Proactive Anomaly Detection and Resource Allocation for Service Provision in Network Slices

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

***Abstract*—Network slicing enhances the flexibility and configurability of service provision, but also faces challenges such as degraded service quality due to resource competition and dynamic requests of User Equipments (UEs). Thus, more efficient anomaly detection and resource allocation for slice service provision is crucial. Existing methods are hard to apply due to the lack of comprehensive objective analysis, effective framework design, as well as proactive and distributed implementation. To address these challenges, we propose a novel anomaly detection and resource allocation scheme to maximize the long-term system gain based on user request analysis, which jointly optimizes UE service quality, system robustness, and mobile network operator's cost. To this end, we design an offline-online framework to prevent anomalies, proactively detect anomalies, and optimize service provision strategies. Particularly, we embed holistic slice-level information to graph representation with edge attributes for anomaly detection. Moreover, we consider distributed UE-level optimization model training to mitigate interference from abnormal UEs. Extensive evaluations demonstrate our scheme achieves up to 66.47% and 139.42% improvements over existing baselines in anomaly detection and resource allocation, significantly reduces abnormal UEs, and improves detection efficiencies.**

***Index Terms*—Service Provision, Network Slicing, Proactive Anomaly Detection, and Resource Optimization.**

I. Introduction

The rapid advancement of network technologies has catalyzed the evolution of service provision, which refers to the process of delivering computing, storage, and communication resources from infrastructures to users, to support desired services and applications with appropriate resource allocation and latency performance [2], [3]. Achieving efficient service provision necessitates flexible resource scheduling, autonomous infrastructure orchestration, and adaptive responsiveness to customized user requests, especially when facing the complex future 6G networks [4]–[6]. Thus, it is essential to introduce flexible and configurable network and resource management. In this context, by integrating network slicing enabled by Network Function Virtualization (NFV) and Software Defined Networking (SDN) into service provision, users can acquire resources from configurable and customizable network slices tailored to their demands. We can therefore achieve more flexible resource scheduling and network configuration, thereby enhancing the capability for better service provision.

To support the diverse, strict, and sometimes conflicting requests of users, slice service provision also faces challenges. On one hand, highly dense and complex deployment topologies of physical Base Stations (BSs) and Virtual Network Functions (VNFs), and stricter competition for resources among BSs/VNFs may lead to higher probabilities of Service Level Agreement (SLA) violations [5]. These can be caused by, for instance, the unavailability of computation/storage resources in facilities or limited network resources, and thus affect the Quality of Service/Experience (QoS/QoE) to users. On the other hand, considering the dynamics of users' requests, their mobility patterns, and the relatively lagging adjustment of networks, the degradation of QoS/QoE will be easily triggered even within the SLA requirements, which require Mobile Network Operators (MNOs) to adjust the re-deployments of VNFs of running slices. Under these circumstances, slice anomaly detection, which aims to identify whether users' requests are fulfilled during the service provision process, becomes critical for locating existing and potential threats, ensuring SLA compliance, and mitigating service quality degradation. Additionally, more intelligent resource allocation will be needed for better service provision experience.

Several researchers have investigated slice anomaly detection and resource allocation in recent years. Many of them utilized machine learning/deep learning-based schemes to detect the threats of slices by facility/link metric analysis [7]–[11]. On the other hand, lots of researchers considered the network architecture and leveraged technologies such as Graph Convolutional Networks (GCN) to capture the correlation between BSs and VNFs from the global perspective [12], [13]. In serving the requests from User Equipments (UEs), several research employed traditional mathematical models or predefined rules to optimize resource allocation, focusing on metrics such as energy consumption and latency [14], [15], while others utilized learning-based methods to achieve resource prediction and scheduling, with dynamic QoS requirements considered [4], [16]–[19].

Despite recent advances, several challenges still need to be effectively addressed. **First**, most existing studies focused on anomaly detection or resource allocation based on system-level metrics rather than actual user requests, and treated these tasks as independent processes, which is unreasonable. The anomalies captured by analyzing indicators such as access

Parts of this work were presented at the IEEE International Conference on Communications (IEEE ICC), USA, June 2024 [1]. This work has made a significant extension to system model, scheme design, and evaluation results.

The research work is supported in part by the Federal Ministry of Research, Technology, and Space (BMFTR), Germany, through the Project 6GEM+ under Grant 16KIS2411; and in part by the European Union's Horizon Europe research and innovation programme under the 6G-Path project (Grant No. 101139172).

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

Tarik Taleb is with the Faculty of Electrical Engineering and Information Technology (EEIT), Ruhr University Bochum, Bochum, Germany (email: tarik.taleb@rub.de).

latency and remaining resources of VNFs are not always caused by dynamic user requests. Instead, they can also arise from various system-level and environmental factors, such as network's routing policy and software misconfigurations [20], [21], load-balancing mechanisms [22], hardware degradation [23], and congestions in the underlying infrastructure. Moreover, anomaly detection and resource optimization should form an integrated process, as detecting anomalies serves as a prerequisite to confirm slice health, ensuring that subsequent resource optimization is applied to normal slices where efforts to further enhance QoE are meaningful. Furthermore, resource distribution and service latency should be synergistically managed, since effective service delivery requires maintaining latency within acceptable thresholds while ensuring sufficient resource provisioning for UEs. **Second**, effective anomaly detection requires a holistic representation of network slices to capture the full spectrum of potential anomalies, including VNFs' resource availability, network topology, and provision relationships between VNFs and UEs. However, current research either lacks an effective representation of the networks or ignored graph's edge information that includes service latency and provision relationships. Additionally, when training resource optimization models, from the slice perspective, this becomes a complex multi-VNF-to-multi-UE decision-making problem, where the high-dimensional action space significantly increases the training instability. On the other hand, for global model training (centralized or federated learning-based paradigms), UEs in abnormal slices may yield sparse DRL training samples or delayed model updates due to anomalies, which could contaminate the global model. Thus, distributed and isolated UE-level resource optimization is necessary. **At last**, the interplay between UE QoE, system robustness, and MNO costs typically manifests as a trilemma, posing a fundamental challenge to consistent service delivery, thus, it remains valuable to address the intricate balance required among these aspects. In addition, in real-world deployments, it is necessary to balance detection efficiency and algorithm performance, especially when handling fine-grained user request variations. Even rule-based methods remain widely adopted, they always require careful design, continuous refinement, and dynamic configuration. To better exploit the superior performance of learning-based schemes, it will be beneficial to consider distributed deployment, online inference, and proactive analysis for framework design.

To address these challenges, we explore proactive anomaly detection and resource allocation in network slices, leveraging user request data that reveals real service demands and jointly optimizing latency and resource efficiency. We model the process for service provision and formulate the problem as maximizing the long-term system gain. This considers the joint impact of UE QoE, system robustness, and MNO costs, where a key marketing indicator reflecting the QoE of the customer, is introduced. To solve this problem, we analyze the upper bound of the key indicator and consider improving the system gain from four aspects. To this end, we design an offline-online framework that segregates the UEs' requests to prevent the anomalies, predicts the UE requests to proactively detect the anomalies, and utilizes Deep Reinforcement Learning (DRL) to optimize the service provision strategies. Particularly, the framework employs slice-level anomaly detection by embedding the holistic slice information to graph samples to effectively capture the representation. Here, considering the dynamic graph structure and importance of edge attributes, we integrate a Graph Isomorphism Network with Edge (GINE)-based convolution layer in our GCN model. Meanwhile, we perform UE-level service provision model training to optimize resource allocation strategies, where DRL models are distributedly trained on UEs to mitigate interference stemming from UEs with poor experience. Subsequently, the refined models are deployed in a distributed manner for online inference to achieve efficient and proactive anomaly detection and resource allocation. As a result, extensive evaluations demonstrate the proposed scheme outperforms existing baselines in anomaly detection and resource optimization, and can significantly reduce abnormal UEs and enhance detection efficiency. To summarize, the contributions of this paper are as follows:

- To our best knowledge, this work represents the first attempt to jointly tackle proactive anomaly detection and resource allocation for network slice service provision based on user request analysis, considering both service latency and resource provided.
- We model the service provision process and formulate the problem as maximizing the long-term system gain, which jointly considers UE QoE, system robustness, and MNO costs. Then, we define a marketing indicator reflecting the QoE, analyze its upper bound, and design an offline-online framework to solve the problem.
- To extract slice-level representation with edge attributes, we integrate the GINE convolution layer into the GCN model for anomaly detection. Moreover, we implement distributed UE-level DRL model training to mitigate the interference stemming from abnormal UEs.
- Extensive evaluation results demonstrate the proposed scheme's superior performance in improving system gain, reducing abnormal UEs, and enhancing efficiency.

The remainder of this paper is organized as follows. Section II discusses the related work. Section III introduces the system model and formulates the problem. In Section IV, we propose a two-phase framework. Section V presents the evaluation results and finally, Section VI concludes this paper.

II. Related Work

In this section, we elaborate on recent research on slice anomaly detection and resource allocation as two-fold.

A. Slice Anomaly Detection

The authors in [7] proposed a distributed online anomaly detection algorithm based on a decentralized one-class support vector machine and further extended it to [10], wherein they developed an algorithm that analyzes real-time measurements of VNFs mapped to BSs in a distributed manner. Additionally, for physical link anomaly detection, they introduced a canonical correlation analysis-based algorithm that examines the correlation of measurements between neighboring VNFs mapped to both ends of a physical link. However, these studies

ignored the prediction of potential anomalies in network slices; when anomalies occur, it is generally hard to adjust the resources in time due to resource competition among BSs/VNFs, which will unavoidably lead to service degradation of UEs. Moreover, Zhou *et al.* proposed a machine learning-driven automatic network slicing anomaly detection framework in the Internet of Things (IoT) scenarios [9]. The framework incorporates deep learning-based slice state prediction, unsupervised learning-based anomaly detection, and reinforcement learning-based slice scaling. They also introduced a robustness-oriented resource preservation scheme to balance network robustness and resource efficiency.

Additionally, considering the resource of slices, Abbas *et al.* proposed an intent-based networking mechanism for efficient orchestration and management of end-to-end network slicing in 5G networks [11]. They integrated a network data analytics function into the intent-based networking platform and implemented a novel hybrid stacking ensemble learning model for the prediction of network resource utilization and a hybrid ensemble learning model for attack detection. Though these studies considered the prediction of slices/UEs, they primarily focused on system-level indicator analysis. However, anomalies observed in system metrics do not necessarily stem from user-driven resource contention or request dynamics, and root cause analysis is needed for service provision anomaly detection, which is always challenging [24]. Additionally, the researchers only considered the resources or the latency indicator for anomaly detection from the BSs/VNFs, but overlooked the relationships among them. The latter is essential for determining slice anomalies from a holistic view [25], [26].

To cope with these issues, in recent years, the authors in [12] and [13] proposed to utilize graph-based methods to extract the abnormal features from the slice perspective. The researchers proposed a graph-based interpretable anomaly detection framework that utilized GCN and the correlations of KPIs to detect the anomalies and forecasted individual time series using Recurrent Neural Network (RNN) [12]. Moreover, in [13], the researchers proposed a knowledge-driven cross-domain collaborative anomaly detection scheme, which consists of a knowledge-driven sub-slice anomaly detection model and uses hierarchical federated learning to achieve inter-slice and intra-slice collaborative anomaly detection. Even these studies adopted graph-based methods to extract the features of slices/VNFs efficiently, they did not consider the service provision relationships among VNFs and UEs and the dynamically changing requests of UEs, and the optimization of normal slices after anomaly detection is always ignored.

B. Resource Optimization and Slice Management

Masoudi *et al.* proposed an energy-optimal end-to-end network slicing framework for cloud-based 5G networks [14]. They developed a slice-based resource allocation algorithm that jointly optimizes bandwidth and processing resources across radio access network, fronthaul, and cloud and that is to minimize the total network energy consumption while meeting slice-specific delay requirements. Additionally, Shu *et al.* proposed a novel QoS framework in 5G and beyond network slicing based on SDN and NFV [15]. They evaluated an SDN controller's performance in a Mininet-based simulation environment, demonstrated the framework's ability to allocate network resources for different types of network slices, and proved reliable end-to-end QoS according to preconfigured requirements. These studies utilized rule-based or optimization-based schemes to solve the formulated problem.

Vasilakos *et al.* proposed an integrated methodology for cognitive network and slice management in virtualized 5G networks using machine learning [19]. The researchers demonstrated the methodology using a practical assessment in two 5G use cases. Researchers in [16] proposed a QoS-guaranteed network slicing orchestration scheme for IoV with two-timescale resource allocation using Long Short-term Memory (LSTM) for long-term resource prediction and Deep Deterministic Policy Gradient (DDPG) for short-term resource scheduling. The LSTM scheme predicts the average resource demands for dedicated allocation, while DDPG performs fine-grained online resource adjustment to meet dynamic QoS requirements. To assist the resource optimization in network slices with predictions, Rezazadeh *et al.* proposed a knowledge-based network slicing orchestration framework for beyond 5G networks [18]. They developed a twin-delayed Soft Actor-Critic (SAC) algorithm to minimize energy consumption and VNF instantiation costs in network slices. These studies adopted centralized decision-making deployments and may suffer from efficiency issues in environments with dynamic requests and extreme latency requirements. Meanwhile, the authors in [17] introduced a statistical federated learning-based analytic engine for zero-touch 6G massive network slicing, which performs slice-level resource prediction by offline learning while respecting preset long-term SLAs. In our previous work [4], we proposed a prediction-based Network Slice Mobility (NSM) framework, which utilized the prediction of user requests to assist decision-making in DRL and integrated it with the federated learning paradigm. However, none of these studies have addressed joint anomaly detection and resource optimization. In contrast to these approaches, our work integrates anomaly prevention, proactive anomaly detection, and normal slices' resource allocation, directly driven by user request data reflecting real service demands, with latency and resource efficiency jointly considered.

III. System Model and Problem Formulation

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

A. System Model

The considered network architecture for slice anomaly detection and resource allocation for service provision is illustrated in the left part of Fig. 1, which consists of three layers from the bottom to the top. In the device layer, multiple UEs like mobile devices, laptops, and smart vehicles are geographically distributed. The UEs dynamically move to different areas over time slots and request resources (like CPU, RAM, or disk resources) with specific latency preferences. We denote the set of UEs as $\mathcal{U}$, the set of resource types as $\mathcal{F}$,

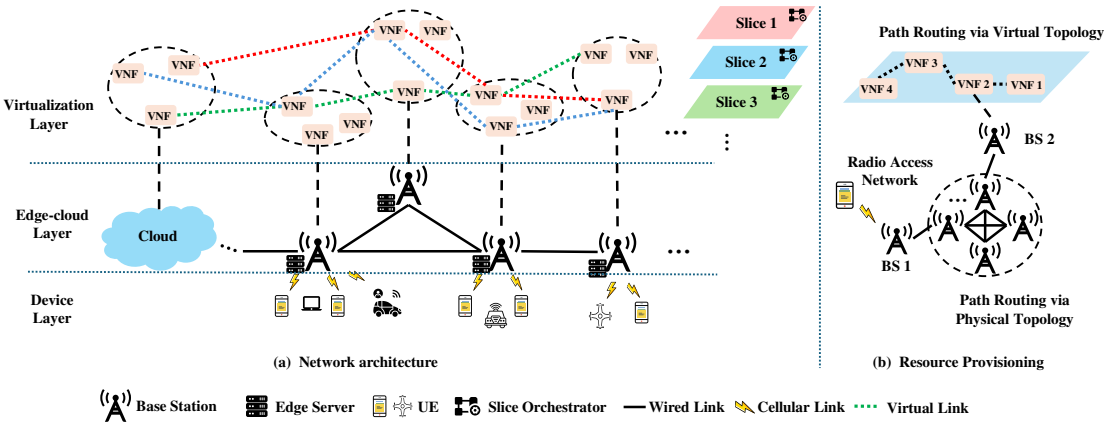


Fig. 1. The considered network architecture for service provision in network slices.

the geographical coordinate of UE $u \in \mathcal{U}$ at time slot t as $(\mathrm{X}_u^t, \mathrm{Y}_u^t)$. Moreover, the requested amount and latency from u of resource f at t are respectively denoted as $R_{u,f}^t$ and $l_{u,f}^t, \forall u \in \mathcal{U}, \forall f \in \mathcal{F}, \forall t$. The UEs are connected to the BSs in the edge-cloud layer by cellular links, each of which is equipped with an Edge Server (ES) with limited resources. Throughout this paper, BS and ES are used interchangeably. These BSs are connected to their neighbors through high-speed optical links and to the remote cloud center through backhaul links. We denote the group of ESs in the system as $\mathcal{M}$, the total amount of resources f of ES m as $R_{m,f}$, the coordinates of ES m as $(\mathrm{X}_m, \mathrm{Y}_m), \forall m \in \mathcal{M}$, respectively, and assume the cloud is built with multiple servers and has elastic resources.

TABLE I
SUMMARY OF NOTATIONS AND DESCRIPTIONS

Notation	Description
$\mathcal{U}$, $\mathcal{M}$, $\mathcal{S}$	Set of UEs, ESs, and slices in the system
$(\mathrm{X}_m, \mathrm{Y}_m)$	Coordinate of ES m
$(\mathrm{X}_u^t, \mathrm{Y}_u^t)$	Geographical coordinate of UE u at t
$R_{u,f}^t$	Requested amount of resource type f from UE u at t
$l_{u,f}^t$	Requested latency for resource type f from UE u at t
$R_{m,f}$	Total amount of resource type f of ES m
$\mathcal{V}_s$	Set of VNFs of slice s
P_v^t	Physical host of VNF v at time slot t
$\kappa_{v,v'}$	Connection relationship of v and v' in slice s
$R_{v,f}^t$	Allocated amount of resource type f for VNF v at t
$\mathcal{U}_s$	UEs served by slice s
$R_{u,v,f}^t$	Allocated resources of user u from v for resource type f
$l_{u,v}^t$	Access latency from u to v
M_u^t	Nearest ES to user u at time slot t
$V_{u,s}^t$	Nearest VNF of slice s to user u at time slot t
η^W/η^O	Wireless latency for u / Optical latency between ESs
η_s^V/η_s^C	Latency between VNFs / Latency to access cloud for s
$\Omega(\cdot)$	Set of links among ESs/VNFs on the shortest path
Δ_s^t	Anomaly state of slice s at time slot t
α_f	User satisfaction of a resource unit for resource type f
δ_f	User penalty when anomaly occurs for resource type f

When UEs request resources from the system, each BS collects the serving UEs' request information, and the ESs and the cloud initializes multiple VNFs. Meanwhile, to support customized user requests, multiple virtual links are built on top of the physical optical links, based on which the VNFs can connect to their neighbors to form slices with different SLAs, as shown in the virtualization layer. We consider each slice to have a slice orchestrator deployed at one of the hosted ESs to achieve the management of VNFs. The UEs are then allocated to the slices regarding to the request characteristics, e.g., UEs that request large amounts of resources with high latency tolerance will be allocated to slices with more resources but ordinary QoS and vice versa. We denote the set of slices as $\mathcal{S}$, the VNFs of slice s as $\mathcal{V}_s$, and the allocated amount of resource type f for VNF v at t as $R_{v,f}^t, \forall v \in \mathcal{V}, \forall f \in \mathcal{F}, \forall t$. VNF v can be hosted by the ESs or the remote cloud. We denote the physical host of VNF v at time slot t as P_v^t. For VNFs $v, v' \in \mathcal{V}_s$, we use $\kappa_{v,v'} \in \{0,1\}$ to indicate their connection relationship, whereby "1" means they are directly connected and "0" means the opposite. Moreover, we denote the UEs served by slice s as $\mathcal{U}_s$ and assume each UE to be served by only one slice and thus derive $\mathcal{U}_s \subseteq \mathcal{U}$ and $\mathcal{U}_s \cap \mathcal{U}_{s'} = \emptyset, \forall s, s' \in \mathcal{S}$. In this paper, we neglect the details of slice initialization.

Each UE can access resources from multiple VNFs, as shown in the right part of Fig. 1. When a UE u accesses resources from the VNFs in slice s, u should first connect to the nearest BS (BS 1) as the Access Point (AP) through wireless links to achieve better wireless communication latency. After that, BS 1 routes the request to the BS hosting the nearest VNF of slice s to UE u (VNF 2 hosted by BS 2) via physical optical links. Then VNF 2 forwards the request to the VNFs that provide u with resources by the virtual links of s. For UE u, we denote the allocated amount of resource f from $v \in \mathcal{V}_s$ at t as $R_{u,v,f}^t$ and the access latency from u to v as $l_{u,v}^t$, intuitively, if $l_{u,v}^t$ cannot satisfy the request latency from the user, i.e., $l_{u,v}^t > l_{u,f}^t$, $R_{u,v,f}^t$ is set to 0.

Denoting the nearest ES to user u at t as M_u^t and the nearest VNF of slice s to u at t as $V_{u,s}^t$, M_u^t can be obtained as $M_u^t = \arg\min_{m \in \mathcal{M}}\{(\mathrm{X}_m - \mathrm{X}_u^t)^2 + (\mathrm{Y}_m - \mathrm{Y}_u^t)^2\}, \forall u \in \mathcal{U}_s, \forall s \in \mathcal{S}$, and $V_{u,s}^t$ can be obtained as $V_{u,s}^t = \arg\min_{v \in \mathcal{V}_s}\{(\mathrm{X}_{\mathrm{P}_v^t} - \mathrm{X}_u^t)^2 + (\mathrm{Y}_{\mathrm{P}_v^t} - \mathrm{Y}_u^t)^2\}, \forall u \in \mathcal{U}_s, \forall s \in \mathcal{S}$. To support diverse UE request latency, different slices typically exhibit distinct virtual link latency performances, which can be realized by allocating better network channels or adjusting resource allocation strategies. In this way, we can flexibly

control the queuing and processing behaviors of virtual links to support latency-sensitive services based on the physical links [27]–[29]. We denote the average wireless latency for user u to access the nearest ES as η^W, the average optical latency between two consecutive ESs as η^O, the virtual link latency between consecutive VNFs in slice s as $\eta_s^V, \forall s \in \mathcal{S}$. Generally, $V_{u,s}^t$ should be near to the UE u and we can regard $\mathrm{P}_{V_{u,s}^t}^t$ as an ES. As accessing the remote cloud generally has much higher latency, we denote the latency for the VNFs deployed in ESs to the cloud as η_s^C. Thus, $l_{u,v}^t$ can be calculated by

$$\begin{aligned} l_{u,v}^t = \eta^W + \eta^O |\Omega(M_u^t, \mathrm{P}_{V_{u,s}^t}^t)| + \Pi_{(\mathrm{P}_v^t \in \mathcal{M})}\ \eta_s^V |\Omega(V_{u,s}^t, v)| \\ + (1 - \Pi_{(\mathrm{P}_v^t \in \mathcal{M})})\ \eta_s^C, \forall u \in \mathcal{U}_s, \forall v \in \mathcal{V}_s, \forall s, \end{aligned} \tag{1}$$

where $\Pi_{(\cdot)}$ is an indicator variable that equals to 1 when "·" holds otherwise equals to 0; $\Omega(\cdot)$ denotes the links among ESs/VNFs on the shortest path of the physical/virtual network topology; $|\cdot|$ denotes the cardinality of a set.

B. Problem Formulation

Generally, the UEs' QoE will be severely affected by the running status of slices, while anomalies will inevitably lead to slice migration and possible service degradation [30]. We denote $\Delta_s^t \in \{0, 1\}$ to indicate the running status of a slice s at t, where "1" means that s experiences anomalies and "0" means the opposite. When serving UEs' requests cannot be fulfilled by the VNFs of the slice, i.e., the requested resources cannot be provided by the slice with preferred latency, the slice anomaly will occur, and the UEs not fulfilled are considered as abnormal UEs. Expecting that all UEs' service requests are satisfied, we set $\Delta_s^t = 0$ when $\sum_{v \in \mathcal{V}_s} R_{u,v,f}^t \geq R_{u,f}^t, \forall u \in \mathcal{U}_s$ holds and otherwise $\Delta_s^t = 1$.

To measure the QoE of UEs, we introduce the Net Promoter Score (NPS) that is a key marketing indicator reflecting the experience of customers [31], [32]. Let NPS_u^t denote the NPS of user u at time slot t, which can be calculated as

$$\begin{aligned} \mathrm{NPS}_u^t = \sum_{f \in \mathcal{F}} \{(1 - \Delta_s^t)\ \alpha_f \min(1, \frac{R_{u,f}^t}{\sum\limits_{v \in \mathcal{V}_s} R_{u,v,f}^t}) \sum_{v \in \mathcal{V}_s} \frac{R_{u,v,f}^t}{l_{u,v}^t} \\ + \Delta_s^t\ \ \delta_f R_{u,f}^t\}, \forall u \in \mathcal{U}_s. \end{aligned} \tag{2}$$

Here, the first part indicates when a slice s runs normally ($\Delta_s^t = 0$), $\alpha_f > 0$ denotes the user satisfaction rate for resource type f. The QoE is related to the allocated resources from VNFs and the corresponding latencies of resources provided, where lower latencies and more resources will contribute to higher user satisfaction [33]. Particularly, $\min(1, R_{u,f}^t / \sum_{v \in \mathcal{V}_s} R_{u,v,f}^t)$ means that when the provided resources exceed the requests, the QoE of UEs will have a discount based on the actual request. This is because QoE saturates once the demand is satisfied, and the discount keeps NPS focused on efficient fulfillment instead of excessive allocation. In the second part, if an anomaly occurs in the slice ($\Delta_s^t = 1$), the UE will undergo a degradation based on its requested resources (more degradation for higher resources), we use $\delta_f < 0$ to denote the degradation rate. As a result, the NPS of slice s is expressed as $\mathrm{NPS}_s^t = \sum_{u \in \mathcal{U}_s} \mathrm{NPS}_u^t$.

Moreover, the remaining resources of VNFs are critical for slices to get "free" of anomalies, since more remaining resources offer the system more flexibility for resource scheduling within resource constraints. We denote the Remaining Resource Level (RRL) for resource f of $v \in \mathcal{V}_s$ at t as $\mathrm{RRL}_{v,f}^t$, which reflects the remaining resources after allocation to UEs, and can be calculated by $\mathrm{RRL}_{v,f}^t = 1 - (\sum_{u \in \mathcal{U}_s} R_{u,v,f}^t) / R_{v,f}^t$, where $\sum_{u \in \mathcal{U}_s} R_{u,v,f}^t$ indicates the sum of allocated resources to UEs. Thus, the RRL of slice s at t can be measured as the sum of all VNFs of slice s, calculated as $\mathrm{RRL}_s^t = \sum_{f \in \mathcal{F}} \sum_{v \in \mathcal{V}_s} \mathrm{RRL}_{v,f}^t$, which reflects the total remaining resources of slice s.

Additionally, we consider the cost of MNOs to maintain the service to UEs. Denoting the MNO cost of slice s at t as C_s^t, which is related to the allocated resources and the latency configurations of the slice, i.e., more resources and lower latency will lead to higher cost. Thus, C_s^t can be calculated as $\mathrm{C}_s^t = \sum_{u \in \mathcal{U}_s} \sum_{v \in \mathcal{V}_s} \sum_{f \in \mathcal{F}} R_{u,v,f}^t / \eta_s^V$. We aim to jointly optimize system NPS and RRL, while simultaneously reducing MNO cost. Thus, the problem can be formulated as maximizing the long-term system gain expressed as

$$\max_{R_{u,v,f}^t} \quad \mathbf{G} = \sum\nolimits_t \sum\nolimits_{s \in \mathcal{S}} (\mathrm{NPS}_s^t + \mathrm{RRL}_s^t - \mathrm{C}_s^t), \tag{3a}$$

$$\mathbf{C1}: \sum_{s \in \mathcal{S}} \sum_{v \in \mathcal{V}_s} \Pi_{(\mathrm{P}_v^t = m)} R_{v,f}^t \leq R_{m,f}, \forall m \in \mathcal{M}, \forall f, \forall t, \tag{3b}$$

$$\mathbf{C2}: \sum\nolimits_{u \in \mathcal{U}_s} R_{u,v,f}^t \leq R_{v,f}^t, \forall v \in \mathcal{V}_s, \forall f, \forall t, \tag{3c}$$

$$\mathbf{C3}: \ \mathcal{U}_s \cap \mathcal{U}_{s'} = \emptyset, \forall s, s' \in \mathcal{S}, \tag{3d}$$

$$\mathbf{C4}: \ \kappa_{v,v'} \in \{0, 1\}, \forall v \in \mathcal{V}_s, \forall s \in \mathcal{S}, \tag{3e}$$

$$\mathbf{C5}: \ \Delta_s^t \in \{0, 1\}, \forall s \in \mathcal{S}, \forall t. \tag{3f}$$

Here, **C1** indicates that the total resources of hosted VNFs of ES m cannot exceed the resource of the ES; **C2** ensures that the allocated resources for all serving UEs of VNF v cannot be more than the total resources of the VNF; **C3** ensures each UE can only request one slice for service provision; **C4** and **C5** ensure the ranges of values for variables.

From the formulated problem, we can see when $\Delta_s^t = 0$, the NPS can be improved by providing resources to UEs from the VNFs with lower latency. As increasing $R_{u,v,f}^t$ improves the NPS but also reduces the RRL and leads to higher MNO cost; to see how the provided resources will affect the system gain, we introduce the following theorem:

Theorem 1. *When slices are running normally, the system NPS has an upper bound related to $R_{u,f}^t$.*

Proof. The system NPS can be calculated by $\sum_t \sum_{s \in \mathcal{S}} \mathrm{NPS}_s^t$, when $\Delta_s^t = 0$, it can be rewritten as

$$\sum_t \sum_{s \in \mathcal{S}} \sum_{f \in \mathcal{F}} \sum_{u \in \mathcal{U}_s} \{\alpha_f \min(1, \frac{R_{u,f}^t}{\sum\limits_{v \in \mathcal{V}_s} R_{u,v,f}^t}) \sum_{v \in \mathcal{V}_s} \frac{R_{u,v,f}^t}{l_{u,v}^t}\}. \tag{4}$$

As the slices are running normally, we have $\sum_{v \in \mathcal{V}_s} R_{u,v,f}^t \geq R_{u,f}^t$, indicating that all resource requests

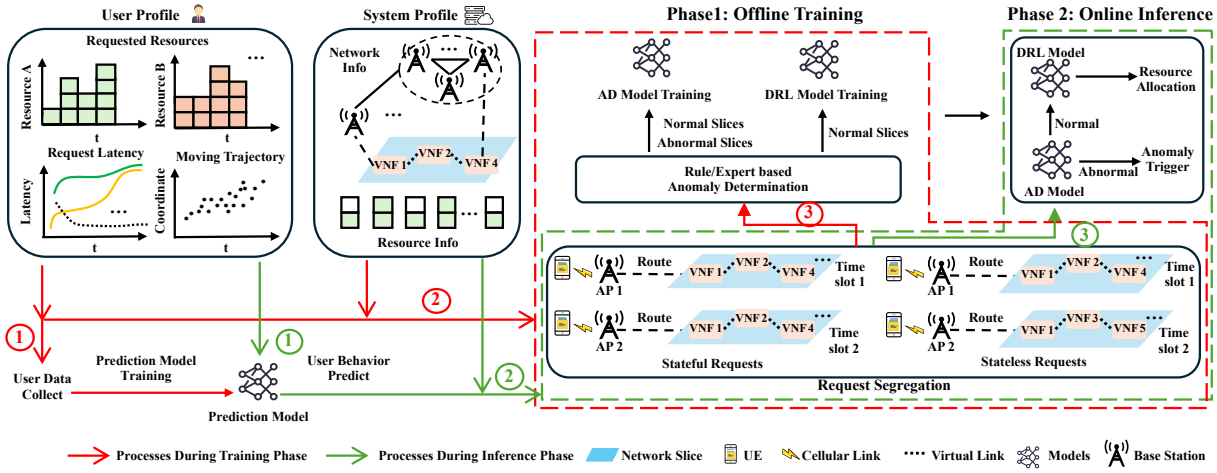


Fig. 2. The considered offline training and online inference framework.

of UEs should be satisfied, according to the Cauchy–Schwarz inequality, (4) can be derived as

$$\begin{aligned}
&\sum_{t}\sum_{s\in\mathcal{S}}\sum_{f\in\mathcal{F}}\sum_{u\in\mathcal{U}_s}\{\alpha_f\frac{R^t_{u,f}}{\sum\limits_{v\in\mathcal{V}_s}R^t_{u,v,f}}\sum_{v\in\mathcal{V}_s}\frac{R^t_{u,v,f}}{l^t_{u,v}}\}\\
&\leq\sum_{t}\sum_{s\in\mathcal{S}}\sum_{f\in\mathcal{F}}\sum_{u\in\mathcal{U}_s}\{\alpha_f\frac{R^t_{u,f}}{\sum\limits_{v\in\mathcal{V}_s}R^t_{u,v,f}}\sum_{v\in\mathcal{V}_s}R^t_{u,v,f}\sum_{v\in\mathcal{V}_s}\frac{1}{l^t_{u,v}}\}\\
&=\sum\nolimits_{t}\sum\nolimits_{s\in\mathcal{S}}\sum\nolimits_{f\in\mathcal{F}}\sum\nolimits_{u\in\mathcal{U}_s}\{\alpha_f R^t_{u,f}\sum\nolimits_{v\in\mathcal{V}_s}\frac{1}{l^t_{u,v}}\},
\end{aligned}\tag{5}$$

which is related to $R^t_{u,f}$, this completes the proof. □

As the NPS's upper bound does not increase with more provided resources from VNFs, increasing $R^t_{u,v,f}$ when the NPS is close to the upper bound will reduce the RRL and increase MNO cost. This cannot further optimize the NPS, leading to a lower system gain. Based on the above discussions, we propose to improve the system gain $\mathbf{G}$ from the following aspects: *1)* Reduce the anomaly occurrence of slices by reducing abnormal UEs caused by unfulfilled requests, this is because when $\Delta^t_s = 1$, the NPS derives a negative value due to δ_f, and continuously introduces cost for MNOs; *2)* Locate the anomalies in advance to reduce the downtime of abnormal slices, as abnormal slices need to be adjusted to cope with the anomalies by methods like NSM [4], [34], which is a time-consuming process, thus, it is necessary to predict the anomalies to eliminate the downtime of slices; *3)* Optimize service provision strategies (i.e., better resource to latency ratio) for UEs to achieve higher NPS; *4)* Avoid providing extra resources for UEs to reduce the MNO cost and improve the RRL to cope with the potential anomalies.

IV. Overall Framework Design

In this section, we present the overall framework that consists of two phases to solve the problem, as shown in Fig. 2. During the training phase, as shown by the red lines, UEs' requested resources, latencies, and moving trajectories (defined as **UE Profile**) are continuously collected as historical data to train the prediction model. Moreover, the physical/virtual topologies of the network and the resources of BSs/VNFs (defined as **System Profile**) are combined with the UE profile and sent to the request segregation module, which is designed to prevent abnormal UEs. The running status of slices is then determined and labeled by an anomaly determination module, which is based on a hybrid technique with defined rules and experience. All slice profile data with labels are stored to train the anomaly detection model, and the normal slices' profile data are then utilized to train the DRL-based service provision model to optimize the resource allocation strategies. In the inference phase, shown by the green lines, the system timely collects UEs' profile data and predicts their future requests, which is then segregated and combined with the system profile data. Afterward, the well-trained anomaly detection model infers the potential anomaly slices and the DRL model decides on the service provision strategies for UEs in normal slices. In the following subsection, we introduce the request segregation module, and the details for labeling the slices in the anomaly determination module will be presented in Sect. V-A.

A. Stateful and Stateless Request Segregation

For UEs with stateful requests, each subsequent request depends on the previous context or status, thus requiring sticky sessions [35], [36]. In other words, all stateful requests from the same UE must always be routed to the same target VNFs; otherwise, session information would be lost. In contrast, stateless requests are independent of prior state and can be processed by any available target VNFs, as long as latency requirements are met. Conventional service provision often presumes UEs maintain connections with prior VNFs for uninterrupted service, treating all requests as stateful [4], [34], as exemplified by file synchronization and streaming media which require sustained bandwidth from caching VNFs. However, in scenarios such as content delivery networks, object storage services [37], and IoT measurement reporting, UE requests can be stateless and the current requests are independent of previous states. Inspired by these characteristics, we consider segregating UEs' requests to reduce the number

of abnormal UEs caused by resource availabilities in restricted candidate serving VNFs. To this end, we denote the request type of UE u at time slot t as $\Upsilon_u^t \in \{0,1\}$, where "0" means stateless requests and "1" means stateful. We illustrate the service provision process in the request segregation module of Fig. 2. Specifically, for UE u with stateful requests, to access the resources from a corresponding slice, u connects to the nearest BS as the AP and then is routed to the VNFs that meet the latency preferences. When the UE moves to a new area in the next time slot, it connects to the current nearest BS as the new AP and then is routed to the associated VNFs. For the UE u' with stateless requests, different from u, it can be served any VNFs that meet the requested latency, indicating that it can switch VNFs freely without being associated with specific ones. Based on the request segregation, we have the following theorem that the abnormal UEs caused by resource unavailability can be alleviated partially:

Theorem 2. *Given identical request and trajectory conditions, UE u' with stateless requests has a higher probability of successful request fulfillment than UE u with stateful requests.*

Proof. We consider the requested resources and latencies of the two UEs as $R_{u,f}^t, l_{u,f}^t$, $R_{u',f}^t, l_{u',f}^t$, the potential VNFs that can serve u can be denoted as $\mathcal{V}_{P,u}$, which should be chosen from the previous serving VNFs, obtained by $\mathcal{V}_{P,u} = \{v \in \mathcal{V}_s | (l_{u,v}^t \leq l_{u,f}^t) \wedge (R_{u,v,f}^{t-1} > 0), \forall f \in \mathcal{F}\}$. For UE u', $\mathcal{V}_{P,u'}$ can be chosen from any VNFs that meet the latency requirements, w.r.t., $\mathcal{V}_{P,u'} = \{v \in \mathcal{V}_s | l_{u',v}^t \leq l_{u',f}^t, \forall f \in \mathcal{F}\}$. As $l_{u,f}^t = l_{u',f}^t$ and $l_{u,v}^t = l_{u',v}^t, \forall v \in \mathcal{V}_s$ due to the identical request and trajectory, we can derive $\mathcal{V}_{P,u} \subseteq \mathcal{V}_{P,u'}$. Thus, if the request of u can be fulfilled by $\mathcal{V}_{P,u}$, the request of u' can be fulfilled by $\mathcal{V}_{P,u'}$ while the converse doesn't always hold, this completes the proof. □

B. Prediction Model Training

To learn the characteristics of UEs' requests and moving trajectories, we collect the historical UE profile data for several time slots, assuming that all the UEs' behavior data over time slots are Identically Independently Distributed (IID), and the cloud uniformly trains a global prediction model. Here, the LSTM is chosen for prediction model training, which introduces the concept of gates and adds internal cell loops to solve the long-distance dependency issue [4]. Specifically, for each user $u \in \mathcal{U}$, we collect φ time slots' historical UE profile data and split it into training and testing samples, the overall historical data of UE u can be expressed as $\mathcal{K}_u^{his} = \{\mathrm{X}_u^1, \mathrm{Y}_u^1, \{R_{u,f}^1, l_{u,f}^1\}_{f\in\mathcal{F}}, \ldots, \mathrm{X}_u^\varphi, \mathrm{Y}_u^\varphi, \{R_{u,f}^\varphi, l_{u,f}^\varphi\}_{f\in\mathcal{F}}\}$. Denoting that each training sample consists of χ time slots' UE data, the $\varphi - \chi$ training samples can be written as $x_{u,1} = \{\mathrm{X}_u^1, \mathrm{Y}_u^1, \{R_{u,f}^1, l_{u,f}^1\}_{f\in\mathcal{F}}, \ldots, \mathrm{X}_u^\chi, \mathrm{Y}_u^\chi, \{R_{u,f}^\chi, l_{u,f}^\chi\}_{f\in\mathcal{F}}\}, \ldots, x_{u,\varphi-\chi} = \{\mathrm{X}_u^{\varphi-\chi}, \mathrm{Y}_u^{\varphi-\chi}, \{R_{u,f}^{\varphi-\chi}, l_{u,f}^{\varphi-\chi}\}_{f\in\mathcal{F}}, \ldots, \mathrm{X}_u^{\varphi-1}, \mathrm{Y}_u^{\varphi-1}, \{R_{u,f}^{\varphi-1}, l_{u,f}^{\varphi-1}\}_{f\in\mathcal{F}}\}$. Additionally, the corresponding $\varphi - \chi$ testing samples can then be expressed as $y_{u,1} = \{\mathrm{X}_u^{\chi+1}, \mathrm{Y}_u^{\chi+1}, \{R_{u,f}^{\chi+1}, l_{u,f}^{\chi+1}\}_{f\in\mathcal{F}}\}, \ldots, y_{u,\varphi-\chi} = \{\mathrm{X}_u^\varphi, \mathrm{Y}_u^\varphi, \{R_{u,f}^\varphi, l_{u,f}^\varphi\}_{f\in\mathcal{F}}\}$. The training and testing samples will be sent to the LSTM model for updating the parameters by the backward propagation algorithm and the Adam optimizer, based on the Mean Squared Error (MSE) loss between the testing data and model prediction.

C. Slice-Level Anomaly Detection Model Training

1) Slice Information Embedding: To train the anomaly detection model, we first extract each slice's information at each time slot. The requests from UEs, the resources of VNFs, and the service provision latency among VNFs and UEs jointly affect the anomaly status of slices. Moreover, considering the stateful UEs, their associated VNFs in previous time slots also need to be included. Thus, we combine each slice's information based on graph embedding, where UEs and VNFs are considered as GCN nodes, the requests and resources of VNFs are embedded as attributes of GCN nodes, and their connection/provision relationships are built as edges, respectively. Specifically, we define the graph embedding of slice s at time slot t as follows:

i) The nodes of the graph include two parts, expressed as $\vec{\mathbf{N}}_{s,1}^t = ((1,0),(2,0),\ldots,(|\mathcal{U}_s|,0))$ and $\vec{\mathbf{N}}_{s,2}^t = ((1+|\mathcal{U}_s|,1),(2+|\mathcal{U}_s|,1),\ldots,(|\mathcal{V}_s|+|\mathcal{U}_s|,1))$, where "0" indicates a UE node and "1" means a VNF node. We use $\mathrm{I}(\cdot)$ to denote the node index, where $\mathrm{I}(u) \in \left[1, |\mathcal{U}_s|\right]$ and $\mathrm{I}(v) \in \left[|\mathcal{U}_s|+1, |\mathcal{U}_s|+|\mathcal{V}_s|\right]$, respectively.

ii) The edges of the graph include three parts, where the first indicates the connections among the VNF nodes, expressed as $\vec{\mathbf{E}}_{s,1}^t = (\mathrm{I}(v), \mathrm{I}(v'))_{v,v'\in\mathcal{V}_s:\kappa_{v,v'}=1}$; the second part refers to the latency between the UE and VNF nodes, defined as $\vec{\mathbf{E}}_{s,2}^t = (\mathrm{I}(u), \mathrm{I}(v))_{u\in\mathcal{U}_s, v\in\mathcal{V}_s}$; and the third part indicates the resources provision mappings from VNF v to UE u, w.r.t., $\vec{\mathbf{E}}_{s,3}^t = (\mathrm{I}(u), \mathrm{I}(v))_{u\in\mathcal{U}_s, v\in\mathcal{V}_s}$. Here, the provisioned resources may change over time slots, the specific resources are reflected on the weights of the edges dynamically.

iii) The attributes of UE nodes indicate the request and can be written as $\vec{\mathbf{A}}_{s,1}^t = ((R_{u,f}^t, l_{u,f}^t)_{f\in\mathcal{F}}, \Upsilon_u^t)_{u\in\mathcal{U}_s}$; the attributes of VNF nodes are the total resources of VNFs, as when deciding the slice anomalies, VNFs don't actually allocate resources for UEs and only determine whether all requests can be fulfilled, thus, we have $\vec{\mathbf{A}}_{s,2}^t = (R_{v,f}^t)_{v\in\mathcal{V}_s, f\in\mathcal{F}}$.

iv) The weights of edges include three parts, where the first denotes the latencies among VNFs, expressed as $\vec{\mathbf{W}}_{s,1}^t = (\eta_s^V)_{v,v'\in\mathcal{V}_s:\kappa_{v,v'}=1}$; $\vec{\mathbf{W}}_{s,2}^t = (l_{u,v}^t)_{u\in\mathcal{U}_s, v\in\mathcal{V}_s}$ indicates the latencies between UEs and VNFs; the third part means the resource provisioned at time slot $t-1$, which identifies if a stateful request is served by the same VNF nodes, thus, we have $\vec{\mathbf{W}}_{s,3}^t = (R_{u,v,f}^{t-1})_{u\in\mathcal{U}_s, v\in\mathcal{V}_s, f\in\mathcal{F}}$.

As a result, the graph sample g_s^t for slice s at t can be built for GCN training, to this end, we combine the index, type, and attributes of nodes as $n = \textbf{Padding}([\vec{\mathbf{R}}_{s,1}^t; \vec{\mathbf{R}}_{s,2}^t])$, where row vectors $\vec{\mathbf{R}}_{s,1}^t$, $\vec{\mathbf{R}}_{s,2}^t$ are expressed by $\vec{\mathbf{R}}_{s,1}^t = (N_u, A_u)_{N_u\in\vec{\mathbf{N}}_{s,1}^t, A_u\in\vec{\mathbf{A}}_{s,1}^t, u\in\mathcal{U}_s}$ and $\vec{\mathbf{R}}_{s,2}^t = (N_v, A_v)_{N_v\in\vec{\mathbf{N}}_{s,2}^t, A_v\in\vec{\mathbf{A}}_{s,2}^t, v\in\mathcal{V}_s}$. Here, we handle the different dim of row vectors by the padding method, where the missing values of the matrix are extended by 0 according to the maximum length of the row vectors. The edge indies are expressed by $e^i = [\vec{\mathbf{E}}_{s,1}^t, \vec{\mathbf{E}}_{s,2}^t, \vec{\mathbf{E}}_{s,3}^t]^{\mathrm{T}}$, and the edge attributes

are expressed as $e^a = [(\vec{\mathbf{W}}^t_{s,1}); (\vec{\mathbf{W}}^t_{s,2}); (\vec{\mathbf{W}}^t_{s,3})]$, where we handle each weight attribute to a 1-dimension arrow. Thus, the data sample for GCN learning is formulated as a graph sample $g^t_s = (n, e^i, e^a, \Xi)$, where Ξ denotes the sample label.

2) GCN Model Training: To train the GCN model uniformly, we first process the graph samples by extending the nodes, as different slices may have various numbers of UEs/VNFs. The processed data is then sent to the convolution layers. Here, considering the importance of the edge attributes, we utilize the GINE convolution for feature extraction [38], [39], which is a trending extension of graph isomorphism network [40]. In this neural network, each convolution layer includes two fully connected layers, the max pooling layer combines all the outputs from the convolution layer, and the output layer predicts the probability of anomalies. Here, the parameters update in layer $z+1$ is expressed as

$$h_i^{z+1} = f_\psi((1+\epsilon)h_i^z + \sum\nolimits_{j\in\mathcal{N}_i} \text{ReLU}(h_j^z + e_{j,i}^z)), \quad (6)$$

where $\mathcal{N}_i$ denotes the neighboring nodes of node i, $e^z_{j,i}$ is the weight of edge between nodes i and j, h_i^z means the node feature at layer z, ϵ is a learnable parameter, and f_ψ is a neural network transformation.

To process the imbalanced datasets, we set the weighted cross-entropy as the criterion. Denoting the model predicts logits $\vec{\boldsymbol{\xi}}_d = (\xi_{d,0}, \xi_{d,1})$ for each sample d, which indicates the logits of the sample being anomaly or not, the cross-entropy loss for GCN training can be given by

$$\mathcal{L}^{\text{CE}} = (\sum\nolimits_{c=0}^{1} w_c q_c) \log(\sum\nolimits_{c=0}^{1} e^{\xi_{d,c}}) - \sum\nolimits_{c=0}^{1} w_c q_c \xi_{d,c}, \quad (7)$$

where w_0, q_0, and w_1, q_1 indicate the weight and label of normal and abnormal samples, respectively. Based on the loss function, the parameters of the GCN model are updated by the backward propagation technique based on the Adam optimizer.

D. UE-Level Service Provision Model Training

1) Markov Decision Process Formulation: When a slice operates normally at time slot t, the service provision from VNFs to UEs can be optimized to improve the system gain, by considering the requests of UEs, the resources of VNFs, and the latencies between them. When training DRL agents during UEs' interaction with the environment, due to intermittent slice anomalies, only UEs in normal slices can be optimized. This affects experience of agents that will be updated in subsequent time slots rather than the current one. Thus, we consider distributed DRL training at UEs without experience sharing among them. To this end, we model the Markov decision process for service provision as follows:

i) State: The state of UE u at time slot t should include three parts, the first includes UE requested resources, latency, and type; the second consists of the latencies from u to the VNFs in the slice, and the resources VNFs have provisioned in the previous time slot; the third includes the remaining resources of VNFs. These data are either directly obtained from the UE, or are periodically reported and stored in the slice orchestrator, from which each UE agent retrieves the full state before making decisions. Thus, we express the state as $\mathbf{S}^t_u = \{\{R^t_{u,f}\}_{f\in\mathcal{F}}, \{l^t_{u,f}\}_{f\in\mathcal{F}}, \Upsilon^t_u, \{\{R^{t-1}_{u,v,f}\}_{f\in\mathcal{F}}, l^t_{u,v}, R_{v,f} - \sum_{u\in\mathcal{U}_s} R^t_{u,v,f}\}_{v\in\mathcal{V}_s}\}$.

ii) Action: UE u decides how many resources it should obtain from the VNFs of the slice, we design the action related to the percentages of its requested resources. To optimize its alignment with model training, we denote the action as $\mathbf{A}^t_u = \{\zeta^t_{u,v,f}\}_{f\in\mathcal{F},v\in\mathcal{V}_s}$, where $\zeta^t_{u,v,f} \in [-1, 1]$. When u interacts with the environment, the percentages that u accesses v for resources can be obtained as $\hat{\zeta}^t_{u,v,f} = (\zeta^t_{u,v,f} + 1)/2 \in [0, 1]$. To further ensure stable convergence in distributed DRL model training, we introduce a scheduling restriction where the decision updates of UEs are performed sequentially to avoid mutual interference and enhance overall training stability. Note that in large-scale scenarios, this process can be conducted within sub-slices (grouping limited UEs) to ensure scalability and acceptable latency [41], [42].

iii) Reward Function: The action-indicated resource allocation from v is derived as $R^{t*}_{u,v,f} = \hat{\zeta}^t_{u,v,f} \times R^t_{u,f}$, which is directly output by the agent during training process. However, $R^{t*}_{u,v,f}$ may differ from the actual allocated resources considering the following constraints: *1)* when an action implies more resource allocation than VNFs' remaining resources, $R^t_{u,v,f}$ should be scaled to the remaining resources; *2)* $R^t_{u,v,f}$ should be 0 if $l^t_{u,v} \le l^t_{u,f}$, as v does not meet the UE's requested latency; *3)* if u is not associated with v in previous iterations and $\Upsilon^t_u = 1$, $R^t_{u,v,f}$ should be 0, as v cannot serve u due to the stateful request nature. We denote the differences between actual allocated resources and $R^{t*}_{u,v,f}$ as $\Gamma^t_u = \textbf{MSE}(R^t_{u,v,f}, R^{t*}_{u,v,f})$. To improve system NPS, RRL, and reduce cost, we define the reward of u as $\mathbf{R}^t_u = 100 \sum_{v\in\mathcal{V}_s} \sum_{f\in\mathcal{F}} (\hat{\zeta}^t_{u,v,f} \times \text{RRL}^t_{v,f}/l^t_{u,v}) + \Gamma^t_u \times \varrho$, where $\varrho < 0$ is a penalty coefficient, applied when constraints are violated. This reward function steers the agent to perform actions to improve the ratio of provided resources to latency and the RRL of slices.

2) DRL Agent Training: We aim to maximize the expected discounted cumulative reward for UE u by finding an optimal policy as $\pi^*_u = \text{argmax}_{\pi_u} \mathbb{E}_{(\mathbf{S}^t_u, \mathbf{A}^t_u \sim \rho_{\pi_u})}[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} \mathbf{R}^\tau_u]$, where τ denotes the step index, $\mathbf{R}^\tau_u$ denotes the reward received in step τ, γ denotes a discounted factor, and ρ_{π_u} denotes the state and state-action distribution introduced by policy π_u. Considering the continuous decision-making characteristics, the SAC algorithm, which employs the entropy regularized formalism to augment exploration to find the optimal policy, is adopted to solve this problem. Specifically, by introducing the entropy formalism, π^*_u can be rewritten as $\pi^*_u = \text{argmax}_{\pi_u} \mathbb{E}_{(\mathbf{S}^t_u, \mathbf{A}^t_u \sim \rho_{\pi_u})}[\sum_{\tau=0}^{\infty} \gamma^\tau (\mathbf{R}^{\tau+1}_u + \sigma H(\pi(\cdot|\mathbf{S}^\tau_u)))]$, where $H(\pi(\cdot|\mathbf{S}^\tau_u))$ denotes the entropy of policy π_u, σ represents the parameter that balances the entropy and reward. We denote V^t_u as the entropy-augmented accumulated return under state $\mathbf{S}^t_u$, where $V^t_u = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} (\mathbf{R}^{\tau+1}_u + \sigma H(\pi_u(\mathbf{A}^\tau_u|\mathbf{S}^\tau_u)))$. Accordingly, the soft-Q function to represent the expected soft return under $\mathbf{S}^t_u$ and $\mathbf{A}^t_u$ is given as $Q_u(\mathbf{S}^t_u, \mathbf{A}^t_u) = \mathbf{R}^t_u + \gamma \mathbb{E}_{\mathbf{S}^{t+1} \sim \rho_{\pi_u}}[V^{t+1}_u]$.

We use the Deep Neural Network (DNN) to approximate $Q_u(\mathbf{S}^t_u, \mathbf{A}^t_u)$ and policy π_u. To this end, we define the parameterized functions Q^θ_u and $\pi^{\theta'}_u$ and try to minimize the soft Bellman residual and expected Kullback-Leibler divergence

Algorithm 1: Offline Training Algorithm

Input: VNFs and UEs of slices $\mathcal{V}_s, \mathcal{U}_s, \forall s \in \mathcal{S}$.

1 **Initialize**: Each UE initializes $\{\theta_i\}_{i=1,2}$, $\{\hat{\theta}_i\}_{i=1,2}$, θ', historical stack $\mathcal{K}_u^{his} = \emptyset$. Set $\hat{\theta}_i \leftarrow \theta_i\ \forall i$, the cache of slice s as $\mathcal{D}_s = \emptyset$, the soft-update coefficient as o.
2 **for** Each time slot $t \leq \varphi$ **do**
3 Update the historical stack $\mathcal{K}_u^{his} \leftarrow \mathcal{K}_u^{his} + \{\mathrm{X}_u^t, \mathrm{Y}_u^t, \{R_{u,f}^t, l_{u,f}^t\}_{f\in\mathcal{F}}\}$, $\forall u$.
4 Label slices s, cache the data with label to $\mathcal{D}_s, \forall s$.
// SAC Training During Interaction
5 **for** Each UE $u \in \{\mathcal{U}_s | \Delta_s^t = 0, s \in \mathcal{S}\}$ **do**
6 Observes state $\mathbf{S}$, get action $\mathbf{A} = \zeta_{u,v,f}^t$.
7 Obtain resources from $v \in \mathcal{V}_s$ by $R_{u,v,f}^t$.
8 Get reward $\mathbf{R}$ and next state $\mathbf{S}'$.
9 Store tuple $(\mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{S}')$ in replay buffer.
10 Sample minibatch $\mathcal{B}$ from replay buffer with given batch size to calculate target values.
11 Update $\{\theta_i\}_{i=1,2}$, θ', and σ by calculating $\nabla\mathcal{L}(Q_u^\theta)$, $\nabla\mathcal{L}(\pi_u^{\theta'})$, and $\nabla\mathcal{L}(\sigma)$ based on (8), (9), and (10), respectively.
12 Update the target network parameters regularly by soft copy, w.r.t. $\hat{\theta}_i \leftarrow o\ \hat{\theta}_i + (1-o)\ \theta_i,\ \forall i$.
// LSTM and GCN Models Training
13 Cloud collects $\{\mathcal{K}_u^{his}\}_{u\in\mathcal{U}}$ and $\mathcal{D}_s, \forall s \in \mathcal{S}$, splits training and testing datasets.
14 **for** Each LSTM train epoch **do**
15 With $x_{u,1}, \ldots, x_{u,\varphi-\chi}$, fit the LSTM model, get the output as $\hat{y}_{u,1}, \ldots, \hat{y}_{u,\varphi-\chi}$.
16 Calculate the MSE loss between $\hat{y}_{u,1}, \ldots, \hat{y}_{u,\varphi-\chi}$ and $y_{u,1}, \ldots, y_{u,\varphi-\chi}$.
17 Update the parameters of LSTM by Adam optimizer and backward propagation method.
18 **for** Each GCN train epoch **do**
19 Build graph embedding sample $g_s^t = (n, e^i, e^a, \Xi)$.
20 Extend nodes and edge attributes of g_s^t to get sample $\hat{g}_s^t = (\hat{n}, \hat{e}^i, \hat{e}^a, \Xi)$.
21 With $\hat{g}_s^t$, the first GINE layer updates parameters by (6) and gets $\hat{g}_s^{t'} = (\hat{n}', \hat{e}^{i'}, \hat{e}^{a'}, \Xi)$.
22 The second GINE layer updates nodes and sends to the max pooling layer to predict logits $\vec{\boldsymbol{\xi}}_d$.
23 Update the parameters of the GCN model by calculating $\nabla\mathcal{L}^{\mathrm{CE}}$ based on (7).

Output: LSTM, GCN, and SAC models Θ^{LSTM}, Θ^{GCN}, and $\Theta_u^{\mathrm{SAC}}, \forall u \in \mathcal{U}_s, s \in \mathcal{S}$.

Algorithm 2: Online Inference Algorithm

Input: $\mathcal{U}$, $\mathcal{S}$, φ, χ, $\Theta_u^{\mathrm{SAC}}, \forall u \in \mathcal{U}$, Θ^{LSTM}, Θ^{GCN}.

1 **for** time slot $t > \varphi$ **do**
2 **for** $s \in \mathcal{S}$ **do**
3 **for** $u \in \mathcal{U}_s$ **do**
4 Get historical sequence with length χ.
5 Input the historical sequence to model Θ^{LSTM} to obtain the prediction $\mathbf{P}_u^{t+1}$.
6 Embed $\mathbf{P}_u^{t+1}, \forall u \in \mathcal{U}_s$ to graph samples.
7 Θ^{GCN} detects anomaly status Δ_s^{t+1}.
8 **if** $\Delta_s^{t+1} = 0$ **then**
9 Θ_u^{SAC} decides $\mathbf{A}_u^{t+1} = \zeta_{u,v,f}^{t+1}$ for $u \in \mathcal{U}_s$.
10 Obtain resources from $v \in \mathcal{V}_s$ by $R_{u,v,f}^{t+1}$.
11 **else**
12 Output anomaly trigger alarm Δ_s^{t+1}.

Output: $\Delta_s^{t+1}, \forall s \in \mathcal{S}$ or $\mathbf{A}_u^{t+1}, \forall u \in \mathcal{U}_s, \forall s \in \mathcal{S}$.

[43], respectively. The soft Bellman residual is expressed as

$$\mathcal{L}(Q_u^\theta) = \mathbb{E}_{(\mathbf{S}_u^t, \mathbf{A}_u^t)\sim\mathcal{B}}[1/2(Q_u^\theta(\mathbf{S}_u^t, \mathbf{A}_u^t) - \hat{Q}_u(\mathbf{S}_u^t, \mathbf{A}_u^t))^2], \quad (8)$$

where $\mathcal{B}$ and $\hat{Q}_u(\mathbf{S}_u^t, \mathbf{A}_u^t)$ denote a mini batch and target soft-Q function, and the expected Kullback-Leibler divergence is

$$\mathcal{L}(\pi_u^{\theta'}) = \mathbb{E}_{\mathbf{S}_u^t\sim\mathcal{B}}[\mathbb{E}_{\mathbf{A}_u^t\sim\pi_u^{\theta'}}(\sigma\ log\ \pi_u^{\theta'}(\mathbf{A}_u^t|\mathbf{S}_u^t) - Q_u^\theta(\mathbf{S}_u^t, \mathbf{A}_u^t))]. \quad (9)$$

When $\pi_u^{\theta'}(\mathbf{A}_u^t|\mathbf{S}_u^t)$ is directly used to generate samples, θ' cannot be backpropagated in a normal manner, we adopt the reparameterization trick to solve this issue [43]. In addition, when an action space has been almost explored, it is more reasonable to adjust the parameter σ to a smaller value. We configure σ to be learned in the DNN and the loss can be given as

$$\mathcal{L}(\sigma) = \mathbb{E}_{\mathbf{S}_u^t\sim\mathcal{B}}[\mathbb{E}_{\mathbf{A}_u^t\sim\pi_u}(-\sigma\ log\ \pi_u^{\theta'}(\mathbf{A}_u^t|\mathbf{S}_u^t) - \sigma H_0], \quad (10)$$

where H_0 is the target entropy.

As a result, Algorithm 1 summarizes the training processes for the prediction, anomaly detection, and service provision models. Note that the service provision model is trained during the interaction at each time slot, while the other models are trained afterward. The complexity of Algorithm 1 during the interaction mainly comes from Lines 4, 6, and 11. We label the slices by latency filtering, resource feasibility check, and identification of stateful UEs' linked VNFs [1], which takes $\mathcal{O}(|\mathcal{U}_s| \times |\mathcal{V}_s|)$ to complete. The propagation of neural networks in Lines 6 and 11 and the LSTM/GCN training are related to the sample sizes, we denote the training episodes of SAC as E and the epochs of LSTM/GCN as E', both with l hidden DNN layers, each layer has h neurons. Overall, the complexity of Algorithm 1 can be expressed as $\mathcal{O}(|\mathcal{U}_s| \times |\mathcal{V}_s|) + E'hlk + Eh((4h+5f+10+3l)+(\mathbf{N}+\mathbf{E})(2f+4+lh))$, where k denotes the number of random samples in SAC training, $\mathbf{N}$ and $\mathbf{E}$ denote the number of nodes and edges in the GCN model, respectively.

E. Online Inference

After φ time slots' interaction with the environment, at each time slot $t > \varphi$ in the online inference phase, we utilize the well-trained models to infer the anomalies and service provision strategies for the next time slot $t+1$, the process is summarized in Algorithm 2. Here, for efficient distributed inference, each UE deploys a copy of the global well-trained LSTM model and the local SAC model, while each slice's orchestrator deploys a copy of the global GCN model to capture the anomalies from the slice perspective. Specifically,

TABLE II
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value	Parameter	Value
Number of ESs	10	ES-UE latency	2 ms	LSTM/GCN batch size	256
Area size	10×10 KM^2	Inter-ES latency	6 ms	LSTM/GCN hidden layers	2
Number of UEs	50	VNF latency (low/high)	4/8 ms	LSTM/GCN neurons	64
ES resources	[4, 8, 128]	Cloud latency (low/high)	100/150 ms	SAC parameters	LR=0.001, $\gamma = 0.99$
UE request mean	[0.01, $\lvert\mathcal{M}\rvert/\lvert\mathcal{U}\rvert R_{m,f}$]	Time slots (training)	1000	Stateful UE ratio (λ)	0.5
UE request var	[0.01, $\lvert\mathcal{M}\rvert/2\lvert\mathcal{U}\rvert R_{m,f}$]	Time slots (inference)	200	Satisfaction (α_f)	10
Latency request	40-150 ms	Replay buffer size	256	Penalty (δ_f)	-2
History length φ	1000	Constraint penalty	-10	LSTM series size (χ)	5 time slots

for time slots $t > \varphi$, we first utilize the well-trained LSTM model to predict each UE's request at time slot $t+1$, expressed as $\mathbf{P}_u^{t+1} = \{\hat{\mathrm{X}}_u^{t+1}, \hat{\mathrm{Y}}_u^{t+1}, \{\hat{R}_{u,f}^{t+1}, \hat{l}_{u,f}^{t+1}\}_{f\in\mathcal{F}}\}$. Then, we embed the prediction with system information to graph samples, and the GCN model inferences the anomaly status of slice s at $t+1$, w.r.t. Δ_s^{t+1}. An anomaly alarm will be triggered for a slice with $\Delta_s^{t+1} = 1, \forall s \in \mathcal{S}$, after which the slice will be adjusted by manipulating their VNFs to migrate between physical hosts, scale the allocated resources, and even change the connection relationships among VNFs [4], [34]. For the slices that operate normally, the DRL agents of UEs integrate the prediction to infer the resource provision strategies. Overall, the complexity of Algorithm 2 can be expressed as $\mathcal{O}(h((4h+5f+10+(3+k)l)+(\mathbf{N}+\mathbf{E})(2f+4+lh)))$.

V. EVALUATION RESULTS

In this section, we present the evaluation results, and the main experiment parameters are summarized in Table II. Note that in the experiments, when the comparative parameters vary, the others remain fixed to isolate the effect.

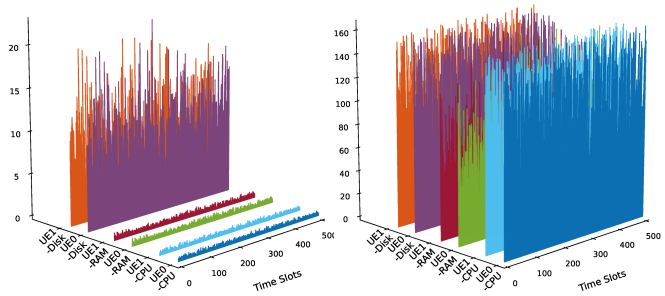
A. Settings

To evaluate the framework, we first consider 10 ESs are geographically evenly distributed within a 10×10 KM^2 square area, where 50 UEs dynamically move across and make requests. The total resources of ESs, w.r.t., $\{R_{m,f}\}_{f=\{1,2,3\}}$ are uniformly set as 4 CPUs, 8 GB RAM, and 128 GB disk. To mimic the IID assumption of UEs' requests, we consider their requests form a normal distribution, with the mean and variance of the requested resources randomly set from 0.01 to $\frac{|\mathcal{M}|}{|\mathcal{U}|}R_{m,f}$ and $\frac{|\mathcal{M}|}{2|\mathcal{U}|}R_{m,f}$. The mean and variance of latencies are randomly set as $40-150$ ms and $20-75$ ms, the parameter λ, denoting the ratio of UEs with stateful requests, is set as 0.5. Furthermore, the average wireless latency from UEs to APs η^W is set as 2 ms, while the average optical latency between adjacent ESs η^O is set as 6 ms. As the slices can be configured with varying latency performances, we set the virtual link latency between consecutive VNFs η_s^V as $4-8$ ms, and the latency for VNFs deployed in ESs to access the cloud η_s^C as $100-150$ ms, which depends on the configuration of the slice. The allocation of CPU, RAM, and disk resources is jointly determined, where the former is treated as a continuous variable representing proportional core utilization, and the others are measured in bits.

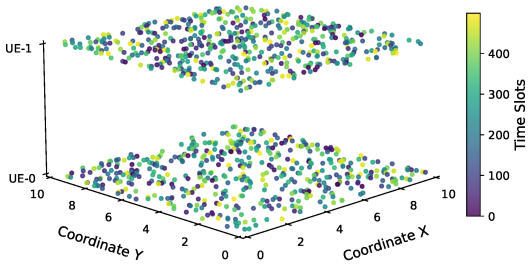
While training the LSTM and GCN models, we set the maximum training epochs as 1000 with the early-stop technique, with the maximum patience set as 150 epochs [44]. We consider 1000 and 200 time slots for the training and inference phases, respectively. At each time slot, the slices are labeled to be normal/abnormal, we complete this process by two steps: *1)* Latency filtering and resource feasibility check, wherein we test whether slices can satisfy UEs' extreme resource demands via latency and resource feasibility checks in the manner of [1]; *2)* Check that if the stateful requests are assigned only to previously linked VNFs. As a result, a dataset of four slices with labels is provided in [45].

B. Baselines and Performance Metrics

In this paper, we consider the baselines as follows: *1)* Different anomaly detection models including the GCN model without GINE integration (GCN-no-GINE) [46], the AutoEncoder (AE) model [47], the Mahalanobis Distance Detection (MDT) model [48], and a convex-optimization-based method Gurobi that tries to maximize the number of provisioned UEs and check if any UE remains unserved in the optimal solution [49]. Here, the GCN-no-GINE model serves as an ablation baseline that removes edge attributes to assess the contribution of edge and attributes. Additionally, considering the gap between the prediction and real user requests, we introduce the GCN-RNN as a baseline [12]; furthermore, to see the best possible performance regarding predictions, we collect UEs' real request data after the training iterations and send it to the GCN model as a baseline, referred as the Proposed-Real method. *2)* Different optimization models including the advantage Actor Critic (A2C), DDPG, Twin Delayed DDPG (TD3), and Proximal Policy Optimization (PPO) models, which are generally widely employed in DRL tasks [50]. Moreover, to see the advantage of the distributed SAC algorithm without experience sharing among UEs, we add FedAve and FedWeight paradigms as baselines, where the model parameter weights of FedWeight are set as the reward weights of agents. To see the advantage of UE observations design, we introduce the GraphSAC as a baseline [51], [52], which observes global slice information for service provision. At last, to see the advantage compared to conventional optimization-based methods, we integrate the Interior Point Methods (IPM) [53] with the greedy method as a baseline (IPM-greedy), where we try to allocate resources based on the IPM method for UEs in normal slices. If the constraints cannot be met, the resources are provided greedily.

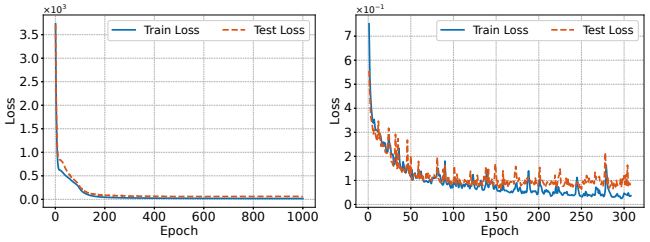


(a) UE Request Resources (b) UE Request Latency



(c) UE Moving Trajectory

Fig. 3. The request of two UEs over 500 time slots.



(a) LSTM Loss (b) GCN Loss

Fig. 4. The loss of LSTM and GCN models versus training epochs.

To see the anomaly detection performance, we consider the most common classification indicators, i.e., accuracy, precision, recall, and F1-score for the anomaly detection task, which can be calculated based on the confusion matrix [23]. To see the service provision optimization performance, we adopt the system NPS, RRL, MNO cost, and gain as the indicators.

C. UE Request and Mobility Analysis

We first illustrate two UEs' request data and mobility trajectory of 500 time slots, as illustrated in Fig. 3. Specifically, in Fig. 3(a) and Fig. 3(b), we can observe that the requested CPU, RAM, and disk resources fluctuate over time slots with a similar tendency, as these resources have the same data distribution and only have small fluctuations different from each other. The CPU and RAM resources are much less than the requested disk resources, while the requested latency of RAM resources is much stricter than the other two resources.

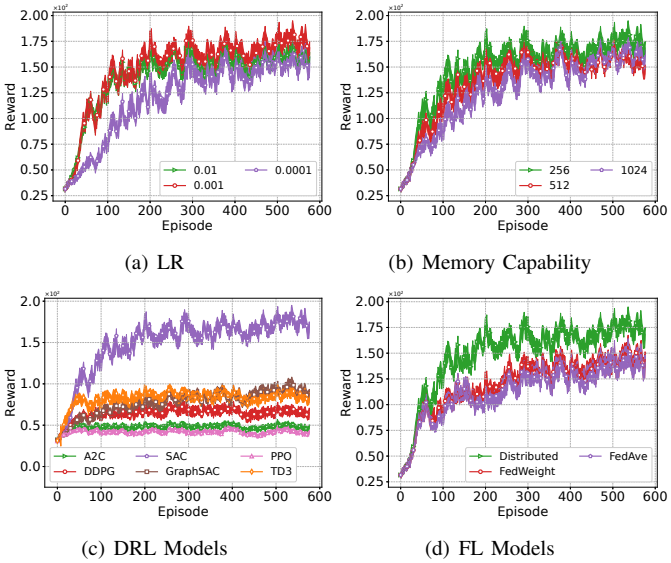
The moving trajectories of two UEs over 500 time slots are visualized in the lower and upper halves of Fig. 3(c), respectively, where the two UEs show different movement patterns. UE-1 moves in the center-left portion during the early stage, while UE-0 displays a more dispersed trajectory across the entire area. The color gradient of the scatter points represents the temporal progression, revealing the dynamic nature of their movements over time slots.

D. Training Performance and Parameter Analysis

We then illustrate the losses of the LSTM and GCN models in Fig. 4. In Fig. 4(a), we observe the training and test losses of the LSTM model decrease rapidly in the first 200 epochs and converge smoothly after 250 epochs. The model achieves stable learning with a small gap between training and test losses, indicating satisfied generalization performance. In Fig. 4(b), the losses of GCN model start at a higher value around 0.8 and fluctuate during training. Despite the oscillations, both losses decrease significantly in the first 50 epochs and stabilize around 0.1 after 100 epochs, while the training loss still decreases in a small range but the testing loss remains stable. With early stopping, the training terminates at nearly 250 epochs when no further improvement is observed.

We then present the average reward of the SAC models of UEs with different learning rate (LR) and memory capability, and the reward of different DRL models and training paradigms in Fig. 5. Note that UEs only perform DRL training when the slice is normal at each time slot, the training episodes of agents are generally less than 1000. From Fig. 5(a), we can see the LR 0.01 and 0.001 have close performance and increase rapidly before 200 episodes, while the other one converges more slowly. The reward of all LRs gradually gets stable around 300 episodes, while the LR 0.001 has a slight advantage compared to others. In Fig. 5(b), we can see the reward of memory capacity set as 256 has the best performances at the beginning, followed by 512 and 1024, as higher memory leads to a slower training speed. The rewards of 512 and 1024 then gradually converge to a close level, but still lower than 256. In Fig. 5(c), the proposed scheme has the best reward, followed by the TD3/GraphSAC, DDPG, A2C, and PPO methods. As SAC, TD3, and DDPG are off-policy models and are more suitable for continuous variable-based DRL tasks. Moreover, the GraphSAC method shows worse performance, as the UEs make decisions independently and slice-level graph observations will lead to more noise. In Fig. 5(d), the proposed distributed SAC algorithm has better performances compared to the other algorithms, as the sharing of experience among agents will lead to a decrease of training performance due to interference from UEs.

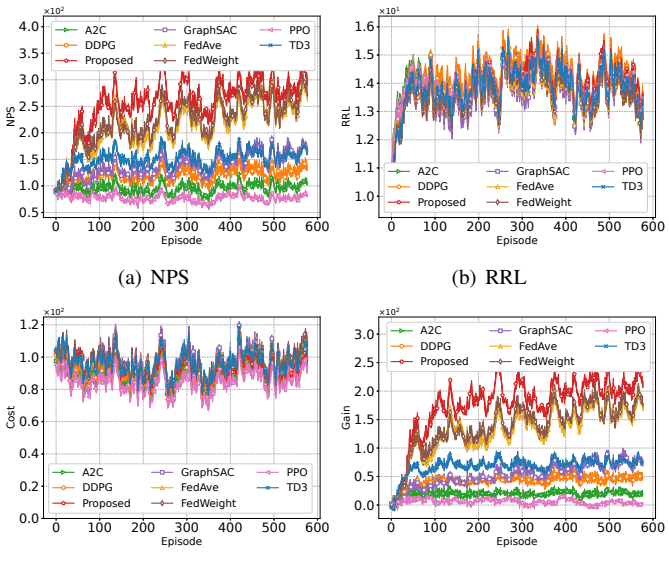
Finally, to see the optimization of system indicators, we illustrate the system indicators during the DRL training, as shown in Fig. 6. From Fig. 6(a), we observe that the NPS of all models increases with episodes, the proposed distributed SAC scheme has the highest NPS, followed by FedAve, FedWeight, TD3, DDPG, A2C, and PPO-based schemes. This aligns with the reward performances of all schemes. In Fig. 6(b), we can see that RRL increases markedly in the first nearly 50 episodes and then gets stabilized, indicating the agent quickly learns to balance load by steering requests to nodes with more available resources, which validates that the DRL agents increase more remaining resources to cope with UE



(a) LR (b) Memory Capability

(c) DRL Models (d) FL Models

Fig. 5. The reward of different SAC training parameters and training models/paradigms versus episodes.



(a) NPS (b) RRL

(c) Cost (d) Gain

Fig. 6. The system indicators of different models versus training episodes.

requests and enhance system robustness. In Fig. 6(c), the system cost is non-increasing and slightly declines as the agent learns to allocate just-enough VNF resources and avoid over-provisioning. As the cost maintains a higher base and is divided by the VNFs' latency η_s^V, the saved resources do not significantly lead to its reduction. Finally, as shown in Fig. 6(d), the proposed scheme obtains the highest system gain compared to other schemes. Comparing to the baselines, our method consistently attains higher NPS while maintaining similar RRL and cost, thereby achieving QoE improvements without additional resource expenditure or increased cost.

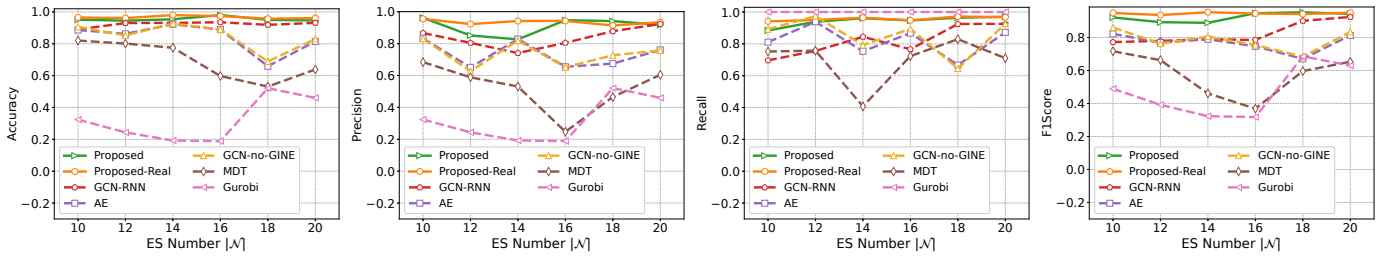
E. Evaluation Results on Different Numbers of ESs

We present the anomaly detection indicators with different $|\mathcal{M}|$, as shown in Fig. 7. From Fig. 7(a) we can see the proposed GCN model has relatively stable performance in all ESs, while the Proposed-Real scheme has the best performance, followed by the proposed, GCN-RNN, AE/GCN-no-GINE, and the Gurobi schemes. The gap between the proposed scheme and the Proposed-Real scheme is very small, while the GCN-RNN method falls behind. The AE and GCN-no-GINE schemes have close and relatively worse performance compared to the GCN-based schemes, as the graph information like the service provision mapping, is not captured. Here, $|\mathcal{M}| = 18$ is a special case, wherein we observe UEs' requested resources are relatively less than the general patterns, leading to a decrease of anomalies. This pattern change also affects the AE, GCN-no-GINE, and Gurobi schemes. In Fig. 7(b)-7(c), most schemes achieve high precision and recall escapes the Gurobi scheme, as it erroneously detects abnormal slices due to failure of problem-solving, leading to high recall and low precision. The proposed scheme has relatively stable behavior while other schemes gradually fluctuate with $|\mathcal{M}|$. From Fig. 7(d), we can see that the proposed scheme has a small gap with the Proposed-Real scheme but still outperforms other baselines when $|\mathcal{M}| \leq 14$. With $|\mathcal{M}|$ increases, the gap becomes very small and even disappears. Overall, the proposed scheme can improve the F1-score by up to 17.10%, 28.24%, 29.59%, 61.08%, and 66.47% compared to the GCN-RNN, GCN-no-GINE, AE, MDT, and Gurobi schemes. Compared to the Proposed-Real scheme, the maximum gap is 7.21%.

Meanwhile, we present the system indicators with different $|\mathcal{M}|$, as shown in Fig. 8. In Fig. 8(a), we see that the proposed scheme outperforms all baselines. When $|\mathcal{M}| \geq 16$, the proposed scheme has a slight decrease in performance. This is because increasing $|\mathcal{M}|$ amplifies the latency among VNFs due to the growing topologies complexity, and the NPS suffers from a latency lower bound. Fig. 8(b) shows the RRLs of all schemes increases with $|\mathcal{M}|$, as more ESs can provide more resources for UEs. The RRLs converge as $|\mathcal{M}|$ reaches 18 and 20, with minimal differences observed among the schemes. In Fig. 8(c), we can see the MNO cost increases with $|\mathcal{M}|$, the proposed, FedWeight, and TD3 schemes incur higher costs, especially when $|\mathcal{M}| \geq 14$, which could be attributed to more resource allocation to UEs. A2C and PPO consistently show lower costs across different $|\mathcal{M}|$. Finally, even the proposed scheme incurs higher costs in some cases, it achieves higher system gain across different $|\mathcal{M}|$, as shown by Fig. 8(d). This suggests that the proposed method offers a trade-off between resource utilization and optimization performance. Overall, when $|\mathcal{M}| \in [10, 20]$, the proposed scheme can improve the system gain by up to 87.80%, 86.99%, 62.58%, 88.71%, 120.91%, 128.69%, 137.56%, and 139.42% compared to the FedAve, FedWeight, TD3, DDPG, A2C, GraphSAC, IPM-Greedy, and PPO schemes, respectively.

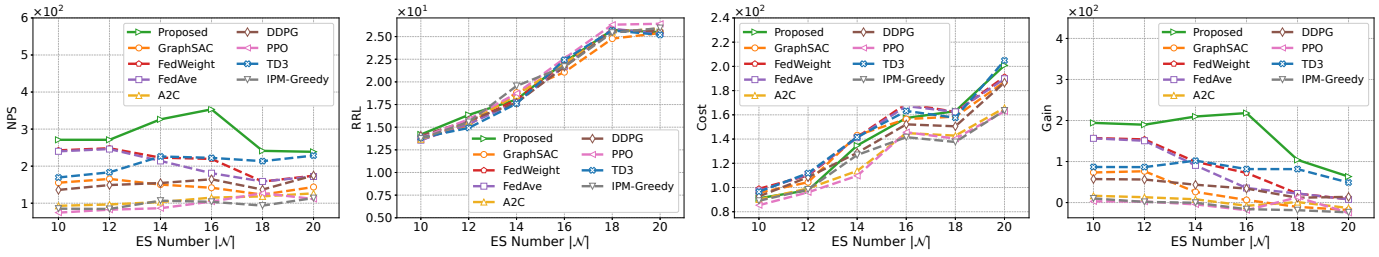
F. Evaluation Results on Different Numbers of UEs

Then, we present the anomaly detection indicators with different $|\mathcal{U}|$, as shown in Fig. 9. From Fig. 9(a), we can see that the proposed and Proposed-Real schemes consistently outperform other methods, maintaining high accuracy around $90\% - 95\%$. The AE and GCN-no-GINE schemes



(a) Accuracy (b) Precision (c) Recall (d) F1-score

Fig. 7. The accuracy, precision, recall, and F1-score of different models versus ES number $|\mathcal{M}|$.



(a) NPS (b) RRL (c) Cost (d) Gain

Fig. 8. The system NPS, RRL, cost, and gain of different models versus ES number $|\mathcal{M}|$.

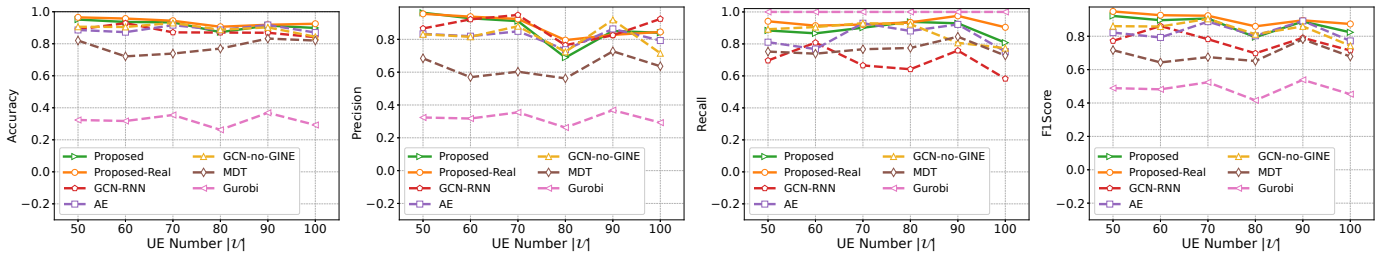
show similar performances, slightly lower than the GCN-based methods, followed by the MDT and Gurobi schemes. Fig. 9(b) demonstrates that most methods achieve high precision, except for the MDT and Gurobi methods. Fig. 9(c) shows that the Gurobi method achieves perfect recall, as it tends to detect normal slices to be abnormal. The proposed and Proposed-Real schemes maintain high recall. In Fig. 9(d), the proposed and Proposed-Real schemes outperform other baselines, and their gap is relatively small. Here, some anomalies arise from resource demands exceeding VNF capacity rather than accessing latency, limiting the effectiveness of GCN's latency-based edges and yielding results similar to or only slightly better than baselines. Overall, the proposed scheme can improve the F1-score by up to 16.11%, 11.54%, 9.86%, 28.13%, and 47.65% compared to the GCN-RNN, AE, GCN-no-GINE, MDT, and Gurobi schemes, respectively. Compared to the Proposed-Real scheme, the maximum gap is 8.17%.

We present the system indicators with different $|\mathcal{U}|$ in Fig. 10. From Fig. 10(a), we can see the proposed method significantly outperforms other baselines in terms of NPS, the FedWeight and FedAve methods show the next best performance, while other methods lag behind. In Fig. 10(b), all schemes show fluctuating trends in RRL as $|\mathcal{U}|$ changes, with a notable peak when $|\mathcal{U}| = 90$. In this case, we observe an unreasonable data distribution and many VNFs remain idle while the others' resources are exhausted. In Fig. 10(c), the MNO costs of schemes increase with $|\mathcal{U}|$, since more UE will request more resources. The proposed method, along with FedWeight and FedAve, incur higher costs compared to other baselines but introduce much better NPS. Finally, the proposed method demonstrates better system gain compared to other baselines across all $|\mathcal{U}|$ in Fig. 10(d), with the advance widening as $|\mathcal{U}|$ increases. Overall, the proposed scheme improves the system gain by up to 40.96%, 38.32%, 64.84%, 74.96%, 82.99%, 97.54%, 103.59%, and 105.01% compared to the FedAve, FedWeight, TD3, GraphSAC, DDPG, A2C, IPM-Greedy, and PPO schemes, respectively.

G. Anomaly Prevention and Time Efficiency Analysis

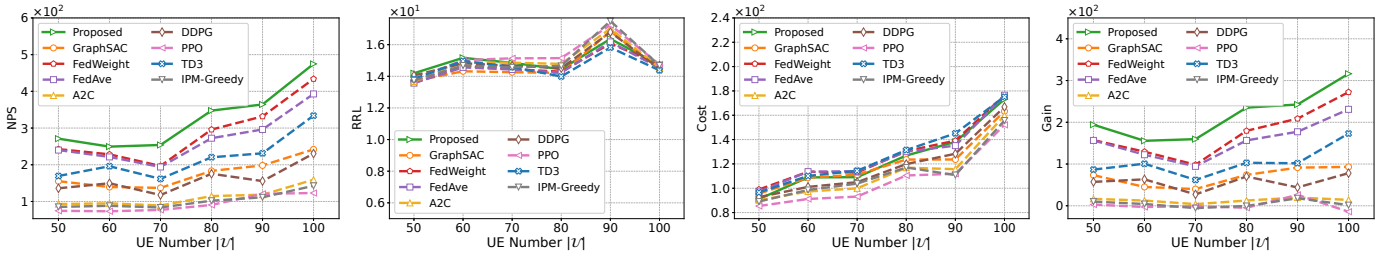
Fig. 11-12 illustrate the effect of the stateful UE ratio λ on preventing abnormal UEs and the run time of the detection scheme. In Fig. 11(a), as λ increases from 0.25 to 0.75, the abnormal UEs increase significantly across $|\mathcal{M}|$. Compared to the case $\lambda = 0.25$, $\lambda = 0.5$ leads to a 30.54% increase of abnormal UEs, while $\lambda = 0.75$ leads to 81.16% increase. This indicates more stateful requests will result in more abnormal UEs. To illustrate the impact on slice anomalies, we explicitly labeled the Abnormal Slices (AS), which typically depends on the distribution of abnormal UEs. Even with more anomalous UEs, if they are concentrated in a few specific slices, the number of AS may remain the same. However, in general cases, the abnormal UEs tend to cluster in fewer slices at low loads but spread system-wide as abnormal UEs rise.

Similarly, Fig. 11(b) shows the impact of λ across different $|\mathcal{U}|$. Abnormal UEs increase by 29.68% when λ increases from 0.25 to 0.5. When λ reaches 0.75, we observe an 80.52% increase. This demonstrates that by segregating stateful and stateless requests, the abnormal UEs can be reduced especially for scenarios with various stateless requests. Moreover, with fewer UEs ($|\mathcal{U}| = 50$), anomalies are localized within 2 specific slices; with $|\mathcal{U}|$ increases to $|\mathcal{U}| = 100$, the abnormal UEs gradually spread across all 4 slices. Note that in the simulation, we intentionally configure many abnormal UEs to assist model training and validate mitigation effectiveness. In actual deployments with NSM, a threshold mechanism can be introduced to reduce the adjustment overhead.



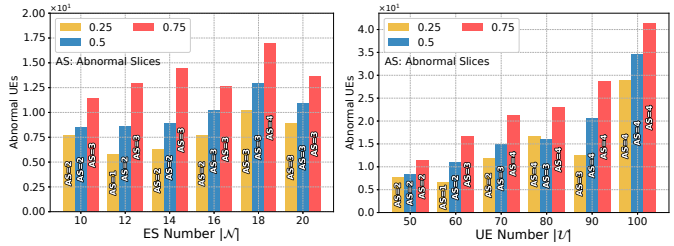
(a) Accuracy (b) Precision (c) Recall (d) F1-score

Fig. 9. The accuracy, precision, recall, and F1-score of different models versus UE number $|\mathcal{U}|$.



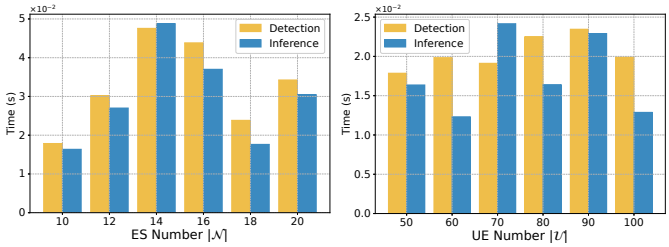
(a) NPS (b) RRL (c) Cost (d) Gain

Fig. 10. The system NPS, RRL, cost, and gain of different models versus UE number $|\mathcal{U}|$.



(a) Abnormal UEs versus $|\mathcal{M}|$ (b) Abnormal UEs versus $|\mathcal{U}|$

Fig. 11. The abnormal UEs with different stateful ratios versus ES number $|\mathcal{M}|$ and UE number $|\mathcal{U}|$, with abnormal slices indicated.



(a) Detection time versus $|\mathcal{M}|$ (b) Detection time versus $|\mathcal{U}|$

Fig. 12. The detection time versus ES number $|\mathcal{M}|$ and UE number $|\mathcal{U}|$.

Finally, Fig. 12(a)-12(b) compare the inference and detection time, and the inference efficiency outperforms the detection in most points. Special cases are $|\mathcal{M}| = 14$ and $|\mathcal{U}| = 70$, wherein we find the data here introduces much stricter latency requirements, and most anomalies are directly recognized by latency filtering, leading to tiny detection time. Overall, with different $|\mathcal{M}|$, the inference reduces time by up to 35.19% compared to the detection method. With various $|\mathcal{U}|$, the improvements can be up to 61.13%.

VI. Conclusion and Future Work

In this paper, we have investigated proactive anomaly detection and resource allocation in network slices, aiming to maximize the long-term system gain based on user request analysis. An offline-online framework that incorporates request segregation, GCN-based anomaly detection, and SAC-based resource optimization, is designed to solve this problem. The GCN model embeds holistic slice information as graph samples and the SAC models are distributedly trained for service provision optimization in normal slices to alleviate the interference from abnormal UEs. As a result, extensive evaluations demonstrate that our proposed scheme outperforms existing baselines in anomaly detection performance, improves the system gain, reduces abnormal UEs, and improves detection efficiency. Furthermore, considering the unique characteristics of each user's requests and the dynamic nature of real-world deployment scenario, future work will explore integrating a shared base model with user-specific adaptations and investigate fully online framework design. Additionally, a more flexible slice anomaly mechanism can be investigated, where the slice is allowed to tolerate a controllable proportion of abnormal UEs.

References

[1] Z. Ming, H. Yu, and T. Taleb, "User request provisioning oriented slice anomaly prediction and resource allocation in 6G networks," in *Proc. IEEE International Conference on Communications (ICC)*, June 2024, pp. 3640–3645.

[2] A. Boualouache, T. E. T. Djaidja, S.-M. Senouci, Y. Ghamri-Doudane, B. Brik, and T. Engel, "Deep learning-based intra-slice attack detection for 5G-V2X sliced networks," in *Proc. IEEE Vehicular Technology Conference (VTC-Spring)*, June 2022, pp. 1–5.

[3] X. Yuan, C. Sun, and S. Chen, "Cooperative DNN partitioning for accelerating DNN-empowered disease diagnosis via swarm reinforcement learning," *Appl. Soft Comput.*, vol. 148, p. 110844, Sept. 2023.

[4] Z. Ming, H. Yu, and T. Taleb, "Federated deep reinforcement learning for prediction-based network slice mobility in 6G mobile networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 11 937–11 953, Dec. 2024.

[5] S. Bukhari, K. Sharif, L. Zhu, C. Xu, F. Li, and S. Biswas, "Dynamic fine-grained SLA management for 6G eMBB-plus slice using mDNN & smart contracts," *IEEE Trans. Services Comput.*, Sept. 2024.

[6] T. Taleb, R. L. Aguiar, I. Grida Ben Yahia, B. Chatras, G. Christensen, U. Chunduri, A. Clemm, X. Costa, L. Dong, J. Elmirghani *et al.*, "White paper on 6G networking," June 2020.

[7] W. Wang, Q. Chen, T. Liu, X. He, and L. Tang, "A distributed online learning approach to detect anomalies for virtualized network slicing," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2021, pp. 1–6.

[8] J. Lian, L. Wang, H. Sun, and H. Huang, "GT-HAD: Gated transformer for hyperspectral anomaly detection," *IEEE Trans. Neural Netw. Learn. Syst.*, Feb. 2024.

[9] F. Zhou, P. Yu, L. Feng, X. Qiu, Z. Wang, L. Meng, M. Kadoch, L. Gong, and X. Yao, "Automatic network slicing for IoT in smart city," *IEEE Wirel. Commun.*, vol. 27, no. 6, pp. 108–115, Dec. 2020.

[10] 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, Jan. 2022.

[11] K. Abbas, T. A. Khan, M. Afaq, and W.-C. Song, "Ensemble learning-based network data analytics for network slice orchestration and management: An intent-based networking mechanism," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Apr. 2022, pp. 1–5.

[12] A. Chawla, A.-M. Bosneag, and A. Dalgkitsis, "Graph-based interpretable anomaly detection framework for network slice management in beyond 5G networks," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Apr. 2023, pp. 1–6.

[13] Y. Wu, L. Liang, Y. Jia, and W. Wen, "HFL-TranWGAN: Knowledge-driven cross-domain collaborative anomaly detection for end-to-end network slicing," *IEEE Trans. Netw. Service Manag.*, Jan. 2024.

[14] M. Masoudi, Ö. T. Demir, J. Zander, and C. Cavdar, "Energy-optimal end-to-end network slicing in cloud-based architecture," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 574–592, Mar. 2022.

[15] Z. Shu and T. Taleb, "A novel QoS framework for network slicing in 5G and beyond networks based on SDN and NFV," *IEEE Netw.*, vol. 34, no. 3, pp. 256–263, June 2020.

[16] Y. Cui, X. Huang, P. He, D. Wu, and R. Wang, "QoS guaranteed network slicing orchestration for Internet of Vehicles," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 15 215–15 227, Aug. 2022.

[17] 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.

[18] 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 International Conference on Communications (ICC)*, June 2021, pp. 1–6.

[19] 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.

[20] Y. Chen, Q. Yin, Q. Li, Z. Liu, K. Xu, Y. Xu, M. Xu, Z. Liu, and J. Wu, "Learning with semantics: Towards a semantics-aware routing anomaly detection system," in *Proc. USENIX Security Symposium*, Aug. 2024, pp. 5143–5160.

[21] S. Roy and N. Feamster, "Characterizing correlated latency anomalies in broadband access networks," in *Proc. ACM Special Interest Group on Data Communications (ACM SIGCOMM)*, Aug. 2013, pp. 525–526.

[22] J. He, N. Cheng, Z. Yin, H. Zhou, C. Zhou, K. Aldubaikhy, A. Alqasir, and X. Shen, "Load-aware network resource orchestration in LEO satellite network: A GAT-based approach," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15 969–15 984, Jan. 2024.

[23] Z. Ming, X. Li, C. Sun, Q. Fan, X. Wang, and V. C. M. Leung, "Sleeping cell detection for resiliency enhancements in 5G/B5G mobile edge-cloud computing networks," *ACM Trans. Sensor Netw.*, vol. 18, no. 3, pp. 1–30, July 2022.

[24] J. Soldani and A. Brogi, "Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey," *ACM Comput. Surv.*, vol. 55, no. 3, pp. 1–39, Feb. 2022.

[25] H. Qiao, H. Tong, B. An, I. King, C. Aggarwal, and G. Pang, "Deep graph anomaly detection: A survey and new perspectives," *IEEE Trans. Knowl. Data Eng.*, vol. 37, no. 9, pp. 5106–5126, June 2025.

[26] X. Zheng, B. Wu, A. X. Zhang, and W. Li, "Improving robustness of GNN-based anomaly detection by graph adversarial training," in *Proc. Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)*, May 2024, pp. 8902–8912.

[27] 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.

[28] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2018, pp. 1943–1951.

[29] H. Yu, T. Taleb, J. Zhang, and H. Wang, "Deterministic latency bounded network slice deployment in IP-over-WDM based metro-aggregation networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 2, pp. 596–607, Nov. 2021.

[30] N. M. Yungaicela-Naula, V. Sharma, and S. Scott-Hayward, "Misconfiguration in O-RAN: Analysis of the impact of AI/ML," *Comput. Netw.*, vol. 247, p. 110455, June 2024.

[31] S. Lee, "Net promoter score: Using NPS to measure IT customer support satisfaction," in *Proc. ACM SIGUCCS*, Oct. 2018, pp. 63–64.

[32] H. Vijayakumar, "Revolutionizing customer experience with AI: a path to increase revenue growth rate," in *Proc. ECAI*, June 2023, pp. 1–6.

[33] S. Dutta, D. Roy, and G. Das, "Modified split-rendering architecture to enable AI-assisted application-aware MAC for XR slice," *IEEE Netw. Lett.*, pp. 1–1, June 2023.

[34] 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.

[35] C. Puliafito, C. Cicconetti, M. Conti, E. Mingozzi, and A. Passarella, "Stateless or stateful FaaS? I'll take both!" in *Proc. IEEE International Conference on Smart Computing (IEEE SmartComp)*, June 2022, pp. 62–69.

[36] R. Cohen, M. Kadosh, A. Lo, and Q. Sayah, "LB scalability: Achieving the right balance between being stateful and stateless," *IEEE/ACM Trans. Netw.*, vol. 30, no. 1, pp. 382–393, Sept. 2021.

[37] R. de Jesus Martins, C. B. Both, J. A. Wickboldt, and L. Z. Granville, "Virtual network functions migration cost: from identification to prediction," *Comput. Netw.*, vol. 181, p. 107429, May 2020.

[38] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *Proc. ICLR*, Apr. 2020.

[39] PyTorch Geometric. Graph isomorphism network with edge features (GINEConv). [Online]. Available: https://pytorch-geometric.readthedocs.io/en/2.5.1/generated/torch_geometric.nn.conv.GINEConv.html

[40] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. ICLR*, Apr. 2019.

[41] J. Li, W. Shi, P. Yang, Q. Ye, X. S. Shen, X. Li, and J. Rao, "A hierarchical soft RAN slicing framework for differentiated service provisioning," *IEEE Wirel. Commun.*, vol. 27, no. 6, pp. 90–97, Dec. 2020.

[42] M. O. Ojijo, D. Ramotsoela, and R. A. Oginga, "Slice admission control in 5G wireless communication with multi-dimensional state space and distributed action space: A sequential twin actor-critic approach," *Comput. Netw.*, vol. 255, p. 110878, Dec. 2024.

[43] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. M. Leung, "Cooperative computation offloading for multi-access edge computing in 6G mobile networks via soft actor critic," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 6, pp. 5601–5614, Nov. 2024.

[44] Y. Bai, E. Yang, B. Han, Y. Yang, J. Li, Y. Mao, G. Niu, and T. Liu, "Understanding and improving early stopping for learning with noisy labels," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 24 392–24 403, Dec. 2021.

[45] Z. Ming. (2024, Dec.) Slice resource provisioning data with label. [Online]. Available: https://doi.org/10.5281/zenodo.14340019

[46] X. Zhou, J. Wu, W. Liang, K. I.-K. Wang, Z. Yan, L. T. Yang, and Q. Jin, "Reconstructed graph neural network with knowledge distillation for lightweight anomaly detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 11 817–11 828, Apr. 2024.

[47] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical autoencoder based anomaly detection by using vector reconstruction error," *Cybersecurity*, vol. 6, no. 1, p. 1, Jan. 2023.

[48] K. Dashdondov and M.-H. Kim, "Mahalanobis distance based multivariate outlier detection to improve performance of hypertension prediction," *Neural Processing Letters*, vol. 55, no. 1, pp. 265–277, Nov. 2023.

[49] Q. Li, Y. Liu, Z. Liu, P. Zhang, and C. Pang, "Efficient forwarding anomaly detection in software-defined networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 11, pp. 2676–2690, Nov. 2021.
[50] S. Liu, "An evaluation of DDPG, TD3, SAC, and PPO: deep reinforcement learning algorithms for controlling continuous system," in *Proc. International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI)*, Feb. 2024, pp. 15–24.
[51] X. Li, T. Zhang, J. Wang, Z. Han, J. Liu, J. Kang, D. Niyato, and A. Jamalipour, "Achieving network resilience through graph neural network-enabled deep reinforcement learning," *IEEE Netw.*, Feb. 2025.
[52] W. Wei, L. Fu, H. Gu, Y. Zhang, T. Zou, C. Wang, and N. Wang, "GRL-PS: Graph embedding-based DRL approach for adaptive path selection," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 2639–2651, Sept. 2023.
[53] Z. Liu, Z. Li, Y. Gong, and Y.-C. Wu, "RIS-Aided cooperative mobile edge computing: Computation efficiency maximization via joint uplink and downlink resource allocation," *IEEE Trans. Wirel. Commun.*, pp. 11 535–11 550, Sept. 2024.

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

Prof. Tarik Taleb 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 (RUB), Germany, where he leads research activities on next-generation mobile and distributed systems. Prior to joining RUB, he was a Professor at the University of Oulu, Finland (2018–2023), and an Associate Professor at Aalto University, Finland (2014–2021). Earlier in his career, he served as a Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd., Heidelberg, Germany, where he contributed to the evolution of mobile network architectures and standardization activities. Before joining NEC, he was an Assistant Professor at the Graduate School of Information Sciences, Tohoku University, Japan, working in a research laboratory fully funded by KDDI. He also held a Research Fellowship at the Intelligent Cosmos Research Institute, Japan, from 2005 to 2006. Prof. Taleb is widely recognized for his pioneering contributions to mobile network softwarization, network slicing, cloud-edge continuum management, and autonomous networking. His current research interests include autonomous network and service management, edge-cloud continuum systems, network softwarization and slicing, software-defined security, and AI-native communication networks.