), which considers user mobility, service migration, and resource allocation from a holistic view, is witnessed as a key technology in enabling network slices to respond quickly to service degradation. Existing studies on NSM either ignored the trigger detection before NSM decision-making or didn't consider the prediction of future system information to improve the NSM performance, and the training of deep reinforcement learning (DRL) agents also faces challenges with incomplete observations. To cope with these challenges, we consider that network slices migrate periodically and utilize the prediction of system information to assist NSM decision-making. The periodical NSM problem is further transformed into a Markov decision process, and we creatively propose a prediction-based federated DRL framework to solve it. Particularly, the learning processes of the prediction model and DRL agents are performed in a federated learning paradigm. Based on extensive experiments, simulation results demonstrate that the proposed scheme outperforms the considered baseline schemes in improving long-term profit, reducing communication overhead, and saving transmission time.**

Index Terms—**Prediction-based Network Slice Mobility, Incomplete Observation, Deep Reinforcement Learning**

I. Introduction

During the past few years, network technology has developed rapidly and brought humans into the 6G era with much stricter and more heterogeneous network requirements, e.g., increased data rates, enhanced network capacity, ultra-low latency, and massively connected devices [1]–[3]. To cope with these challenges, network slicing, which is empowered by the emerging software-defined networking (SDN) and network function virtualization (NFV) technologies, aims at building multiple virtual logically independent networks by constructing flexible and highly adaptable communication networks. Network slicing has achieved great progress in supporting specific demands of customers [4], [5], industries [6], and emerging applications [7]–[9] and is witnessed as a key technology for supporting specific demands in 6G networks.

On the other hand, built on top of physical facilities to provide customized services to user equipment (UE), network slices not only depend on physical facilities to adjust the virtual network topology and to decide on the resource allocation strategies, but also are deeply coupled with services and UEs for ensuring service continuity [10], [11]. Under this circumstance, when the slices' serving UEs move across areas, the ongoing sessions between each pair of the moving UEs and slices may suffer a degradation in quality of service (QoS) and thus induce service continuity concern [12], [13]. Additionally, the dynamically changing requests from associated UEs also require the slices to quickly adjust their provisioning resources, which can lead to resource unavailable issues. These problems will even aggregate in 6G networks due to the increasing number of connected equipment and emerging scenarios with high mobility or dynamic resource requests. To tackle these issues, the authors in [14] considered user mobility, request dynamics, and service continuity simultaneously and proposed the notion **network slice mobility (NSM)** to decide the slice migration and resource allocation from a holistic perspective. Specifically, in the considered NSM paradigm, network slices are assembled from multiple virtual network functions (VNFs), which can monitor the requests/positions of users in a timely manner and collect system information, including available physical facilities resources. Additionally, the slices can manipulate their VNFs to migrate between physical hosts, scale the allocated resources, and even change the connection relationships among VNFs. The main NSM triggers were defined and summarized in [15], where the trigger selection methods were also investigated.

Lots of pioneering studies have investigated NSM-related issues including slice trigger classification [14], slice anomaly/trigger detection [15], [16], and VNF/service placement and migration [17], [18], but there still exists several challenges that have not been resolved. Firstly, these studies either focused on the detection of slice anomalies/triggers or aimed at slice deployment and migration independently. However, to ensure service continuity, we maintain that slice migration should be performed only after a slice anomaly

Part of this work was presented at IEEE International Conference on Communications (ICC), Rome, Italy, May 2023, which is cited as [1]. This work has made a significant extension on system modeling, scheme design, and evaluation results. (Corresponding author: Hao Yu).

This research work is supported in part by the European Union's HE research and innovation program HORIZON-JUSNS-2023 under the 6GPath project (Grant No. 101139172), the European Union's HE research and innovation program HORIZON-JUSNS-2022 under the 6GSandbox project (Grant No. 101096328); and by the AerOS project funded by the European Union's Horizon Europe, the EU's key funding program for research and innovation under Grant No. 101069732. The paper reflects only the authors' views, and the European Commission bears no responsibility for any utilization of the information contained herein.

Zhao Ming is with the Centre for Wireless Communications (CWC), University of Oulu, Oulu, 90570 Finland (email: zhao.ming@oulu.fi).

Hao Yu is with ICTFicial Oy, 02130, Espoo, Finland (email: hao.yu@ictficial.com).

Tarik Taleb is with the Ruhr University Bochum, Bochum, Germany (email: tarik.taleb@rub.de).

occurs. As the anomalies in slices occur intermittently, NSM should also be performed intermittently rather than frequently, which is rarely considered. Secondly, though DRL-based schemes are widely adopted for solving NSM decision-making problems and have demonstrated their efficiencies, when we consider NSM intermittently, the DRL agents can only observe the system information (including user requests and resource utilization of servers) for one time slot to make decisions, after the NSM, the requests and positions of UEs will continuously change until the next anomaly triggers. Under this circumstance, the DRL agents face the challenge that they only have incomplete observations of the environment, as they cannot observe future requests that affect the feedback. Thus, they are unable to receive stable rewards. Lots of studies also discussed the effect of incomplete observation on unstable reward in DRL training [19]–[21]. Finally, the dynamically changing user requests and continuously growing BSs in 6G networks also put great challenges to the flexibility and scalability of the core network and radio access network (RAN), as well as the cost for the MNOs to have dedicated RAN facilities. Moreover, centralized algorithms for detecting the NSM anomalies or deciding the slice migration strategies have high computation complexity that increases with the number of slices, which will introduce high communication overhead and execution time.

To cope with these challenges, in this paper, we design a general network architecture for NSM in 6G networks, which considers the decentralized core network to be realized by the VNFs deployed at edge servers (ESs) and the RAN built by the VNFs under the Open-RAN paradigm. Afterward, to mimic the intermittent NSM process, we model it periodically without loss of generality, assuming that the anomaly occurs at the first time slot of each period. To cope with the incomplete observation challenge, we creatively integrate the prediction of user behaviors to improve the NSM decision-making performance inspired by [22], [23]. The problem is formulated to maximize the long-term system profit of MNOs, and we propose a prediction-based federated DRL (FDRL) framework that learns users' requests and position information to solve this problem. Specifically, the framework consists of a long short-term memory (LSTM) prediction module for prediction and a double deep Q-learning (DDQN)-based decision-making module for determining the NSM strategies. Particularly, during each period, the DRL agents can only observe the first time slot's system information and make decisions with the current observation and the predictions, after which the overall system profit of the period is set as the feedback. The training of the LSTM model and DRL agents is performed in a federated learning (FL) manner to reduce the communication overhead and transmission time of original user data and preserve users' privacy. Based on extensive experiments, simulation results demonstrate that our proposed scheme outperforms the considered baseline schemes in improving the long-term system profit and reducing communication overhead/time. The main contributions of this paper are summarized as follows:

- We design a general NSM network architecture that considers a decentralized core network to be realized by the VNFs deployed at ESs and the RAN built under the Open-RAN paradigm to improve the system flexibilities and scalabilities.
- To cope with the dynamic user requests and unstable feedback caused by incomplete observation of agents, we consider NSM from a periodical perspective and utilize future information prediction for DRL agent training to improve the NSM decision-making performance.
- We formulate the problem as maximizing the long-term system profit and propose a prediction-based FDRL framework to solve this problem. Specifically, the framework utilizes LSTM for future information prediction and DDQN for decision-making, and the learning processes of the LSTM model and DRL agents are integrated with the FL paradigm.
- Simulation results demonstrate that the proposed prediction-based NSM scheme outperforms the considered baseline schemes in improving the long-term system profit, reducing the communication overhead, and saving transmission time.

The remainder of this paper is organized as follows. Section II discusses the related work. Section III introduces the system model and then formulates the problem. In Section IV, we propose a prediction-based NSM framework. Simulation results are provided in Section V. Finally, Section VI concludes this paper.

II. Related Work

Lots of researchers have investigated NSM-related techniques, including slice anomaly/trigger detection and decision-making for migration and resource allocation.

A. Anomaly and Trigger Detection

The authors in [24] addressed distributed online anomaly detection of network slices based on the decentralized one-class support vector machine by analyzing real-time measurements of virtual nodes mapped to physical nodes and correlation of measurements between neighbor virtual nodes. However, this study only focused on one type of anomaly in network slices, i.e., the anomalies of physical nodes. It didn't consider the degradation of service introduced by other reasons like unreasonable resource allocation. In this regard, if a physical node could support most of its slices normally but allocated limited bandwidth resources to a VNF, the unsatisfied service level agreement (SLA) of the corresponding slice will not be able to be detected. Moreover, the anomalies in the transmission of the RAN, transport network, and core network, the mobility of UEs, and the dynamically changing user demand were neither considered in this study. The authors in [25] proposed a cognitive network and slice management system, which adopted anomaly detection/prediction to detect arising anomalies in the routes taken by ambulances or their demanded QoS level. The authors considered AI-based techniques to detect and predict anomalies and the cooperation among AI models. Besides, Wang *et al.* proposed a transfer learning-based hidden Markov model to detect abnormal network slices affected by anomalies in shared physical nodes, which utilized the similarity between physical nodes to speed up the convergence and achieved high detection accuracy [16]. The authors in [15] investigated slice mobility trigger selection by a DRL-based method, in which the DRL agent decides where to migrate at each time slot and gives the trigger when the chosen target host is not the current host. However, these studies neglected the dynamic changes in user resource demands, and frequent migration of slices would result in the degradation of service continuity. Therefore, applying these studies in real-world networks would be difficult.

B. Slice Migration and Resource Allocation

On the other hand, in [18], the authors proposed a prediction-assisted VNF placement and link allocation framework to minimize energy consumption and total cost under the premise of meeting the network QoS. In [26], the authors proposed an algorithm based on the belief-state partially observable Markov decision process (MDP) to provide partitions/slices with energy and computational resources. These methods considered dynamic user requests and can avoid wasting resources [27]. Besides, to improve resource utilization efficiencies, decision-making strategies have been widely studied, and some researchers also proposed game theory-based solutions to cope with the conflicts of resource allocation of different servers [26], [28]. Moreover, in [18], the authors proposed an online approach to dynamically determine the slice placement policies and decide on the VNF numbers and resource allocation strategies. The prediction of network demand was considered to estimate the traffic rate demand in advance. Additionally, the authors in [29] proposed to minimize the total latency of the computing tasks with energy constraints by leveraging the combination of non-orthogonal multiple access technique and edge computing. However, the resource allocation policies were determined by centralized heuristic or near-optimal solutions, which may face scalability and robustness issues [30]. Moreover, distributed learning paradigms and their cooperation with networks were not considered in these studies.

Meanwhile, Chergui *et al.* proposed a statistical FL-based framework that performed slice-level resource prediction to minimize energy consumption, where the non-convex FL problem was solved by a proxy-Lagrangian strategy [31]. Furthermore, the authors in [17] introduced a knowledge plane-based management and orchestration framework that invoked a continuous model-free DRL method to minimize energy consumption and VNF instantiation cost. However, resource allocation and optimization of cost and energy are multi-goal optimization problems, and interactive decision-making in multiple network slices will conflict with each other. To address these issues, the authors in [26], [28] proposed game theory solutions based on distributed SDN to decide the resource allocation, where the resource allocation strategies can be derived more closely at the gateway level. In [1], we proposed a prediction-based intelligent slice mobility framework, which considered future information like users' positions and requests to assist the decision-making in advance. This method introduces higher system costs but also achieves much higher system revenue. However, this work didn't consider the slice trigger either and tended to make decisions at each time slot. Moreover, the resource allocation strategies can also be optimized with future information prediction.

In recent years, DRL-based methods have played a vital role in the decision-making field, but utilizing conventional DRL-based schemes for NSM still faces the problem of imperfect observation, and several studies have also considered this problem. For instance, the authors in [19] discussed the effect of noise on the observation of agents and tried to extract real and complete observations from the original ones. In [20] and [21], the authors studied the problem of unstable feedback of agents when dealing with highly dynamic environments. In addition, the combination of future information prediction and DRL has also been extensively studied [32]. Specifically, the authors investigated state prediction for agents' pre-training and allocating resources in advance [33], [34]. Besides, authors in [35] and [36] studied reward prediction and proposed using prediction to control agents' future actions. Recently, the authors in [22] and [23] proposed to cope with the partially observable MDP problem by embedding the predictions to the state space, which achieved good performances for agents in real-time interactions with the environment. Inspired by this concept, in our periodical NSM process, the unobservable future system information can be predicted to embed with the state space to achieve more stable feedback for agents.

Overall, the slice anomaly detection and resource allocation strategies are still worth investigating, the NSM problem is still unexplored well, especially in the periodical NSM process modeling, prediction-based decision-making, and the integration of distributed learning paradigms.

III. System Model and Problem Formulation

In this section, we introduce the system model and formulate the optimization problem. The main notations used in this paper are summarized in Table I.

A. System Model

We consider a general network architecture for prediction-based NSM in 6G networks, which consist of the ground network, the space and air network, and the remote cloud, as shown in Fig. 1. In the ground network, several small base stations (BSs) and macro BSs are geographically evenly distributed, each of which includes an ES with limited calculation/storage resources and runs a hypervisor[1]. Moreover, each BS covers a cell area and serves multiple UEs including smartphones, laptops, IoT/IoV equipment in the ground network, and satellites/UAVs in the space and air network. These UEs can dynamically move across different cells and connect to different BSs by cellular links according to their positions. To support heterogeneous resource requests from users, multiple network slices are built and empowered by the NFV/SDN techniques. We assume that each ES can instantiate several VNFs based on the deployed hypervisor. These VNFs run a diverse range of services for users, and each VNF incorporates an agent to adjust the allocated resources, physical host, and

[1] In this paper, we use BS and ES interchangeably.

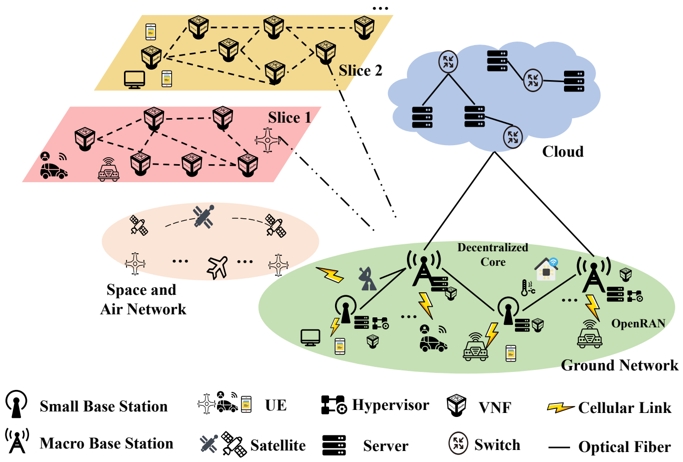


Fig. 1. The considered network architecture in space-air-ground 6G systems.

TABLE I
KEY MODELING PARAMETERS AND NOTATIONS.

Notation	Definition
u, $\mathcal{U}$	Indexes and set of Users
n, $\mathcal{N}$	Indexes and set of ESs
$\mathcal{R}_n = \bigcup\limits_i \{R_{n,i}\}$	Total resources of ES n
t	Time slot index
i	Resource type index
$\mathcal{R}_n^{ava}(t) = \bigcup\limits_i \{R_{n,i}^{ava}(t)\}$	Available resources of ES n at time slot t
x_n, y_n	Coordinate of ES n
$x_u(t), y_u(t)$	Coordinate of user u at time slot t
$N_u(t)$	ES serves u at time slot t
s, $\mathcal{S}$	Indexes and set of slices
v, $\mathcal{V}_s$	Indexes and set of VNFs of slice s
$\pi_v(t)$	Place hosts v at time slot t
$\mathcal{R}_u^{req}(t) = \bigcup\limits_i \{R_{u,i}^{req}(t)\}$	Requested resources of u at time slot t
p_s, p_v, p_u	Priority setting of s,v, and u
$\mathcal{R}_v(t) = \bigcup\limits_i \{R_{v,i}(t)\}$	Allocated resources of VNF v at time slot t
P	Basic running price of a resource unit
$\mathbf{CO}_v^{run}(t)$	The running cost of VNF v at time slot t
λ_i	Weights of prices of resource type i
$\mathcal{R}_v^{req}(t) = \bigcup\limits_i \{R_{v,i}^{req}(t)\}$	Requested resources of VNF v at time slot t
$\mathcal{R}_v^{ust}(t) = \bigcup\limits_i \{R_{v,i}^{ust}(t)\}$	Unsatisfied resource of VNF v at time slot t
η	Penalty of an unsatisfied resource unit
$\mathbf{CO}_v^{ust}(t)$	Unsatisfied cost of VNF v at time slot t
δ	Migration penalty
$\Delta_v(t) \in \{0,1\}$	Indicator for measuring if v migrates
$\mathbf{CO}_v^{mig}(t)$	Migration cost of VNF v at time slot t
$\mathbf{CO}_v(t)$	Cost of VNF v at time slot t
$\mathcal{R}_v^{sat}(t) = \bigcup\limits_i \{R_{v,i}^{sat}(t)\}$	Satisfied resources of v at time slot t
$l_{u,v}(t)$	Latency from $N_u(t)$ to $\pi_v(t)$
$l_v(t)$	Total latency of VNF v at time slot t
L_s	Latency constraint of slice s
μ_s	Latency from a user to its connected ES in s
φ_s	Latency between two connected Ess in s
ρ_s	Latency from the ESs to the cloud in s
$\Omega(N_u(t), \pi_v(t))$	Nodes on shortest path from $N_u(t)$ to $\pi_v(t)$
ω	Coefficient to measure revenue based on P
$\mathbf{RN}_v(t)$	Revenue of VNF v at time slot t
$\mathbf{PR}_v(t)$	Profit of VNF v at time slot t
$\Theta_{v,X}(t) \in \{0,1\}$	Indicator measures if $\pi_v(t)$ equals X

connections to form different types of slices to provide users with differentiated services. Here, we consider the RAN part is built under the Open-RAN paradigm realized by the VNFs at ESs with open standards and interfaces, which is witnessed to have more advantages in maintaining the network and reducing the cost by enhancing the flexibility and scalability of the future 6G system [37]. Besides, the BSs are also connected to their neighbors by BS-to-BS links and to the remote cloud by backhaul links, these links are generally built with high-speed optical fibers and have sufficient transmission speed. We assume the core network is built in a cloud-native way and realized by the VNFs deployed at the ESs to achieve a decentralized core network. In the remote cloud swarm, multiple servers with heterogeneous calculation and storage resources are geographically distributed and connected to their neighbors by switches. We regard the cloud swarm as a logic cloud center that can dynamically adjust the number of servers and thus has elastic resources.

When UEs' requests cannot be met, VNFs from related slices will partly move to new hosts and adjust resources to ensure service continuity, known as NSM [14]. Additionally, as NSM triggers occur intermittently, without loss of generality, we model the NSM process from a periodical view to mimic this process while assuming that the NSM triggers occur in the first time slot of each period, as shown in Fig. 2. We aim to explore prediction-based NSM that utilizes system information predictions, such as requested resources and moving trajectories of UEs, to aid NSM decision-making. Specifically, the overall NSM process is separated into the initialization, collection, and migration phases. In the initialization phase, ESs initialize VNFs based on serving users' request priorities and build network slices by grouping users of the same priority into the same slice. After that, in the collection phase, each ES collects and caches the serving UEs' request resources and users' geographical coordinates (defined as *UE cache*). We assume that user behavior data follows identical and independent distribution, thus, the UE cache from multiple slices can be used to train a global prediction model that learns the behavior of all users. In the following migration phase, during each period, the agent of each VNF observes the serving users' requests and positions at the first time slot and predicts their information for the following several time slots based on historical user behavior and the well-trained model. Meanwhile, the agent also observes the ESs' available resources at the first time slot. Based on the current system observations and future predictions, each VNF's agent decides on migration destinations and resource allocations to meet users' demands over the entire period. After the migration, the users dynamically change their positions and required resources until the next period's NSM decision-making. In this paper, we neglect the time for decision-making, VNF migration, and the disk resources for caching user information.

In the considered system, we denote the set of users and ESs as $\mathcal{U}$ and $\mathcal{N}$, the total resources of ES $n \in \mathcal{N}$ as $\mathcal{R}_n$, where $\mathcal{R}_n$ includes I types of resources like CPU, RAM, and disk. Thus, $\mathcal{R}_n$ can be expressed as $\mathcal{R}_n = \bigcup\limits_{i \in \{1,\ldots,I\}} R_{n,i}$, where $R_{n,i}$ denotes the total amount of resource type i of ES n. As the available resources of ESs will dynamically change when allocating or scaling resources for VNFs, similarly, we denote

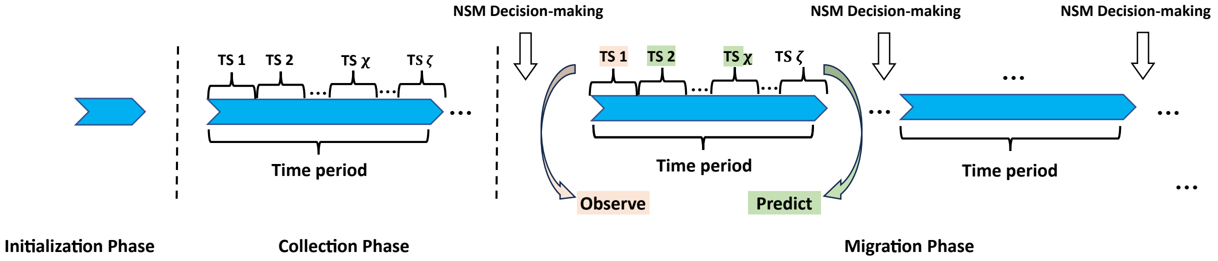


Fig. 2. The illustration of periodical modeling of network slice mobility.

the available resource of ES n at time slot t as $\mathcal{R}_n^{ava}(t) = \underset{i\in\{1,\ldots,I\}}{\cup} R_{n,i}^{ava}(t)$. Considering that ESs have fixed positions while users move dynamically over time slots, we denote the coordinate of ES n as (x_n, y_n) and the coordinate of user $u \in \mathcal{U}$ at time slot t as $(x_u(t), y_u(t))$. Assuming a user can only connect to one ES at each time slot, we denote the ES that user u connects to at time slot t as $N_u(t)$. Generally, $N_u(t)$ should be the ES that is the closest to the user's position [38], which can be obtained by

$$N_u(t) = \arg\min_{n\in\mathcal{N}}\{(x_n - x_u(t))^2 + (y_n - y_u(t))^2\}. \quad (1)$$

We denote the set of slices in the system as $\mathcal{S}$. In the initialization phase, each user is assigned to a slice and then served by the VNFs in that slice. We assume each VNF can only belong to one slice and denote the VNFs of slice s as $\mathcal{V}_s$, thus, we have $\mathcal{V}_s \cap \mathcal{V}_s' = \emptyset, \forall s, s' \in \mathcal{S}$. For VNF $v \in \mathcal{V}_s$, we denote $\pi_v(t) \in \{0, 1, \ldots, |\mathcal{N}|\}$ to indicate the place that hosts v at time slot t, where "0" means v is migrated to the cloud, $1, \ldots, |\mathcal{N}|$ means v is hosted by the corresponding ES[1]. Furthermore, to ensure service continuity, we assume that a UE cannot switch the VNF serving it in the prediction-based NSM scenario, and an illustrative case is shown as Fig. 3. Initially, UE A and UE B are served by VNFs hosted on ESs dedicated to them. As time progresses, UEs move and are served by different ESs, causing VNFs to migrate accordingly. Under this circumstance, UEs still should connect to their corresponding VNFs by serving ESs for uninterrupted service. Thus, users served by a specific VNF should remain constant over time slots and can be denoted as $\mathcal{U}_v$.

The requested resources of user u at time slot t can be denoted as $\mathcal{R}_u(t) = \underset{i\in\{1,\ldots,I\}}{\cup} R_{u,i}^{req}(t)$, where $R_{u,i}^{req}(t)$ represents different types of resources. Moreover, we further consider the priority of users and denote the priority of the request from user u as p_u. The priority of a slice is based on the users' priorities, as well as the VNFs of the slice, thus, we denote p_s and p_v as the priority of slice s and VNF $v \in \mathcal{V}_s$, and have $p_s = p_v = p_u, \forall u \in \mathcal{U}_v, \forall v \in \mathcal{V}_s, \forall s$. Running the network slices for supporting user requests introduces cost and the MNOs get revenue from users, which is relevant to users' requests and the continuity of service [39]. For instance, while some of the users' requests are not satisfied, they may complain about it, which affects MNO's maintenance. We denote the allocated resources of VNF v at time slot t as $\mathcal{R}_v(t) = \underset{i\in\{1,\ldots,I\}}{\cup} R_{v,i}(t)$, which is determined by the requested resources of VNFs and the available resources of the physical host of VNF v.

When a UE tends to access the VNF that serves it, it first connects to its nearest ES through cellular links and then routes to the ES/cloud that hosts the corresponding VNF by optical links. Generally, different slices can have customized network configurations and latency settings for wireless/wired connections by configuring separate network channels/bandwidth [40], [41]. Thus, in slice s, we denote the wireless communication latency for its users to connect to the serving ES as μ_s, the wired communication latency between two directly connected ESs as φ_s. Besides, as the latency to access the remote cloud is generally much higher than local connections, we consider the latency from the VNFs to the remote cloud as a fixed value denoted as ρ_s. Thus, the latency $l_{u,v}(t)$ for user $u \in \mathcal{U}_v$ to connect to VNF v at time slot t can be calculated by

$$l_{u,v}(t) \;\; = \begin{cases} \mu_s + \rho_s, \pi_v(t) = 0 \\ \mu_s + \varphi_s|\Omega(N_u(t), \pi_v(t))|, \; otherwise \end{cases}, \quad (2)$$
$$\forall u \in \mathcal{U}_v, \forall v \in \mathcal{V}_s, \forall s \in \mathcal{S},$$

where $\Omega(N_u(t), \pi_v(t))$ denotes the set of shortest path from ES $N_u(t)$ to facility $\pi_v(t)$. Moreover, considering the customized latency requirements of slices, we denote the latency constraint of slice s as $L_s, \forall s \in \mathcal{S}$ and have $l_{u,v}(t) \le L_s$, which indicates that the access latency of each user should meet the latency requirements of slice s. As a result, we can derive the total latency of VNF v at time slot t for serving all its users as $l_v(t) = \sum_{u\in\mathcal{U}_v} l_{u,v}(t)$.

B. Problem Formulation

We denote the running price of a resource unit as P, and the running cost of VNF v at time slot t can be calculated by

$$\mathbf{CO}_v^{run}(t) = p_v \frac{1}{l_v(t)} \mathrm{P} \sum_{i=1}^{I} \lambda_i R_{v,i}(t), \forall v \in \mathcal{V}_s, \forall s, \forall t, \quad (3)$$

where λ_i denotes the weights for measuring the prices of resource type i. The requested resources of VNF v at time slot t can be the sum of the requests from its users and can be denoted as $\mathcal{R}_v^{req}(t) = \underset{i\in\{1,\ldots,I\}}{\cup} R_{v,i}^{req}(t)$, where $R_{v,i}^{req}(t)$ can be calculated by $R_{v,i}^{req}(t) = \sum_{u\in\mathcal{U}_v} R_{u,i}^{req}(t)$. As the allocated resources of VNFs are determined when performing the NSM process and the requested resources of users change timely, the requested resources of VNFs can be more than their allocated

[1] Here, $|\cdot|$ denotes the number of nodes in the set.

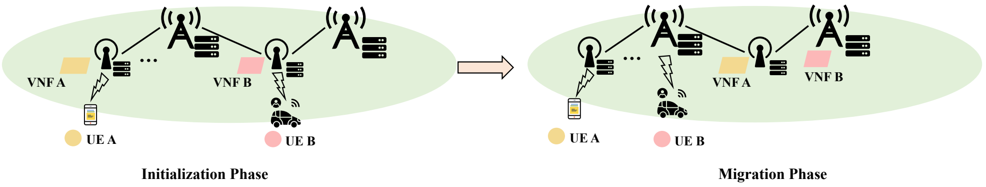


Fig. 3. The illustration of user mobility and slice migration.

resources when users have spike resource requests, introducing unsatisfied resources. We denoted the unsatisfied resources of VNF v at time slot t as $\mathcal{R}_v^{ust}(t) = \underset{i\in\{1,\ldots,I\}}{\cup} R_{v,i}^{ust}(t)$. Intuitively, the unsatisfied resources of type i should be 0 when the requested resources are less than the allocated resources, otherwise should be a difference between them, thus, we can derive $R_{v,i}^{ust}(t) = \max\{R_{v,i}^{req}(t) - R_{v,i}(t), 0\}$. We denote η as the penalty of an unsatisfied resource unit, thus, the unsatisfied cost of VNF v at time slot t can be calculated by

$$\mathbf{CO}_v^{ust}(t) = \eta \sum_{i=1}^{I} \lambda_i R_{v,i}^{ust}(t), \forall v \in \mathcal{V}_s, \forall s, \forall t. \tag{4}$$

Besides, as the VNFs' migration from one host to another may cause user service switch and resource reconfiguration, as well as service replacement, we denote δ as the migration penalty to measure the cost of MNOs to perform VNF migration; to indicate if VNF v migrates to another physical host at time slot t, we further denote $\Delta_v(t) \in \{0,1\}$, where 1 means VNF v migrates to another host and 0 means not. Thus, $\Delta_v(t)$ can be calculated by

$$\Delta_v(t) \ = \begin{cases} 0, & \pi_v(t) = \pi_v(t-1) \\ 1, & otherwise \end{cases}, \forall v \in \mathcal{V}_s, \forall s, \forall t, \tag{5}$$

thus, the migration cost of VNF v at time slot t can be calculated by

$$\mathbf{CO}_v^{mig}(t) = \delta \Delta_v(t), \forall v \in \mathcal{V}_s, \forall s, \forall t, \tag{6}$$

and the cost of VNF v at time slot t can be calculated as

$$\mathbf{CO}_v(t) = \mathbf{CO}_v^{run}(t) + \mathbf{CO}_v^{ust}(t) + \mathbf{CO}_v^{mig}(t), \forall v \in \mathcal{V}_s, \forall s, \forall t. \tag{7}$$

At the same time, the revenue from the VNFs should be measured not only based on the satisfied resources of users but also the requests' priorities and the latency for satisfying the requests. Generally, the higher priority will have higher prices from the MNOs, introducing higher revenue [1]. We denote the satisfied resources of v at time slot t as $\mathcal{R}_v^{sat}(t) = \underset{i\in\{1,\ldots,I\}}{\cup} R_{v,i}^{sat}(t)$. When the requested resources are less than the allocated resources, the satisfied resources should be the requested resources; otherwise, the users can only be fulfilled by the allocated resources, and the satisfied resources are then equal to the allocated resources. Thus, for resource type i, the satisfied resources of v at time slot t can be calculated by $R_{v,i}^{sat}(t) = \min\{R_{v,i}^{req}(t), R_{v,i}(t)\}$. We denote the coefficient to measure the revenue based on P as ω, thus, the revenue of VNF v at time slot t can be calculated as

$$\mathbf{RN}_v(t) = \omega p_v \frac{1}{l_v(t)} \mathrm{P} \sum_{i=1}^{I} \lambda_i R_{v,i}^{sat}(t), \forall v \in \mathcal{V}_s, \forall s, \forall t. \tag{8}$$

We aim to maximize the long-term profit of the MNOs, which is affected by satisfied user requests, latencies for service, and request priority. Denoting the profit of VNF v at time slot t as $\mathbf{PR}_v(t)$ calculated by $\mathbf{PR}_v(t) = \mathbf{RN}_v(t) - \mathbf{CO}_v(t)$, and $\Theta_{v,X}(t) \in \{0,1\}$ as an indicator for measuring if $\pi_v(t)$ equals X or not, thus, the problem in this paper can be formulated as

$$\max_{\pi_v(t), \mathcal{R}_v(t)} \sum_t \sum_{s\in\mathcal{S}} \sum_{v\in\mathcal{V}_s} \mathbf{PR}_v(t), \tag{9a}$$

$$s.t. \sum_{s\in\mathcal{S}} \sum_{v\in\mathcal{V}_s} \Theta_{v,0}(t) R_{v,i}(t) \le R_i^C, \forall i, \forall t, \tag{9b}$$

$$\sum_{s\in\mathcal{S}} \sum_{v\in\mathcal{V}_s} \Theta_{v,n}(t) R_{v,i}(t) \le R_{n,i}, \forall i, \forall n, \forall t, \tag{9c}$$

$$l_{u,v}(t) \le L_s, \forall u \in \mathcal{U}_v, \forall v \in \mathcal{V}_s, \forall s, \forall t, \tag{9d}$$

$$\pi_v(t) \in \{0, \ldots, |\mathcal{N}|\}, \Delta_v(t) \in \{0,1\}, \forall v \in \mathcal{V}_s, \forall s, \forall t, \tag{9e}$$

$$\mathcal{V}_s \cap \mathcal{V}_s' = \emptyset, \forall s, s' \in \mathcal{S}, \tag{9f}$$

$$\sum_{i=1}^{I} \lambda_i = 1. \tag{9g}$$

Here, (9b) and (9c) ensure that the allocated resources of VNFs in the physical host will not exceed the total resources of the host; (9d) ensures that the serving latency of users in slices meet the SLA requirements; (9e) ensures that each VNF selects only one host for migration and has only two distinct migration statuses at each time slot; (9f) ensures that each VNF can only be assigned to one slice; (9g) ensures a thorough weight setting for measuring the prices of resources.

The above-formulated problem is a Mixed Integer Linear Programming (MILP) problem due to the integer variables $\Theta_{v,X}(t), \Delta_v(t), \pi_v(t)$ and the continuous variable $R_{v,i}(t)$, which is proved to be NP-hard [42], we consider using a DRL-based scheme integrated with the FL paradigm to solve this problem considering: *1)* Users have dynamically changing requests and positions, conventional heuristic schemes are hard to cope with new user requests, in contrast, DRL has strong generalization capabilities, well-trained DRL agents can efficiently deal with the dynamic environment; *2)* In a 6G system with massive BSs that continue to grow, the dimensions of the problem will grow with the number of slices, heuristic schemes in this case will spend more time searching for the solutions, which means that it suffers a significant performance degradation in a finite time scale.

IV. Prediction-based Network Slice Mobility Framework

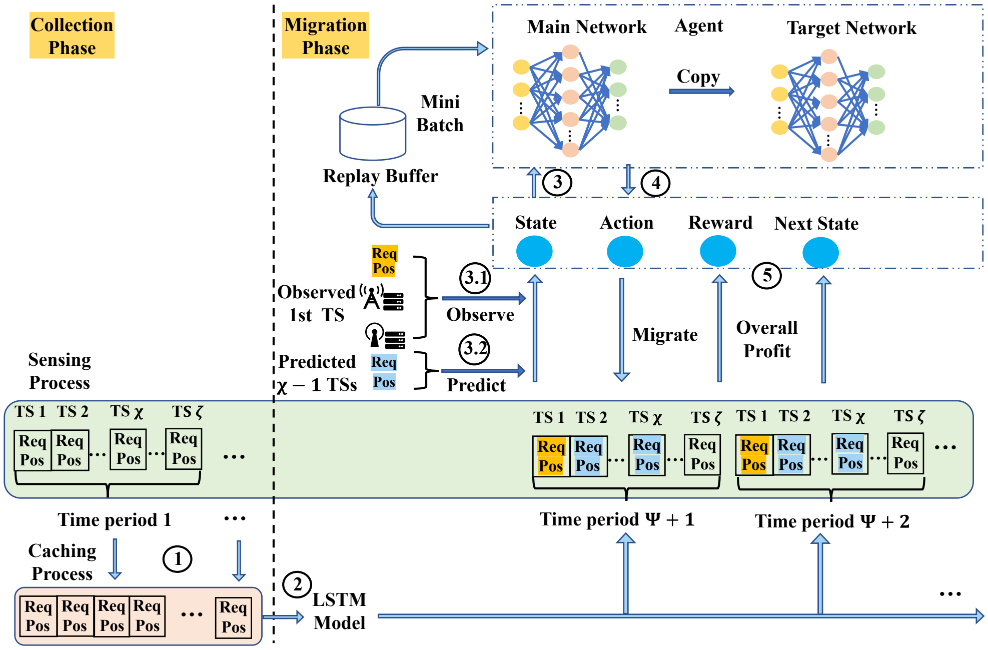


Fig. 4. The illustration of the overall framework in prediction-based slice mobility.

In this section, we introduce the framework design, elaborate on the model for learning users' behaviors for future information prediction, solve the problem using a DDQN-based method with the predictions, and finally integrate the training processes with the FL paradigm.

A. Overall Framework Design

In our proposed framework, during the collection phase, we have two independent processes of each VNF, i.e., the sensing and caching processes, as shown in Fig. 4. Specifically, the sensing process continuously collects the requested resources and position information of the serving users and periodically sends the information to the caching process. After that, the caching process caches the user information and all the VNFs collaboratively train a global user behavior prediction model. During each period in the following migration phase, the VNFs observe serving users' information and the available resources of system facilities at the first time slot and call the prediction model to predict the user information of the following several time slots. Based on the observation and prediction information, each VNF's agent determines which physical host to migrate and how many resources should scale up/down. This decision-making process aims to maximize the MNOs' long-term profit. We transform this problem into an MDP to solve it using the DRL-based scheme, where the state information includes the observation of the first time slot and predictions of the following several time slots, and the agent of each VNF performs action as selecting the physical host for NSM. Besides, the allocated resources are determined based on the chosen host and user requests. After the migration, each VNF stays with the chosen host for the overall period, and the total profit for this period is set as the reward.

B. User Behavior Prediction

We adopt the LSTM model for user request/position prediction. Specifically, as a special morph of the conventional recurrent neural networks (RNN), LSTM introduces the mechanism of the gate to cope with the long-distance dependency problem by adding internal LSTM cell loops. As a result, LSTM can significantly reduce learning difficulty by providing long-term memory for valuable information [43], [44].

The training process of the LSTM model is elaborated in Algorithm 1. Specifically, we consider each period T consists of ζ time slots and the collection phase consists of Ψ periods for collecting user information as training data before the agent starts the NSM decision-making. To obtain the training data, we firstly combine the user request and position information from the Ψ periods as a sequence, denoted as $\mathbf{D}^{cob} = \underset{\forall u \in \mathcal{U}}{\cup} \mathbf{D}_u^{cob}$, where $\mathbf{D}_u^{cob}$ can be expressed as

$$\begin{aligned}&\underbrace{\{\mathcal{R}_u^{req}(1), x_u(1), y_u(1), \ldots, \mathcal{R}_u^{req}(\zeta), x_u(\zeta), y_u(\zeta)}_{\textbf{Time period 1}}, \ldots,\\ &\mathcal{R}_u^{req}(\Psi\zeta-\zeta+1), x_u(\Psi\zeta-\zeta+1), y_u(\Psi\zeta-\zeta+1), \ldots,\\ &\mathcal{R}_u^{req}(\Psi\zeta), x_u(\Psi\zeta), y_u(\Psi\zeta)\}.\end{aligned} \tag{10}$$

After that, we split the combined sequence $\mathbf{D}_{cob}$ to be a dataset for training the LSTM model. Considering each training sample consists of θ time slots' user request and position information, for each user u, we can split $\Psi\zeta - \theta$ training samples expressed as $\mathbf{X}_{u,1} = \{\mathcal{R}_u^{req}(1), x_u(1), y_u(1) \ldots, \mathcal{R}_u^{req}(\theta), x_u(\theta), y_u(\theta)\}$, $\mathbf{X}_{u,2} = \{\mathcal{R}_u^{req}(2), x_u(2), y_u(2) \ldots, \mathcal{R}_u^{req}(\theta+1), x_u(\theta+1), y_u(\theta+1)\}$, …, $\mathbf{X}_{u,\Psi\zeta-\theta} = \{\mathcal{R}_u^{req}(\Psi\zeta-\theta), x_u(\Psi\zeta-\theta), y_u(\Psi\zeta-\theta), \ldots, \mathcal{R}_u^{req}(\Psi\zeta-1), x_u(\Psi\zeta-1), y_u(\Psi\zeta-1)\}$. Moreover, the corresponding testing samples can be expressed as $\mathbf{Y}_{u,1} = \{\mathcal{R}_u^{req}(\theta+1), x_u(\theta+1), y_u(\theta+1)\}$,

$\mathbf{Y}_{u,2} = \{\mathcal{R}^{req}_u(\theta+2), x_u(\theta+2), y_u(\theta+2)\}$, ..., $\mathbf{Y}_{u,\Psi\zeta-\theta} = \{\mathcal{R}^{req}_u(\Psi\zeta), x_u(\Psi\zeta), y_u(\Psi\zeta)\}$ (Line 1). The training dataset and testing dataset will be sent to the model for updating the parameters by the backward propagation algorithm and adaptive moment estimation (Adam) optimizer [45], [46] based on the mean squared error (MSE) loss between the testing data and model prediction (Lines 3-6). After the training process, we utilize the trained LSTM model for future information inference during the migration phase, specifically, for each period T, at the first time slot T_1 we set the historical stack as the user request and position information of current time slot and the previous $\theta - 1$ time slots, i.e., from time slot $T_1 - \theta + 1$ to T_1 (Line 8). As the trained model can only predict one time slot's future information and to support multiple time slots' prediction, we update the historical stack based on the prediction information iteratively (Lines 10-14). We set the prediction order as $\chi \in [0, \zeta - 1]$, with 0 indicating no future prediction and $\zeta - 1$ representing prediction for the entire period. Thus, the predicted user information with χ time slots can be obtained as $\mathbf{P}_u(T) = \{\mathcal{R}^{pre}_u(T_2), x^{pre}_u(T_2), y^{pre}_u(T_2), \ldots, \mathcal{R}^{pre}_u(T_{\chi+1}), x^{pre}_u(T_{\chi+1}), y^{pre}_u(T_{\chi+1})\}$, which will then be combined with the observations at the first time slot T_1 to determine the NSM strategies.

C. DDQN-based NSM Decision-making

In this part, we model the decision-making process of each VNF as an MDP, introduce the state, action, and reward settings, and aim to maximize the value function for improving the long-term profit of the MNOs.

1) VNF State: For each period T in the migration phase, each VNF v observes the serving users' request and position information at the first time slot T_1 and receives the prediction information $\mathbf{P}_u(T)$. To determine where to migrate and how many resources should be allocated, VNF v also observes the available resources of all ESs. Thus, the state of VNF v in period T should include the available resources of the ESs, the serving users' request and position information at the first time slot T_1, and the predicted user information of following χ time slots. Moreover, considering each VNF may serve different numbers of users, to unify the state space, we extend the state to cover at most $|\mathcal{U}|$ users' information. If the serving users are less than $|\mathcal{U}|$, we set the corresponding value in the state space as "-1". As each serving user's information includes I types of resources and 2 position coordinates for $\chi + 1$ time slots, the state information of v should be expressed as

$$\mathbf{S}_v(T) = \Big(\bigcup_{\forall n\in\mathcal{N}} \mathcal{R}^{ava}_n(T_1), \bigcup_{\forall u\in\mathcal{U}_v} \{\mathcal{R}_u(T_1), x_u(T_1), y_u(T_1), \mathbf{P}_u(T)\}, \underbrace{-1,\ldots,-1}_{(I+2)\times(\chi+1)\times(|\mathcal{U}|-|\mathcal{U}_v|)}\Big). \quad (11)$$

2) VNF Action: VNF v determines where to migrate based on the state information, and the allocated resources should be decided based on the current and future requested resources and the available resources of the chosen host. We denote the action of VNF v at period T as $\mathbf{A}_v(T)$ to represent where to migrate, i.e., to ES $1, \ldots, |\mathcal{N}|$, or to the cloud 0. Thus, we have $\pi_v(T_1) = \pi_v(T_2) = \ldots = \pi_v(T_\zeta) = \mathbf{A}_v(T)$. Moreover, the allocated resources should jointly consider the requested resources of the first time slot and the prediction of the further time slots and should be less than the available resources of the chosen host, thus, we have

$$\mathcal{R}_v(T_1) = \mathcal{R}_v(T_2) = \ldots = \mathcal{R}_v(T_\zeta) = \bigcup_{i=\{1,2,\ldots,I\}} \min\{R^{ava}_{\mathbf{A}_v(T),i}(T_1), \frac{R^{req}_{v,i}(T_1) + \sum_{t=T_2}^{T_{\chi+1}} R^{pre}_{v,i}(t)}{1+\chi}\}. \quad (12)$$

3) VNF Reward: To maximize the long-term system profit, in each period T, we set the reward as the overall profit of the period, we denote the reward of period T as $\mathbf{R}_v(T)$ and can be derived by $\mathbf{R}_v(T) = \sum_{t=T_1}^{T_\zeta} \mathbf{PR}_v(t)$. We define the NSM decision-making policy as the mapping from the state to the possible actions, denoted as κ, where $\kappa(\mathbf{A}|\mathbf{S}_v(T))$ indicates the possibility of taking action $\mathbf{A}$ with state $\mathbf{S}_v(T)$ under the policy κ. The VNFs aim to find an optimal policy to maximize the long-term system profit, and the value function is given as

$$V(\mathbf{S}) = \mathbb{E}\left[\sum_{\Psi}^{\infty} \gamma^{T-1} \mathbf{R}_v(T) | \mathbf{S}_v(\Psi) = \mathbf{S}\right], \quad (13)$$

Algorithm 1: LSTM Training and Inference Algorithm

```
   Input: $\zeta$, $\Psi$, $\chi$, $\theta$, $\mathbf{D}^{cob}_u, \forall u \in \mathcal{U}$.
1  Initialize: Split $\mathbf{D}^{cob}_u, \forall u \in \mathcal{U}$ to be training samples
     and testing samples of user $\forall u \in \mathcal{U}$, obtain
     $\mathbf{X}_{u,1}, \mathbf{X}_{u,2}, \ldots, \mathbf{X}_{u,\Psi\zeta-\theta}$ and
     $\mathbf{Y}_{u,1}, \mathbf{Y}_{u,2}, \ldots, \mathbf{Y}_{u,\Psi\zeta-\theta}$.
2  for $u \in \mathcal{U}$ do
3      for Each train epoch do
4          With $\mathbf{X}_{u,1}, \mathbf{X}_{u,2}, \ldots, \mathbf{X}_{u,\Psi\zeta-\theta}$, fit the LSTM
             model, obtain $\hat{\mathbf{Y}}_{u,1}, \hat{\mathbf{Y}}_{u,2}, \ldots, \hat{\mathbf{Y}}_{u,\Psi\zeta-\theta}$.
5          Calculate the MSE loss between
             $\mathbf{Y}_{u,1}, \mathbf{Y}_{u,2}, \ldots, \mathbf{Y}_{u,\Psi\zeta-\theta}$ and
             $\hat{\mathbf{Y}}_{u,1}, \hat{\mathbf{Y}}_{u,2}, \ldots, \hat{\mathbf{Y}}_{u,\Psi\zeta-\theta}$.
6          Update the parameters of LSTM by Adam
             optimizer and backward propagation method.
7      for Each period $T > \Psi$ do
8          Get historical stack sequence with length $\theta$ as
             $\mathbf{H}_u(T) = \{\mathcal{R}^{req}_u(T_1-\theta+1), x_u(T_1-\theta+1),$
             $y_u(T_1-\theta+1) \ldots, \mathcal{R}^{req}_u(T_1), x_u(T_1), y_u(T_1)\}$.
9          Set $\mathbf{P}_u(T) = \emptyset$ to store the prediction of user
             information in period $T$.
10         for time slot $t \in \{T_2, T_3, \ldots, T_{\chi+1}\}$ do
11             Input the historical stack $\mathbf{H}_u(T)$ to the
                 LSTM to obtain the predicted information
                 $\{\mathcal{R}^{pre}_u(t), x^{pre}_u(t), y^{pre}_u(t)\}$ of time slot $t$.
12             Pop out the first record of historical stack
                 $\mathbf{H}_u(T)$.
13             Update $\mathbf{H}_u(T) \leftarrow$
                 $\mathbf{H}_u(T) + \{\mathcal{R}^{pre}_u(t), x^{pre}_u(t), y^{pre}_u(t)\}$.
14             Update $\mathbf{P}_u(T) \leftarrow$
                 $\mathbf{P}_u(T) + \{\mathcal{R}^{pre}_u(t), x^{pre}_u(t), y^{pre}_u(t)\}$.
   Output: $\mathbf{P}_u(T), \forall T, \forall u \in \mathcal{U}$.
```

where γ is a discount factor. Based on the Bellman function [47], the value function can be further transformed to

$$V(\mathbf{S}) = \sum_{\mathbf{A}} \kappa(\mathbf{A}|\mathbf{S}) \left[\mathbf{R} + \gamma \sum_{\mathbf{S}'} Pr\{\mathbf{S}'|(\mathbf{S},\mathbf{A})\} \cdot V(\mathbf{S}') \right], \quad (14)$$

where $\mathbf{A}$ is the action the VNF v takes in state $\mathbf{S}$, $\mathbf{R}$ denotes the received reward, and $\mathbf{S}'$ represents the possible next state. Thus, the above formulated problem can be transformed to

$$\begin{aligned} &\mathbf{max}\ V(\mathbf{S}), \\ &\quad s.t.\ \text{(9b), (9c), (9d), (9e), (9f), (9g).} \end{aligned} \quad (15)$$

In this paper, we use the DDQN model to train agents to avoid possible overestimation of the conventional DQN. Specifically, the DDQN model leverages two neural networks (target network and main network) for action selection and evaluation, and the Q-function can be given as

$$\begin{aligned} Q(\mathbf{S}_v(T), \mathbf{A}_v(T)) = \mathbf{R}_v(T) + \gamma \sum_{\mathbf{S}_v(T+1)} Pr\{\mathbf{S}_v(T+1)| \\ \mathbf{S}_v(T), \mathbf{A}_v(T)\} \cdot \max_{\mathbf{A}_v(T+1)} Q(\mathbf{S}_v(T+1), \mathbf{A}_v(T+1)). \end{aligned} \quad (16)$$

We adopt deep neural network (DNN) to approximate $Q(\mathbf{S}_v(T), \mathbf{A}_v(T))$ and update the parameters of DNN by the stored experience in the replay buffer. Let $Q_\iota(\mathbf{S}_v(T), \mathbf{A}_v(T); \beta(T))$ denote the DDQN model with parameters in episode ι, we have

$$\begin{aligned} &Q_{\iota+1}(\mathbf{S}_v(T), \mathbf{A}_v(T); \beta(T)) = \\ &\quad Q_\iota(\mathbf{S}_v(T), \mathbf{A}_v(T); \beta(T)) + \alpha^\iota \cdot \{\mathbf{R}_v(T) + \gamma \cdot Q_\iota(\mathbf{S}_v(T+1), \\ &\quad \underset{\mathbf{A}_v(T+1)}{\arg\max}\ Q_\iota(\mathbf{S}_v(T+1), A_v(T+1); \beta(T)); \hat{\beta}(T)) - \\ &\quad Q_\iota(\mathbf{S}_v(T), \mathbf{A}_v(T); \beta(T))\}, \end{aligned} \quad (17)$$

where α^ι denotes the learning rate, $\beta(T)$ and $\hat{\beta}(T)$ denote the parameters of the main network and the target network. Thus, the main network's loss function used for updating the parameters by gradient descent can be expressed as

$$L(\beta(T)) = \sum_{(\mathbf{S}_v(T), \mathbf{A}_v(T)) \in \mathcal{B}_T} (y_\iota - Q_\iota(\mathbf{S}_v(T), \mathbf{A}_v(T); \beta(T)))^2, \quad (18)$$

where $y_\iota = \mathbf{R}_v(T) + \gamma \cdot Q_\iota(\mathbf{S}_v(T+1), \underset{\mathbf{A}_v(T+1)}{\arg\max}\ Q_\iota(\mathbf{S}_v(T+1), A_v(T+1); \beta(T)); \hat{\beta}(T))$. Here, $\mathbf{R}_v(T)$ denotes the reward in episode ι, $\mathcal{B}_T$ denotes a mini-batch. As a result, Algorithm 2 illustrates the proposed DDQN-based NSM process.

Algorithm 2: DDQN-Based NSM Algorithm

```
Input: $\mathcal{N}$, $\zeta$, $P_u(T), \forall u \in \mathcal{U}$.
Initialize: The Q-function $Q(\mathbf{S}_v(T), \mathbf{A}_v(T); \beta(T))$ of
  the target network with random $\beta(T)$, the decay rate
  of $\epsilon$ as $\xi$, the episode index $\iota = 1$.
for $\iota \leq$ Epsisode Number do
    Get the system state $\mathbf{S}_v(T)$ from the environment.
    With probability $\epsilon$, select an action $\mathbf{A}_v(T)$
      randomly, otherwise select
      $\mathbf{A}_v(T) = \underset{\mathbf{A}_v(T)}{\arg\max} Q_\iota(\mathbf{S}_v(T), \mathbf{A}_v(T); \beta(T))$.
    VNF $v$ migrates to the corresponding facility that
      $\mathbf{A}_v(T)$ represents with allocated resource
      calculated by (12).
    Set $\Delta_v(T_1) = 1$ if $\mathbf{A}_v(T)$ represents the current
      facility else 0.
    Set $\Delta_v(T_2) = \Delta_v(T_3) = \ldots = \Delta_v(T_\zeta) = 0$.
    Calculate $\mathbf{R}_v(T)$ during period $T$ and get the
      next-state $\mathbf{S}_v(T+1)$.
    Store the data $(\mathbf{S}_v(T), \mathbf{A}_v(T), \mathbf{R}_v(T), \mathbf{S}_v(T+1))$
      in the replay buffer.
    Randomly sample from the replay buffer.
    Calculate the loss function $L(\beta(T))$ by (18).
    Update $\beta(T)$ to minimize $L(\beta(T))$ by gradient
      descent.
    Update $\epsilon \leftarrow e^{-\iota/\xi}$ and $\iota \leftarrow \iota + 1$.
    Update the parameters in the target network
      periodically, i.e., $\hat{\beta}(T) \leftarrow \beta(T)$ after several
      episodes.
```

D. Federated Learning Framework Integration

Each VNF has to decide the NSM strategies based on the prediction model and the DRL agent, which will be trained using the user data collected by the ESs. However, it is not reasonable to train the prediction model and the DRL agents individually due to: *1)* Each VNF can only collect very few user data in each period, which will take a long time for the ES to collect enough user data for model training; *2)* Few training data samples will unavoidably lead to model overfitting. Moreover, uploading all the original user data to the centralized cloud for model training will induce high communication costs and possible user data leakage.

To cope with these problems, we consider using the FL paradigm for model training and describe the FL process of training the LSTM model and the DRL agents as follows. In the collection phase, all the VNFs collect the user information, including user requests and positions for Ψ periods. After that, VNFs train the LSTM model locally and upload the model parameters to the cloud for parameter aggregation, which is generally achieved by the FedAvg method [48], and then the aggregated new parameters will be broadcast to all VNFs. In the migration phase, the VNFs send the parameters of local DRL agents instead. We use τ to denote the iteration index for model training and update. Denote each VNF v has a dataset $\mathcal{D}_v$, which can be the user information for prediction model training or experience for NSM decision-making policy learning. For each sample d in $\mathcal{D}_v$, we denote the loss function as $l_d(w)$. Here, $l_d(w)$ can be the MSE of the predicted user information of user u, i.e., $\hat{\mathbf{Y}}_{u,1}, \hat{\mathbf{Y}}_{u,2}, \ldots, \hat{\mathbf{Y}}_{u,\Psi\zeta-\theta}$ and the real user information $\mathbf{Y}_{u,1}, \mathbf{Y}_{u,2}, \ldots, \mathbf{Y}_{u,\Psi\zeta-\theta}$ in the prediction model learning process, and the loss function (18) in DRL learning process. Denote tha parameter of v as w_v, we can calculate the local loss function at VNF v by

$$L_v(w_v) = \frac{1}{|\mathcal{D}_v|} \sum_{d \in \mathcal{D}_v} l_d(w). \quad (19)$$

Algorithm 3: Federated Network Slice Mobility Framework Implementation Algorithm

```
   Input: $\mathcal{N}$, $\zeta$, $P_u(T), \forall u \in \mathcal{U}$.
1  Initialize: Set the available resources of ESs as their
     total resources, i.e., $\mathcal{R}_n^{ava}(0) = \mathcal{R}_n, \forall n \in \mathcal{N}$.
2  for $n \in \mathcal{N}$ do
3      Get the serving users from $\mathcal{U}$ if $N_u(0) = n$,
         denoted as $\mathcal{U}_n(0)$.
4      Classify the serving users according to $p_u$ and
         generate corresponding VNFs.
5  VNFs initialize local prediction model parameters.
6  for $T \leq \Psi$ do
7      Each VNF $v$ collects serving users' information
         $\mathcal{R}_u^{req}(t)$ and $x_u(t), y_u(t), \forall u \in \mathcal{U}_v, \forall t$.
8      $v$ updates the local model's parameter by (21).
9      $v$ sends local model's parameter to the cloud.
10     Cloud aggregates the global parameter of the
         LSTM model by (22) and dispatches to all VNFs.
11 VNFs initialize local DRL agent parameters.
12 for $T \geq \Psi + 1$ do
13     Each VNF $v$ gets $\mathbf{S}_v(T)$, $\mathbf{A}_v(T)$, $\mathbf{R}_v(T)$ and
         $\mathbf{S}_v(T+1)$ based on Algorithm 2.
14     $v$ updates local agent's parameter by (21).
15     $v$ sends local agent's parameter to the cloud.
16     Cloud aggregates the global parameter of DRL
         agents by (22) and dispatches to all VNFs.
17     Perform Lines 7-10 after several periods.
```

We denote and calculate the dataset of all VNFs in the system as $\mathcal{D} = \bigcup_v \mathcal{D}_v$, the global loss function can be given by

$$L(w) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} l_d(w) = \frac{1}{|\mathcal{D}|} \sum_{s \in \mathcal{S}} \sum_{v \in \mathcal{V}_s} |\mathcal{D}_v| L_v(w_v). \quad (20)$$

To find the optimal parameter $w^* = \arg\min_w L(w)$, let each VNF v computes its parameters w_v^τ according to the update rule by

$$w_v^\tau = w_v^{\tau-1} - \varrho \nabla L_v(w_v^{\tau-1}), \quad (21)$$

where $\varrho \leq 0$ denotes the gradient step size, and the global parameter w^τ can be updated by

$$w^\tau = \frac{\sum_{s \in \mathcal{S}} \sum_{v \in \mathcal{V}_s} |\mathcal{D}_v| w_v^\tau}{|\mathcal{D}|}. \quad (22)$$

Thus, based on the analysis above, we can integrate the overall framework with the FL shown as Algorithm 3. In the initialization phase, all the BSs get the serving users by the user positions and receive users' request information, including users' requested resources and priorities, and classify the users into different groups according to their request priorities. Based on the groups (as well as the priorities), the BSs generate network slices and allocated resources for each VNF of slices (Lines 2-4). Note that to avoid the resource unavailable concern in the initialization phase, we consider the users' requested resources to be slight at the beginning. Afterward, in the collection phase, the VNFs initialize the parameters of the prediction model and collect their users' information, including user request and user position, to train the local model by updating the model parameters based on (21), the local parameters are then sent to the cloud for parameter aggregation by (22). The aggregated model will be dispatched back to all VNFs, and this process will iterate until it converges (Lines 6-10). In the migration phase, the VNFs initialize the parameters of their local agents and get the state with current observations and future predictions. Based on the state information, the local agents give the actions under the ϵ-greedy policy, and then VNFs migrate to the chosen host to stay for an overall period. The total profit of this period is set as the reward for the agent, and the next state can be determined at the next period's first time slot. The stored tuples with mini-batch are utilized for local parameter update of the DRL agents based on (21), and the parameters are then aggregated at the cloud based on (22) (Lines 12-16). After several periods, the local prediction model will also be updated based on new user information data and be aggregated and dispatched back to all VNFs (Line 17).

E. Complexity Analysis

In Algorithm 1, Line 1 takes $O((\Psi\zeta - \theta) \times |\mathcal{U}|)$ to split the dataset for $|\mathcal{U}|$ users to get $\Psi\zeta - \theta$ sample pairs. Afterward, Lines 3-6 take $O(EF)$ to train the global LSTM model, where E is the number of training epochs and $O(F)$ is the computation complexity for LSTM training and is related to the number of LSTM parameters [49]. In our case, the input and output dimensions are $I + 2$, indicating the considered resource types and coordinates. Denote the number of LSTM layers as L and each layer consists of U neurons, we have $O(F)$ calculated as $O(LU(4LU + 5I + 13))$. As calculating the MSE loss takes $O(\Psi\zeta - \theta)$ and updating the parameters in each episode takes $O(LU(4LU + 5I + 13))$ similar to LSTM training, the complexity of Algorithm 1 can be calculated as $O(|\mathcal{U}| \cdot ((\Psi\zeta - \theta) + 2ELU(4LU + 5I + 13))$.

The computations of Algorithm 2 mainly come from Lines 4, 12, and 13. In Line 4, when an agent adopts greedy policy to choose the action, it needs to calculate the Q function to obtain the action, which takes $O(|\mathcal{A}_v(T)|) = O(|\mathcal{N}| + 1)$. The complexity of calculating the loss function and updating the parameters is related to the number of agent parameters and the random samples from the replay buffer. Denote the training process has E' episodes, N_ι ramdom samples, and L' DNN layers, each layer has U' neurons, the complexity of Algorithm 2 is $O(((|\mathcal{N}| + 1)E' + E'L'U'N_\iota))$ [50].

In Algorithm 3, Lines 2-4 take $O(|\mathcal{N}||\mathcal{U}|)$ for the ESs to obtain their users and assign them to the corresponding slices according to their priorities. During Lines 6-10, the VNFs collect user information, update the local parameters, and send to the cloud in parallel, which takes $O(\max_v |\mathcal{U}_v|((\Psi\zeta - \theta) + 2ELU(4LU + 5I + 13)))$. Similarly, during Lines 12-16, the VNFs update the parameters of local DRL agents and send them to the cloud, which takes $O((|\mathcal{N}| + 1)E' + E'L'U'N_\iota)$. Thus, the complexity of Algorithm 3, as well as the overall framework can be expressed as $O(|\mathcal{N}||\mathcal{U}| + \max_v |\mathcal{U}_v|((\Psi\zeta - \theta) + 2ELU(4LU + 5I + 13)) + (|\mathcal{N}| + 1)E' + E'L'U'N_\iota)$. As

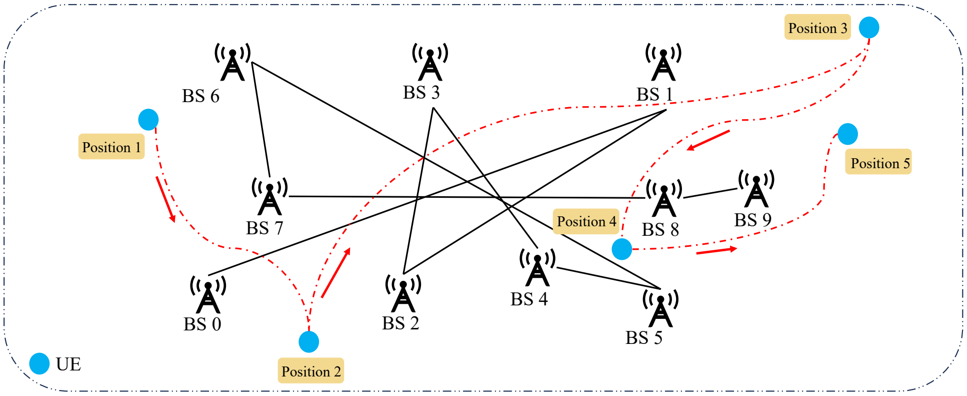


Fig. 5. The demo network topology for 10 BSs and the mobility case of a user for 5 time slots.

TABLE II
BASIC SIMULATION PARAMETERS.

Parameter	Value
The size of area	100×100 M^2
The ES number $\lvert\mathcal{N}\rvert$	10
The user number of each ES	5
Unsatisfied penalty η	50
Migration penalty δ	10
Weights $\lambda_i (i = 1, 2, 3)$	$0.4, 0.4, 0.2$
Priorities	$1, 2, 3$
Total resources of ESs	4,8,128
Latency μ_s	1 - 3 ms
Latency φ_s	3 - 5 ms
Latency ρ_s	100 - 120 ms
Latency constraint L_s	30 - 150 ms
Revenue coefficient ω	150
Run price per resource unit P	100
Time slot number of each period ζ	5
Learning rate	0.001
Batch size	256
Discount factor	0.9
Memory capacity	1024
Epsilon start	0.9
Epsilon end	0.01
Epsilon decay	1000
Hidden layer number	2
Hidden layer neural number	64

the training parameters E, E' *et al.* are generally constant in simulation, when we have $|\mathcal{N}|$ and $|\mathcal{U}|$ growing large enough, the complexity can be closely approximated to $O(|\mathcal{N}||\mathcal{U}|)$. Compared to other baselines without distributed paradigm integration [1], [15], the complexity of the proposed FL-based scheme does not directly increase with the number of slices. Considering the scalability issue in the 6G system [37], the proposed scheme will be feasible to deploy slices in scenarios with a relatively stable number of BSs and UEs.

V. SIMULATION RESULTS

This part presents the simulation settings, baselines, performance metrics, and evaluation results from different perspectives.

A. Settings

We consider a scenario for NSM as an area with 100×100 M^2 consisting of several BSs spread geographically evenly, each BS covers 5 UEs. The network topology of the BSs is generated by the **Networkx** package in Python [51]. As a case, we present the network topology of 10 BSs according to their positions as shown in Fig. 5, in which the black solid lines represent the connection relationships of the BSs. Moreover, at each time slot t, we consider the users in the system randomly move from their current positions to new ones and illustrate the coordinate trajectories $x_u(t), y_u(t)$ of a demo user u for 5 time slots by blue dots and red dash lines. From Fig. 5, we can see the user firstly initializes the request at position 1 (next to BS 6), then continuously moves to position 2 (the middle between BSs 0 and 2), and then to position 3 located at the edge of the scenario, position 4 that with dense BSs serving, and finally to position 5 (next to BS 9). We consider three types of resources for user requests, i.e., the CPU, RAM, and disk resources, and set $I = 3$. Since the resources of ESs are limited, we set the total resources of each ES as 4 CPU cores, 8 GBits RAM, and 128 GBits disk, respectively. The requested resources for user u at time slot $t \geq 1$ are randomly set from 0 to $2t$ CPU cores, $4t$ GBits RAM, and $12t$ disk resources, respectively. To distinguish the users according to their request priorities for slice generation, we set the priorities of users as $p_u \in \{1, 2, 3\}, \forall u$. To evaluate the communication overhead and transmission time for the FL training comparison, we set the upload speed from VNFs to the cloud uniformly as 5 Mbits/s, the download speed as 10 Mbits/s, and the aggregation round for FL training as 10. At last, we do the simulation in a server with two Intel Xeon Gold 6226 2.7 GHz CPUs, 376 GB of RAM, and a Tesla V100 GPU.

Moreover, to measure the customized slice settings, in slice s, we set the wireless communication latency μ_s randomly as $1-3$ ms, the optical communication latency φ_s as $3-5$ ms, and the latency to the remote cloud ρ_s as $100-120$ ms. Additionally, we set the latency constraint L_s randomly from $30-150$ ms. To measure the system cost and profit from the MNO perspective, we set the running price of a resource unit P as 100 and the revenue coefficient to measure the revenue ω as 150. To mimic the considered periodical NSM scenario, we set each period consisting of $\zeta = 5$ time slots and predict 4 time slots except the first time slot and set $\chi = 4$. The migration of VNFs between two time periods will introduce migration costs, and provisioning user requests during an overall time

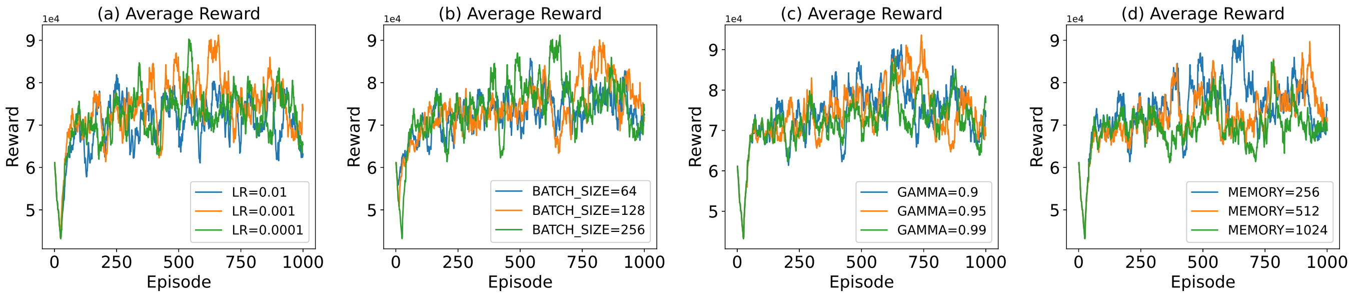


Fig. 6. The average reward of the proposed scheme versus training episodes with different training parameters.

period will introduce unsatisfied costs. To measure these costs, we set the migration penalty $\delta = 10$ and the unsatisfied penalty $\eta = 100$. Finally, we set the weights λ_i for measuring the prices of the CPU, RAM, and disk resources as 0.4, 0.4, and 0.2, respectively. As a result, we summarize the basic simulation parameters, including the system configuration and parameters in LSTM model training and DRL agent training in Table II.

B. Baseline and Performance Metrics

To evaluate our proposed NSM framework, we consider the corresponding baselines as *1)* DRL-based NSM scheme with no future information prediction (**NoPre-DRL**) [15], in which the DRL method is adopted in NSM decision-making and no further user information is provided, which means the DRL agents can only make decisions based on the first time slot' observation in each period; *2)* Simulated annealing (SA)-based NSM scheme with no future information prediction (**NoPre-SA**), in which the SA method with 500 iterations for each state in each episode is adopted in NSM decision-making and no further user information is provided; *3)* Reset NSM scheme with no future information prediction (**NoPre-Reset**) [1], in which the VNFs stay in the original physical host and re-allocate the resources based on the first time slot' user requests during periods; *4)* Random NSM scheme with no future information prediction (**NoPre-Random**), in which the VNFs randomly migrate to a physical host when performing NSM without future prediction information. Additionally, for the proposed prediction-based scheme, to measure how the prediction orders affect the simulation results, we present simulation results with different prediction orders with χ set from 1 to 3 as baselines (**Proposed,** $\chi = 1, 2, 3$); to measure the effect from prediction deviations, we set the prediction model with RNN to get different prediction performances as a baseline (**Proposed, RNN**). Particularly, in extreme cases, to observe the best possible performance of the prediction-based scheme regarding predictions, we collect real users' behavior data after a few rounds of simulations and input the real data as prediction information to the agents to execute the NSM again to achieve an unbiased prediction as a baseline (**Proposed, Real**); to observe the better performance of the prediction-based scheme regarding decision-making, we run the MDP with prediction information as a baseline, which performs 100 iterations for each state that appears in each episode to explore the better actions of agents through exhaustive search (**Proposed, MDP**). Note that these two extreme cases serve only as benchmarks for ideal experimental comparisons. In real-world NSM scenarios, they are hard to apply due to unavoidable prediction bias and extensive iterations. Finally, to demonstrate the performance of the FL training paradigm, we compare the proposed scheme without FL to see how FL can reduce communication overhead and transmission time (**Proposed, NoFL**).

The considered performance metrics should be related to the optimization goal, as well as the intermediate results. Thus, we choose the average long-term system cost $\sum_t \sum_{s\in\mathcal{S}} \sum_{v\in\mathcal{V}_s} \mathrm{CO}_v(t)/\sum_{s\in\mathcal{S}} |\mathcal{V}_s|$, revenue $\sum_t \sum_{s\in\mathcal{S}} \sum_{v\in\mathcal{V}_s} \mathrm{RN}_v(t)/\sum_{s\in\mathcal{S}} |\mathcal{V}_s|$, and profit $\sum_t \sum_{s\in\mathcal{S}} \sum_{v\in\mathcal{V}_s} \mathrm{PR}_v(t)/\sum_{s\in\mathcal{S}} |\mathcal{V}_s|$ as the performance metrics to evaluate the proposed scheme and baselines. Moreover, the average long-term unsatisfied cost $\sum_t \sum_{s\in\mathcal{S}} \sum_{v\in\mathcal{V}_s} \mathrm{CO}_v^{ust}(t)/\sum_{s\in\mathcal{S}}|\mathcal{V}_s|$ is also set as a performance metric to see the different resource allocation strategies between the considered schemes. Note that the system profit is the final goal of optimization and the most important indicator that the MNOs care about among all the performance metrics.

C. Evaluation Results With Different Learning Parameters

We first illustrate the average reward of the DRL agents of the VNFs to show the convergence performance of the proposed prediction-based NSM scheme with different learning rates, batch sizes, discount factors, and memory capacities, as shown in Fig. 6. From Fig. 6(a), we can observe that the average reward of the agents with the learning rate set as 0.01 fluctuates greatly over learning episodes, while the reward of the learning rate set as 0.001 and 0.0001 have similar behavior and finally the reward of different learning rates can converge to be stable near to 0.8×10^5. Particularly, when the learning rate is set as 0.01, the training of agents will occasionally fall into a local optimum, leading to sharp fluctuations in the later stages and rewards hovering around the global optimum. Also, we adopt different batch sizes, discount factors, and memory capacities for agent training, as shown in Fig. 6(b), Fig. 6(c), and Fig. 6(d), respectively. From the figures, we can observe that the average training reward of the DRL agents can converge to a stable stage near 0.8×10^5 after 100 episodes, which suggests that the VNFs can be adapted

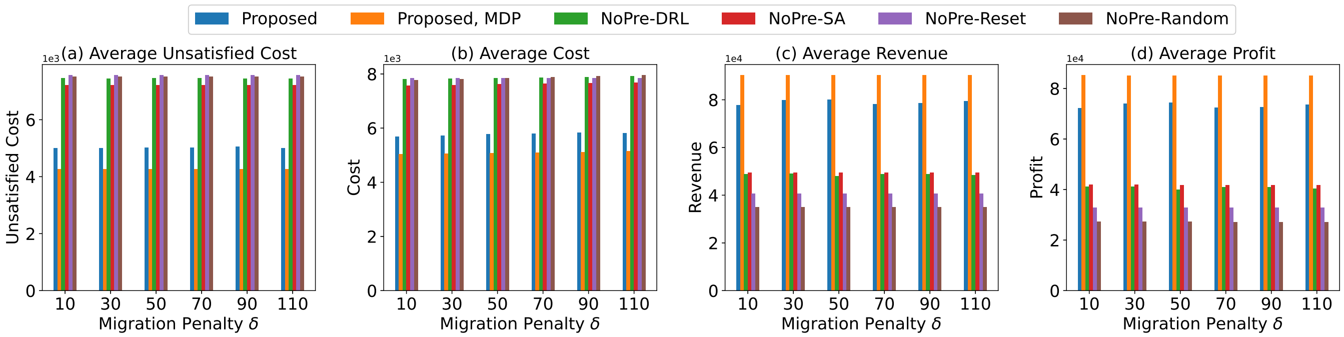


Fig. 7. The unsatisfied cost, cost, revenue, and profit of prediction-based FDRL versus different migration penalties.

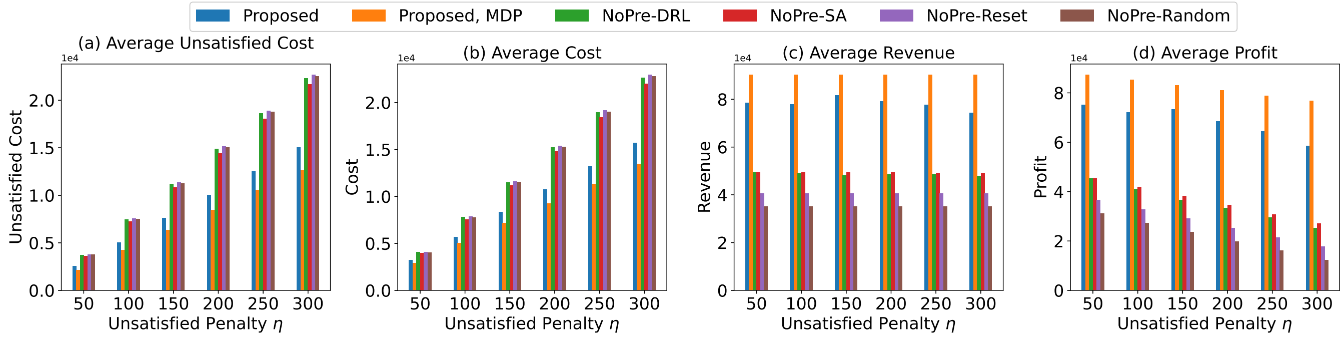


Fig. 8. The unsatisfied cost, cost, revenue, and profit of prediction-based FDRL versus different unsatisfied penalties.

to the environment efficiently and the agents can learn stable strategies to make NSM decisions with prediction information.

D. Evaluation Results Versus Different Penalties

1) The Considered Schemes Adopt Different Migration Penalties: We then compare the average unsatisfied cost, cost, revenue, and profit of the considered schemes with different migration penalties δ, as shown in Fig. 7. From Fig. 7(a) and Fig. 7(b), we can see with the migration penalty δ increases, the unsatisfied cost and cost remain relatively stable, as the migration penalty will not affect the resource allocation strategies that the unsatisfied cost depends on. The proposed scheme and the Proposed MDP scheme have much lower unsatisfied costs and costs compared to others, since in these prediction-based schemes, the agents allocate resources for the VNFs more reasonably with future user requests considered. From Fig. 7(c) and Fig. 7(d), we can see the proposed schemes can achieve the highest system revenue and profit, followed by the NoPre-SA and NoPre-DRL schemes that are very close to each other, the NoPre-Reset scheme, and the NoPre-Random scheme. With the prediction of users' information provided, the proposed schemes can have more intelligent resource allocation strategies to decide on better physical hosts for the VNFs with users' mobility trajectories considered. Moreover, the No-Pre DRL scheme realizes even the same performance compared to the NoPre-SA scheme with 500 iterations and achieves better NSM strategies than the other non-prediction schemes. Overall, compared to the NoPre-DRL, NoPre-SA, NoPre-Reset, and NoPre-Random schemes, the proposed scheme with different migration penalties $\delta \in [10, 30, \ldots, 110]$ can reduce the average unsatisfied cost by up to 49.17%, 44.53% 51.33%, and 50.32%, reduce the average cost by up to 37.24%, 33.05%, 37.76%, and 36.76%, improve the average revenue by up to 40.21%, 38.27%, 49.30%, and 56.23%, and improve the average profit by up to 46.13%, 43.72%, 55.90%, and 63.38%, respectively. Moreover, in this case ($|\mathcal{N}| = 10$) with different δ, the gap in profit with the Proposed MDP scheme is up to 18.08%.

2) The Considered Schemes Adopt Different Unsatisfied Penalties: Besides, we illustrate the performances of the considered schemes with different unsatisfied penalties η, as shown in Fig. 8. From Fig. 8(a), we can observe the unsatisfied cost of the considered schemes increases with η, and the proposed schemes have lower unsatisfied costs than the other schemes. With η increases, the advantages of the proposed schemes become more obvious, since the proposed schemes allocate more resources for the VNFs with future requested resources information in the NSM process, contributing to lower average unsatisfied cost and lower cost. From Fig. 8(c), we can see the average revenues of the considered schemes fluctuate on small scales since the available resources of physical ESs are run out. In this scenario, even with higher unsatisfied penalties, the VNFs cannot be allocated more resources; even if the cloud has elastic resources, it also has higher latency, introducing lower profit. Thus, the allocated resources of VNFs remain stable, as well as the satisfied resources, leading to relatively stable revenue. The proposed schemes have the highest average revenue, followed by the NoPre-SA, NoPre-DRL, NoPre-

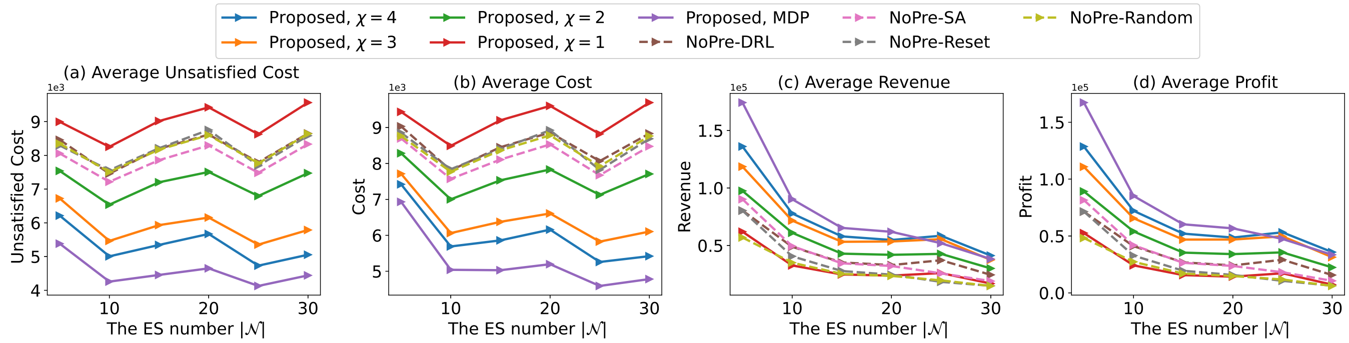


Fig. 9. The performance of considered schemes versus $|\mathcal{N}|$ while the proposed scheme adopts different prediction orders.

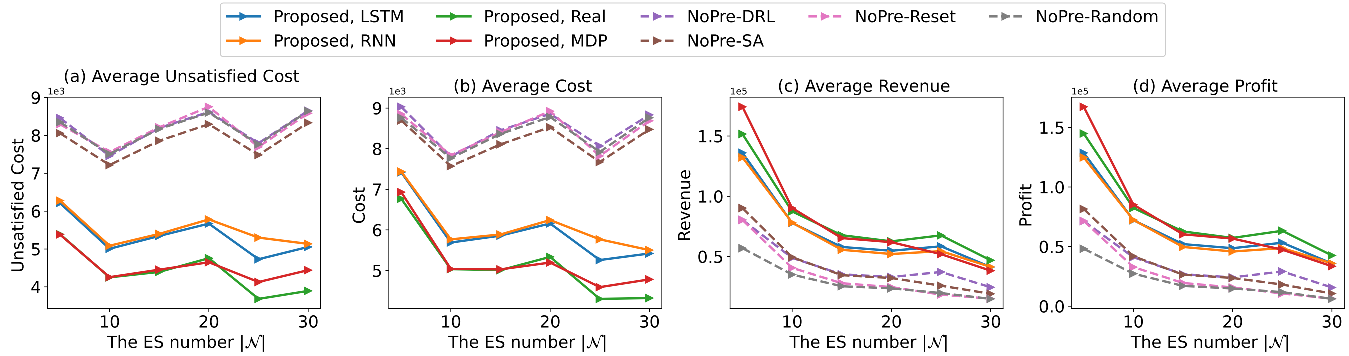


Fig. 10. The performance of considered schemes versus $|\mathcal{N}|$ while the proposed scheme adopts different prediction models.

Reset, and NoPre-Random schemes. Consequently, though the system cost increases heavily with the unsatisfied penalty η, the system profit slightly decreases, as shown in Fig. 8(d). To summarize, compared to the NoPre-DRL, NoPre-SA, NoPre-Reset, and NoPre-Random schemes, the proposed scheme with different unsatisfied penalties $\eta \in [50, 100, \ldots, 300]$ can reduce the average unsatisfied cost by up to 48.96%, 44.14%, 50.97%, and 49.97%, reduce the average cost by up to 44.39%, 40.26% 46.29%, and 45.21%, improve the average revenue by up to 41.08%, 39.49%, 50.23%, and 57.04%, and improve the average profit by up to 56.95%, 53.75%, 69.81%, and 79.01%, respectively. Moreover, in this case ($|\mathcal{N}| = 10$) with different η, the gap in profit with the Proposed MDP scheme is up to 31.11%.

E. Evaluation Results Versus Different Numbers of ESs

1) The Proposed Scheme Adopts Different Prediction Orders: To figure out how the prediction order χ affects the evaluation performance, we set the proposed scheme with different prediction orders χ and further compare the performances of the considered schemes versus different ES numbers, as shown in Fig. 9. From Fig. 9(a) and Fig. 9(b), we can observe that the average unsatisfied cost and cost of the considered schemes vibrate evenly with the increase of the ES number $|\mathcal{N}|$, which suggests that the unsatisfied cost and cost are not affected by the number of ESs, since the number of VNFs also increase with $|\mathcal{N}|$. Particularly, when we have $|\mathcal{N}| = 25$, we find the UEs have relatively fewer requested resources compared to other scenarios, leading to less unsatisfied cost. The Proposed MDP scheme ($\chi = 4$) has the lowest unsatisfied cost and system cost, followed by the proposed scheme with $\chi = 4, 3, 2$, the non-prediction schemes, and the proposed scheme with $\chi = 1$. The non-prediction schemes have close performance to each other, as their resource allocation strategies are based on the first time slot's requests, and their slight differences are the chosen facilities for VNF migration, e.g., choosing an ES with more available resources will contribute to lower unsatisfied cost and cost. Particularly, the proposed scheme with $\chi = 1$ has the highest unsatisfied cost and cost, which indicates that the prediction of $\chi = 1$ provides the VNFs with a negative effect that lets VNFs allocate much fewer resources, leading to higher unsatisfied costs.

From Fig. 9(c) and Fig. 9(d), when $|\mathcal{N}| \in [5, 20]$, we can see that the Propose-MDP scheme has the highest average revenue and profit, followed by the proposed scheme with $\chi = 4, 3, 2$, the NoPre-SA, NoPre-DRL, NoPre-Reset schemes, the proposed scheme with $\chi = 1$, and finally the NoPre-Random scheme. Note that the performances of the Propose-MDP scheme and the NoPre-SA schemes significantly decrease with $\mathcal{N}$, which demonstrates that the exhaustive search-based schemes are hard to deal with scenarios with large numbers of BSs. The average revenue and profit decrease with $|\mathcal{N}|$ since the complexity of network topology increases with the ES number $|\mathcal{N}|$, leading to higher latency between the physical hosts of VNFs and users' connected ESs, causing lower average revenue/profit. When $|\mathcal{N}| = 25$, since the requested resources have a decrease, the agents have more

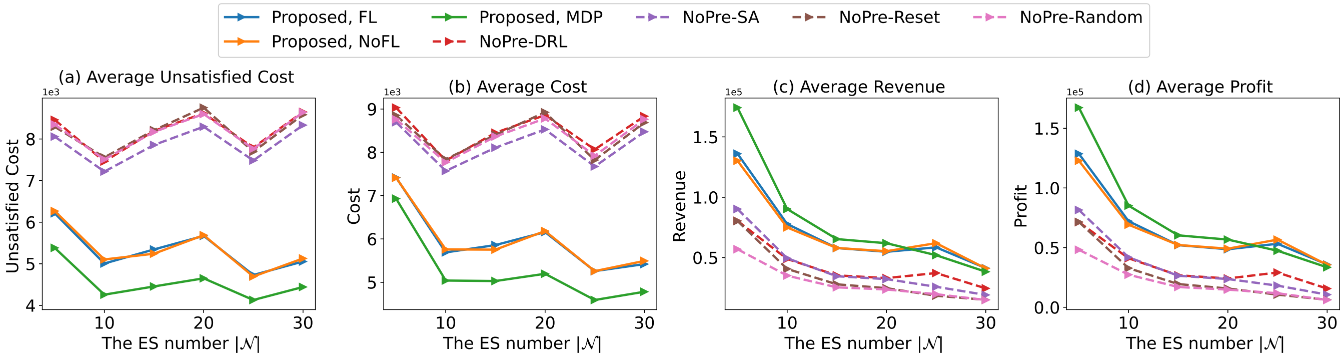
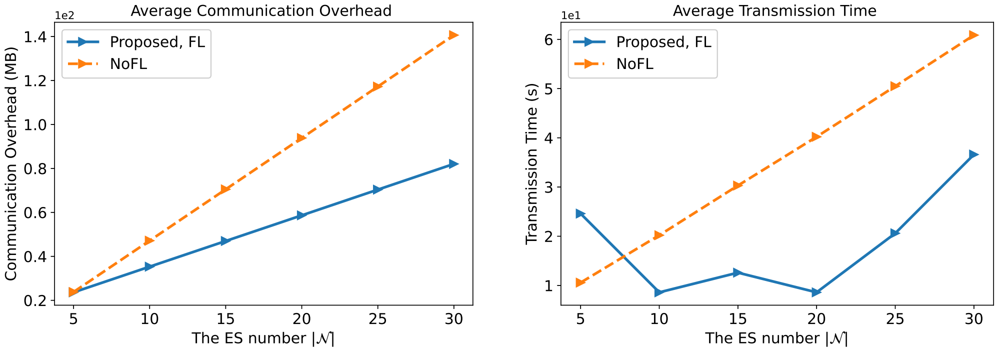


Fig. 11. The performance of considered schemes versus $|\mathcal{N}|$ while the proposed scheme adopts different training paradigms.



(a) Communication Overhead

(b) Transmission Time

Fig. 12. The communication overhead and transmission time versus $|\mathcal{N}|$ when the proposed scheme adopts different training paradigms.

candidate physical hosts to choose from for migration and thus can migrate to physical hosts with lower access latencies, contributing to a slight increase of revenue/profit. When $|\mathcal{N}| \in [25, 30]$, the proposed scheme with $\chi = 4$ has higher revenue/profit compared to the NoPre-MDP scheme, and the NoPre-SA scheme also falls behind the NoPre-DRL scheme. Overall, compared to the NoPre-DRL, NoPre-SA, NoPre-Reset, and NoPre-Random schemes, the proposed scheme with different prediction orders $\chi \in [4, 3, \ldots, 1]$ can reduce the average unsatisfied cost by up to 71.15%, 64.86%, 69.81%, and 70.96%, reduce the average cost by up to 62.97%, 56.36%, 60.26%, and 61.67%, improve the average revenue by up to 40.90%, 55.88%, 68.53%, and 66.34%, and improve the average profit by up to 56.29%, 70.44%, 82.37%, and 82.99%, respectively. Moreover, with $|\mathcal{N}|$ increases from 5 to 30, the gaps in profit with the Proposed MDP scheme are 30.05%, 18.07%, 15.76%, 16.61%, -10.97%, and -6.54%, respectively, which demonstrate that the proposed scheme outperforms the Propose-MDP scheme when $|\mathcal{N}| \geq 25$.

2) The Proposed Scheme Adopts Different Prediction Models: To see how different prediction models affect the NSM performance, we also compare the performances of the considered schemes versus different $|\mathcal{N}|$ where the proposed scheme adopts different prediction models, as shown in Fig. 10. Specifically, we utilize the LSTM model and the RNN model to predict future user information to get different prediction performances and collect real user request data to achieve an unbiased prediction present as the Proposed Real scheme. From Fig. 10(a) and Fig. 10(b) we can observe the average unsatisfied cost and cost of the considered schemes vibrate evenly with the increase of $|\mathcal{N}|$, the Propose-Real scheme gets the lowest average unsatisfied cost and system cost (very close to the Proposed MDP), followed by the proposed scheme with LSTM model and RNN model. The baseline schemes without prediction have very close performance and all get high average unsatisfied cost and system cost compared to the proposed scheme with different models. From Fig. 10(c) and Fig. 10(d), we can see the average revenue and profit decrease with $|\mathcal{N}|$, the performance of the Proposed MDP scheme has the highest system revenue/profit when $|\mathcal{N}| \in [5, 20]$, and the Proposed Real scheme is much better when $|\mathcal{N}| \in [25, 30]$ followed by the proposed scheme with LSTM model and RNN model and non-prediction schemes. To summarize, compared to the NoPre-DRL, NoPre-SA, NoPre-Reset, and NoPre-Random schemes, the proposed scheme with different prediction models adopted can reduce the average unsatisfied cost by up to 122.11%, 113.94%, 120.37%, and 121.86%, reduce the average cost by up to 104.34%, 96.05%, 100.95%, and 102.71%, improve the average revenue by up to 47.90%, 61.81%, 72.76%, and 70.86%, and improve the average profit by up to 63.20%, 75.11%, 85.16%, and 85.67%, respectively. Moreover, with $|\mathcal{N}|$ increases from 5 to 30, the gaps in profit with the Proposed MDP scheme are 15.47%, 3.39%, -3.82%, -0.96%, -25.09%, and -21.31%, respectively, which demonstrate that the proposed scheme with best predictions outperforms the Propose-MDP scheme when

$|\mathcal{N}| \geq 15$.

3) The Proposed Scheme Adopts Different Training Paradigms: Finally, to see how the FL paradigm affects the NSM performance, we compare the performance of the considered schemes versus different $|\mathcal{N}|$ where the proposed scheme adopts different training paradigms, as shown in Fig. 11. Here, we utilize the centralized training paradigm to train the LSTM and DRL models as a baseline in which all the VNFs upload the collected user data in the collection phase and the interactive data in the migration phase to the remote cloud for model training. From Fig. 11(a) to Fig. 11(d) we can observe the NSM performances of the proposed FL-based scheme and the non-FL based scheme are very close to each other, but according to Fig. 12(a) and Fig. 12(b) we can see the non-FL based scheme has much higher communication overhead and transmission time for model training, especially for the cases with more ESs, which demonstrates that the proposed FL-based scheme can significantly reduce the communication overhead and transmission time for model training with the NSM performance ensured. Particularly, in Fig. 12(b) we notice that the transmission time of the proposed FL-based scheme remains relatively stable when we have less than 25 ESs, since in these scenarios we find the number of users each VNF is serving, i.e., $\mathcal{U}_v$ are even, contributing to lower overall local model training time (depends on the largest local model training time among all VNFs). Overall, the proposed FL-based scheme can reduce the communication overhead by up to 71.39% and save the transmission time over 3.67 times compared to non-FL-based schemes.

VI. Conclusion

In this paper, we have investigated the NSM problem from a periodical view to mimic that the VNF should be migrated only after the NSM trigger is detected. We designed a general system model for periodical NSM and formulated the problem of maximizing the long-term profit of MNOs. To solve this problem, we first model the NSM decision-making process as an MDP and creatively propose a prediction-based FDRL framework to solve it. The prediction of further system information is adopted as the suppliance of the system state, and we set the overall feedback for each period as the system reward. By performing extensive experiments, simulation results have demonstrated that our proposed scheme outperforms baseline schemes in improving long-term profit and can significantly reduce communication overhead and save transmission time.

References

[1] H. Yu, Z. Ming, C. Wang, and T. Taleb, "Network slice mobility for 6G networks by exploiting user and network prediction," in *Proc. IEEE International Conference on Communications (ICC)*, May 2023, pp. 4905–4911.

[2] B. Ji, Y. Han, S. Liu, F. Tao, G. Zhang, Z. Fu, and C. Li, "Several key technologies for 6G: challenges and opportunities," *IEEE Commun. Stand. Mag.*, vol. 5, no. 2, pp. 44–51, June 2021.

[3] Y. Liu, Y. Deng, A. Nallanathan, and J. Yuan, "Machine learning for 6G enhanced ultra-reliable and low-latency services," *IEEE Wirel. Commun.*, vol. 30, no. 2, pp. 48–54, Apr. 2023.

[4] T. K. Rodrigues and N. Kato, "Network slicing with centralized and distributed reinforcement learning for combined satellite/ground networks in a 6G environment," *IEEE Wirel. Commun.*, vol. 29, no. 1, pp. 104–110, Feb. 2022.

[5] K. Smida, H. Tounsi, M. Frikha, and Y.-Q. Song, "Fens: Fog-enabled network slicing in SDN/NFV-based IoV," *Wirel. Pers. Commun.*, vol. 128, no. 3, pp. 2175–2202, Sept. 2023.

[6] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, and S. Guo, "A survey of intelligent network slicing management for industrial IoT: Integrated approaches for smart transportation, smart energy, and smart factory," *IEEE Commun. Surv. Tutorials*, vol. 24, no. 2, pp. 1175–1211, Apr. 2022.

[7] Z. Shu, T. Taleb, and J. Song, "Resource allocation modeling for fine-granular network slicing in beyond 5G systems," *IEICE Trans. Commun.*, vol. 105, no. 4, pp. 349–363, Apr. 2022.

[8] X. Tang, L. Zhao, J. Chong, Z. You, L. Zhu, H. Ren, Y. Shang, Y. Han, and G. Li, "5G-based smart healthcare system designing and field trial in hospitals," *IET Commun.*, vol. 16, no. 1, pp. 1–13, Jan. 2022.

[9] S. Karunarathna, S. Wijethilaka, P. Ranaweera, K. T. Hemachandra, T. Samarasinghe, and M. Liyanage, "The role of network slicing and edge computing in the metaverse realization," *IEEE Access*, vol. 11, pp. 25 502–25 530, Mar. 2023.

[10] Y. Sun, S. Qin, G. Feng, L. Zhang, and M. A. Imran, "Service provisioning framework for RAN slicing: user admissibility, slice association and bandwidth allocation," *IEEE Trans. Mob. Comput.*, vol. 20, no. 12, pp. 3409–3422, Dec. 2020.

[11] A. Papa, A. Jano, S. Ayvaşık, O. Ayan, H. M. Gürsu, and W. Kellerer, "User-based quality of service aware multi-cell radio access network slicing," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 1, pp. 756–768, Mar. 2021.

[12] Y. B. Slimen, J. Balcerzak, A. Pagès, F. Agraz, S. Spadaro, K. Koutsopoulos, M. Al-Bado, T. Truong, P. G. Giardina, and G. Bernini, "Quality of perception prediction in 5G slices for e-health services using user-perceived QoS," *Comput. Commun.*, vol. 178, pp. 1–13, May 2021.

[13] S. Choudhury, S. Das, S. Paul, I. Seskar, and D. Raychaudhuri, "Intelligent agent support for achieving low latency in cloud-native nextg mobile core networks," in *Proc. ACM International Conference on Distributed Computing and Networking (ICDCN)*, Jan. 2023, pp. 12–19.

[14] R. A. Addad, T. Taleb, H. Flinck, M. Bagaa, and D. Dutra, "Network slice mobility in next generation mobile systems: challenges and potential solutions," *IEEE Netw.*, vol. 34, no. 1, pp. 84–93, Jan. 2020.

[15] R. A. Addad, D. L. C. Dutra, T. Taleb, and H. Flinck, "Toward using reinforcement learning for trigger selection in network slice mobility," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2241–2253, July 2021.

[16] W. Wang, Q. Chen, X. He, and L. Tang, "Cooperative anomaly detection with transfer learning-based hidden Markov model in virtualized network slicing," *IEEE Commun. Lett.*, vol. 23, no. 9, pp. 1534–1537, Sept. 2019.

[17] F. Rezazadeh, H. Chergui, L. Christofi, and C. Verikoukis, "Actor-critic-based learning for zero-touch joint resource and energy control in network slicing," in *Proc. IEEE Internation Conference on Communications (ICC)*, June 2021, pp. 1–6.

[18] J. Zhou, W. Zhao, and S. Chen, "Dynamic network slice scaling assisted by prediction in 5G network," *IEEE Access*, vol. 8, pp. 133 700–133 712, July 2020.

[19] Y. Xu, J. Yu, and R. M. Buehrer, "The application of deep reinforcement learning to distributed spectrum access in dynamic heterogeneous environments with partial observations," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4494–4506, May 2020.

[20] A. M. Ibrahim, K.-L. A. Yau, Y.-W. Chong, and C. Wu, "Applications of multi-agent deep reinforcement learning: Models and algorithms," *Appl. Sci.*, vol. 11, no. 22, p. 10870, Nov. 2021.

[21] J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu, and Z. Wang, "Exploration in deep reinforcement learning: From single-agent to multiagent domain," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–21, Jan. 2023.

[22] X. Chen, G. Han, Y. Bi, Z. Yuan, M. K. Marina, Y. Liu, and H. Zhao, "Traffic prediction-assisted federated deep reinforcement learning for service migration in digital twins-enabled MEC networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3212–3229, Aug. 2023.

[23] J. Cai, A. Du, X. Liang, and S. Li, "Prediction-based path planning for safe and efficient human–robot collaboration in construction via deep reinforcement learning," *J. Comput. Civil Eng.*, vol. 37, no. 1, p. 04022046, Oct. 2022.

[24] W. Wang, C. Liang, Q. Chen, L. Tang, H. Yanikomeroglu, and T. Liu, "Distributed online anomaly detection for virtualized network slicing environment," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12 235–12 249, July 2022.

[25] X. Vasilakos, N. Nikaein, D. H. Lorenz, B. Koksal, and N. Ferdosian, “Integrated methodology to cognitive network slice management in virtualized 5G networks,” *arXiv preprint arXiv:2005.04830*, May 2020.

[26] Y. Xiao and M. Krunz, “Dynamic network slicing for scalable fog computing systems with energy harvesting,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2640–2654, Sept. 2018.

[27] H. Halabian, “Distributed resource allocation optimization in 5G virtualized networks,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 627–642, Mar. 2019.

[28] S. Dawaliby, A. Bradai, and Y. Pousset, “Distributed network slicing in large scale IoT based on coalitional multi-game theory,” *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 4, pp. 1567–1580, Dec. 2019.

[29] M. A. Hossain and N. Ansari, “Energy aware latency minimization for network slicing enabled edge computing,” *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 4, pp. 2150–2159, Dec. 2021.

[30] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, “Distributed SDN controller system: A survey on design choice,” *Comput. Netw.*, vol. 121, pp. 100–111, Apr. 2017.

[31] H. Chergui, L. Blanco, L. A. Garrido, K. Ramantas, S. Kukliński, A. Ksentini, and C. Verikoukis, “Zero-touch AI-driven distributed management for energy-efficient 6G massive network slicing,” *IEEE Netw.*, vol. 35, no. 6, pp. 43–49, Nov. 2021.

[32] Z. Zhu and H. Zhao, “A survey of deep RL and IL for autonomous driving policy learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14 043–14 065, Dec. 2021.

[33] J. Zeng, D. Ding, K. Kang, H. Xie, and Q. Yin, “Adaptive DRL-based virtual machine consolidation in energy-efficient cloud data center,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2991–3002, Feb. 2022.

[34] Y. Tu, H. Chen, L. Yan, and X. Zhou, “Task offloading based on LSTM prediction and deep reinforcement learning for efficient edge computing in IoT,” *Future Internet*, vol. 14, no. 2, p. 30, Jan. 2022.

[35] A. Havens, Y. Ouyang, P. Nagarajan, and Y. Fujita, “Learning latent state spaces for planning through reward prediction,” *arXiv preprint arXiv:1912.04201*, 2019.

[36] S. Q. Jalil, S. Chalup, and M. H. Rehmani, “Cognitive radio spectrum sensing and prediction using deep reinforcement learning,” in *Proc. International Joint Conference on Neural Networks (IJCNN)*. IEEE, Sept. 2021, pp. 1–8.

[37] F. Rezazadeh, L. Zanzi, F. Devoti, H. Chergui, X. Costa-Perez, and C. Verikoukis, “On the specialization of FDRL agents for scalable and distributed 6G RAN slicing orchestration,” *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3473–3487, Oct. 2022.

[38] J. Liu, S. Zhang, H. Nishiyama, N. Kato, and J. Guo, “A stochastic geometry analysis of D2D overlaying multi-channel downlink cellular networks,” in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, Aug. 2015, pp. 46–54.

[39] H. Abdah, J. P. Barraca, and R. L. Aguiar, “QoS-aware service continuity in the virtualized edge,” *IEEE Access*, vol. 7, pp. 51 570–51 588, Apr. 2019.

[40] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, and X. Costa-Pérez, “LACO: A latency-driven network slicing orchestration in beyond-5G networks,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 667–682, Oct. 2020.

[41] S. O. Oladejo and O. E. Falowo, “Latency-aware dynamic resource allocation scheme for multi-tier 5G network: A network slicing-multitenancy scenario,” *IEEE Access*, vol. 8, pp. 74 834–74 852, Apr. 2020.

[42] L. Meng, Y. Ren, B. Zhang, J.-Q. Li, H. Sang, and C. Zhang, “MILP modeling and optimization of energy-efficient distributed flexible job shop scheduling problem,” *IEEE Access*, vol. 8, pp. 191 191–191 203, Oct. 2020.

[43] Y. Yu, X. Si, C. Hu, and J. Zhang, “A review of recurrent neural networks: LSTM cells and network architectures,” *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, July 2019.

[44] R. Shashidhar, S. Patilkulkarni, and S. Puneeth, “Combining audio and visual speech recognition using LSTM and deep convolutional neural network,” *INT. J. Inf. Technol.*, vol. 14, no. 7, pp. 3425–3436, Dec. 2022.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[46] Z. Zhang, “Improved Adam optimizer for deep neural networks,” in *Proc. IEEE/ACM International Symposium on Quality of Service (IWQoS)*, Jan. 2018, pp. 1–2.

[47] Y. Fei, Z. Yang, Y. Chen, and Z. Wang, “Exponential bellman equation and improved regret bounds for risk-sensitive reinforcement learning,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 34, pp. 20 436–20 446, Dec. 2021.

[48] Y. Zhou, Q. Ye, and J. Lv, “Communication-efficient federated learning with compensated overlap-fedavg,” *IEEE Trans. Parallel. Distrib. Syst.*, vol. 33, no. 1, pp. 192–205, Jan. 2021.

[49] H. Sak, A. W. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” *arXiv preprint arXiv:1402.1128*, 2014.

[50] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, “Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching,” *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[51] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring network structure, dynamics, and function using networkx,” in *Proc. Python in Science Conference (SciPy)*, Jan. 2008, pp. 11 – 15.

Zhao Ming is currently a Ph.D. student at the Centre for Wireless Communications (CWC), University of Oulu, Oulu, Finland. He received the M.E. degree from Chongqing University (CQU), Chongqing, China, in 2022, and the B.S. degree from Wuhan University of Technology (WHUT), Wuhan, China, in 2018. His current research interests include network slicing, multi-access edge computing, anomaly detection, and AI/ML technologies and their combinations.

Hao Yu received the B.E. and Ph.D. degree in communication engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2015 and 2020. He was also a joint-supervised Ph.D. Student with the Politecnico di Milano, Milano, Italy, and was a Postdoctoral Researcher with the Center of Wireless Communications (CWC), University of Oulu, Finland. He is currently a senior researcher at ICTFICIAL Oy, Espoo, Finland. His research interests include network automation, SDN/NFV, time-sensitive networks, and deterministic networking.

Tarik Taleb (Senior Member, IEEE) received the B.E. degree (with distinction) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Full Professor at Ruhr University Bochum, Germany. He was a Professor with the Center of Wireless Communications (CWC), University of Oulu, Oulu, Finland. He is the founder of ICTFICIAL Oy, and the founder and the Director of the MOSA!C Lab, Espoo, Finland. From October 2014 to December 2021, he was an Associate Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. Prior to that, he was working as a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. Before joining NEC and till March 2009, he worked as an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, in a lab fully funded by KDDI. From 2005 to 2006, he was a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. He has been directly engaged in the development and standardization of the Evolved Packet System as a member of the 3GPP System Architecture Working Group. His current research interests include AI-based network management, architectural enhancements to mobile core networks, network softwarization and slicing, mobile cloud networking, SDN/NFV, software-defined security, and mobile multimedia streaming.