Adaptive Resource Management for Edge Network Slicing using Incremental Multi-Agent Deep Reinforcement Learning

Haiyuan Li, Yuelin Liu, Xueqing Zhou,
Xenofon Vasilakos, Reza Nejabati, Shuangyi Yan, and Dimitra Simeonidou

***Abstract*—Multi-access edge computing provides local resources in mobile networks as the essential means for meeting the demands of emerging ultra-reliable low-latency communications. At the edge, dynamic computing requests require advanced resource management for adaptive network slicing, including resource allocations, function scaling and load balancing to utilize only the necessary resources in resource-constraint networks. Recent solutions are designed for a *static* number of slices. Therefore, the painful process of optimization is required again with any update on the number of slices. In addition, these solutions intend to maximize instant rewards, neglecting long-term resource scheduling. Unlike these efforts, we propose an algorithmic approach based on multi-agent deep deterministic policy gradient (MADDPG) for optimizing resource management for edge network slicing. Our objective is two-fold: (i) maximizing long-term network slicing benefits in terms of delay and energy consumption, and (ii) adapting to slice number changes. Through simulations, we demonstrate that MADDPG outperforms benchmark solutions including a static slicing-based one from the literature, achieving stable and high long-term performance. Additionally, we leverage incremental learning to facilitate a *dynamic* number of edge slices, with enhanced performance compared to pre-trained base models. Remarkably, this approach yields *superior* reward performance while *saving* approximately 90% of training time costs.**

***Index Terms*—Multi-access edge computing, network slicing, incremental learning, MADDPG**

I. Introduction and Background

With the rapid development of the Internet of Things (IoT) and mobile networks, there has been an increasing demand for latency-sensitive and computing-intensive services and applications [1, 2]. In response to this, the concept of multi-access edge computing (MEC) has emerged as a prominent solution in fifth-generation (5G) networks. MEC brings network resources closer to users, decentralizing computing demands from data centers and enabling better network experiences in terms of latency and processing speeds [3, 4]. In the context of edge computing networks, network slicing has gained significant attention as an on-demand approach to provide customized services by dividing the physical edge network into multiple logical ones [5, 6].

Benefiting from proximity and localized data processing capabilities, the combination of MEC and network slicing enables the provisioning of a wide range of ultra-reliable low latency communications (URLLC) services, including real-time video processing, edge analysis, edge storage, etc. [7–10]. However, due to the dynamic nature of loads and limited resources on MECs, an effective network slicing resource management solution is required to guarantee service quality and maximize resource utilization. Numerous research has focused on the development of management strategies for network slicing, aiming to tackle the allocation of shared resources among slices on edge computing networks in satisfying diverse 5G applications. Based on employed techniques, these works can be categorized into optimization-based [11–13], game theory-based [14–17] or deep reinforcement learning (DRL)-based strategy [18–23], respectively [11, 24].

In particular, Suh, *et al.* [25] applied a deep Q learning-based algorithm that decides the resource allocation of MECs to multiple slices. However, their strategy, which employs a single agent to handle policies for multiple network slices, is severely limited by the exponentially growing action space. This approach may encounter difficulties in converging and adapting to complex networks. In comparison, Sun *et al.* [20] proposed an autonomous virtual resource-slicing framework, which dynamically reserves resources based on the traffic ratio and then refines the allocation by a single-agent DRL-based algorithm. However, in this method, the single-agent model only determines resource allocation for one network slice. The competition between slices in the same time slot is expanded into time spans of the Markov Decision Process (MDP) and is attenuated by the discount factor.

In order to accommodate excessive slice scenarios and simulate the relationship between slices, Vila *et al.* [19] designed a collaborative multi-agent DRL algorithm that allocates a DRL agent to each slice to define the capacity shares between slices. In addition, Caballero *et al.* [14] established the resource sharing model between slices as a fisher market and converged this game on a Nash equilibrium where each slice reaps the performance benefits of sharing while retaining the ability to customize their owns. However, the authors in [19] and [14]

Haiyuan Li, Yuelin Liu, Xueqing Zhou, Xenofon Vasilakos, Reza Nejabati, Shuangyi Yan, and Dimitra Simeonidou are with the High Performance Network Group, Smart Internet Lab, Faculty of Engineering, University of Bristol, BS8 1UB, U.K. (e-mail: ocean.h.li.2018@bristol.ac.uk).

Haiyuan Li and Yuelin Liu contributed to the work equally and should be regarded as co-first authors.

Manuscript received XXXX YY, ZZZZ; revised YYYYY XX, ZZZZ.

did not account for the release of unutilized resources and the scheduling of both current and *future* requests. This oversight becomes particularly significant in scenarios where the dynamics of resources introduce complexities, and over-provisioning leads to immediate high rewards but at the *cost of resource shortages for subsequent requests*.

In contrast, by formulating the accessing process in wireless networks as a Lyapunov optimization, Feng *et al.* [26] developed a dynamic network slicing and resource allocation algorithm that jointly optimize slice request admission in the long term and resource allocation between users in the short term to maximize the operator's average revenue. In addition, Huynh et al. [22] incorporated long-term returns by establishing an MDP and resolved it with a DRL-based solution that decides the accessing decisions of the requests. Furthermore, to expedite the convergence speed, they proposed a deep dueling architecture, wherein two Q-learning models separately estimate the state values and the advantage functions of actions. However, the primary focus of [26] and [22] lies in optimizing access competition and resource allocation among user requests rather than resource utilization on MEC networks.

Furthermore, although there has been extensive research on designing slice management policies to optimize the utilization of MEC resources, only a few studies have addressed the issue of network slice variations. *Changing the number of slices* would affect state variables in solutions based on Lyapunov optimization [26] and game theory [14], necessitating the redesign of function models or the identification of new game equilibrium. Similarly, algorithms based on multi-agent DRL would require retraining and are unsuitable for such scenarios.

A. Novelty & Contribution

In summary, the current literature poses *three critical gaps* seeking to be resolved by designing efficient network slice management policies and solutions tailored for edge computing networks. First, the majority of the prior algorithms have emphatically focused on instant rewards, neglecting to account for the adverse consequences of resource over-provisioning on long-term returns. Second, the possible action space explosion and slice competition management in DRL-based solutions remain unclear. Third, the research on the subject has been mostly restricted to resource management for a fixed number of network slices, thus neglecting the current and future reality of service dynamics in 5G and beyond.

To overcome these obstacles, the main contributions of this paper are summarized below in the order of problem resolution:

- We formulate an MDP optimization problem to account for the impact of previous resource management actions on future profits with the objective of maximizing the *long-term* benefits of computing latency and energy consumption of the MEC servers.
- We propose a *Multi-Agent Deep Deterministic Policy Gradient (MADDPG)* based algorithm for both resource allocation and scheduling purposes that captures the resource competition relationship among multiple network slices and reduces the action space of DRL. To the best of our knowledge, the current effort marks the first instance of integrating incremental learning into a MADDPG-based solution for long-term resource management in network slicing of edge computing networks.
- In response to the dynamic changes in slice numbers encountered in 5G, we have integrated a *novel incremental learning* scheme into the MADDPG algorithm, eliminating the need for retraining our DRL model from scratch with significant training time cost savings.

In order to assess the performance of the MADDPG, we conduct a comprehensive evaluation over an extended continuous time period against a series of three benchmarks including (i) a random allocation and (ii) an over-allocation approach, as well as (iii) a static slicing-based solution originally proposed in [14]. As performance highlights, we prove that solutions without scheduling management cannot yield reliable results as they pursue higher profits at the expense of future losses. In comparison, MADDPG-based algorithms can adapt to the diversity of resource variation and slice requests, and obtain the highest average return and the lowest variance. More importantly, with the assistance of incremental learning, MADDPG needs only 12% of training time and achieves better performance compared to training our model from scratch.

B. Outline

The remainder of this paper is organized as follows. Section II presents the network slicing scenario and formulates the optimization problem. Section III discusses the details of our proposed incremental multi-agent DRL solution. Then, Section IV provides the setup and the numerical results. Finally, Section V provides a comprehensive summary of our key findings and outlines our future research endeavors.

II. Network slicing in edge computing networks

A multi-server edge computing network example is shown in Figure 1. The primary entities within this network comprise MEC servers, links between MECs, subscribers, service providers that be accessed on certain MECs, and a network slice manager that manages the edge network in a centralized mode. In this network, service providers interact with the subscribers within their coverage, integrate the service requirements and initiate slice requests to the network slice manager. The network slice manager subsequently accepts and processes these requests at fixed intervals. Different VNFs with dedicated objectives such as edge computing, video processing, data storage or traffic routing can be flexibly combined and placed on MECs [27, 28].

Within the MEC network, the overall latency C_{it} for slice i at time slot t comprises the computing latency c_{ibt} and transmission delay c_{iat}. It can be written as

$$C_{it} = c_{iat} + c_{ibt} \tag{1}$$

As each network slice is processed on multiple MECs in parallel, the overall computing latency will be determined by

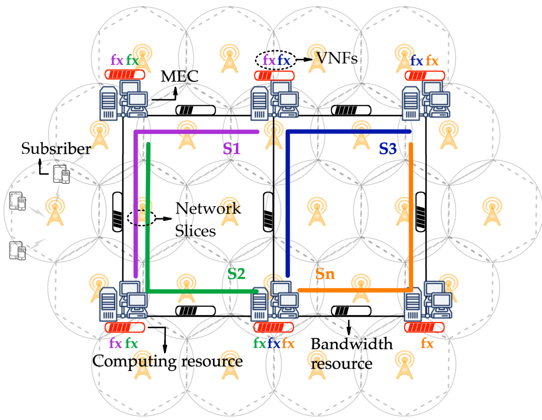


Fig. 1. Network slicing in MEC networks with constrained computing and bandwidth resources.

the maximum delay among all MECs that used by i, i.e. M_{it}. c_{ibt} can be expressed as

$$c_{ibt} = \max_m^{M_{it}} c_{imbt} \tag{2}$$

where c_{imbt} denotes the computing latency on MEC m, which can be calculated by the division of required computing resource E_{imt} for VNFs on m and allocated resource e_{imt}

$$c_{imbt} = E_{imt}/e_{imt} \tag{3}$$

In addition, based on data rate p_{ilt} on link l and workload size on l, D_{ilt}, transmission latency c_{iat} can be achieved by:

$$c_{iat} = \sum_{l}^{L_{it}} D_{ilt}/p_{ilt} \tag{4}$$

where L_{it} represents the links passed by slice i. According to [29], data rate p_{ilt} on link l can be calculated by:

$$p_{ilt} = b_{ilt} \log_2(1+N) \tag{5}$$

where N and b_{ilt} states for signal to noise ratio and allocated bandwidth on link l to slice i at time slot t [30], respectively.

The objective of this paper is to maximize long-term profits for URLLC by reducing latency, while also aiming to minimize the operational costs of network operators in terms of energy consumption of MECs. Therefore, the problem P is formulated as follows:

$$P: \max \sum_{i=1}^{I}\sum_{t=1}^{T}(C_1/C_{it} + E_1/(\sum_{m=1}^{M_{it}} f(U_{mt})\ c_{imbt})) \tag{6a}$$

$$\text{s.t.}\quad C1: 0 \le U_{mt} \le 1, \quad \forall m \in M,\ \forall t \in T \tag{6b}$$

$$C2: 0 \le e_{imt}, \quad \forall i \in I,\ \forall m \in M,\ \forall t \in T \tag{6c}$$

$$C3: 0 \le b_{ilt}, \quad \forall i \in I,\ \forall l \in L,\ \forall t \in T \tag{6d}$$

$$C4: 0 \le \textstyle\sum_i^I e_{imt} \le J,\ \forall m \in M,\ \forall t \in T \tag{6e}$$

$$C5: 0 \le \textstyle\sum_i^I b_{ilt} \le B,\ \forall l \in L,\ \forall t \in T \tag{6f}$$

where, in 6a, $\sum_{m=1}^{M_it} f(U_{mt})c_{imbt}$ sums the power consumption of multiple MECs used by slice i. U_{mt} is the resource utilization ratio of MEC m at time slot t. It is determined by both current and previous resource allocation decisions. $f(*)$ states the function of power cost and utilization rate and is measured by a testament in [31]. C_1 and E_1 represent the minimum overall latency and energy consumption. They are used to normalize the objective function and balance the weights between latency and energy consumption. In this paper, latency and energy consumption are treated as equally weighted objectives. T can be considered as an arbitrarily large time scale. Constraint $C1$ states that the utilization ratio of every MEC at any time should be kept below 1. Constraints $C2$ and $C3$ ensure all slices will be allocated with computing and bandwidth resources on their respective servers and links. Constraints $C4$ and $C5$ are

TABLE I
NOTATIONS USED THROUGHOUT THE PAPER

	Notation	Description
Index	i	Slice / agent
	m	Server
	l	Link
	t	Time slot
Parameter	C_1	Min computing time in previous 500 slots
	E_1	Min energy cost in previous 500 slots
	E_{imt}	Required computing resources of i on m at t
	D_{ilt}	Workload size of i on l at t
	M_{it}	MECs used by i at t
	L_{it}	Link passed by i at t
	I	Number of slices /agents
	T	Time scale (An arbitrarily large number)
	B	Bandwidth capacity
	J	MEC computing capacity
	N	Signal-to-noise ratio
	O	State observation of DRL agent
	γ	discount factor
	α	Learning rate
	k	Training process
	$\mathcal{O}$	Ornstein–Uhlenbeck (OU) noise
	ϵ	Ornstein–Uhlenbeck (OU) noise scale
	μ	Long-term mean of $\mathcal{O}$
	σ	Standard deviation of $\mathcal{O}$
	β	The speed of mean reversion of $\mathcal{O}$
Variable	c_{iat}	Transmission time of i at t
	c_{ibt}	Computing time of i at t
	c_{imbt}	Computing time of i on m at t
	p_{ilt}	Data rate of i on l at t
	U_{mt}	Utilization of m at t at t
	C_{it}	Overall latency of i at t
	e_{imt}	Computing resource allocation for i on m at t
	b_{ilt}	Bandwidth allocation for i on l at t
	Q	Q-value
	S	State of actor and critic
	R	Reward
	A	Action of actor i at t
	s_t	Global state at t, $s_t \in S$
	s_{it}	State of agent i at t, $s_{it} \in S$
	a_{it}	Action of agent i at t, $a_{it} \in A$
	r_t	Reward at t, $r_t \in R$
	ϕ	Parameter of critic i
	θ	Parameter of actor i
	ϕ'	Parameter of target critic i
	θ'	Parameter of target actor i
	$\mathcal{D}$	Reply buffer
	y_{it}	Target Q value
	π	Policy of actor
	d_{it}	Policy of agent i at t

designed to prevent the allocation of resources to network slices from exceeding the capacity of the servers and link. J and B denote the MEC and Link capacity reserved for URLLC. In general, problem P can be interpreted as optimizing e_{imt} and b_{ilt}, $\forall i, m, l, t$, through the design of resource allocation and scheduling strategies, with the aim of minimizing computing latency and energy consumption. This problem presents the non-convex property and is hard to be resolved by conventional methods. For ease of reference, important notations used throughout this paper are summarized in Table I.

III. Incremental Multi-agent DRL-based strategy

To optimize long-term network slicing resource management, we consider P as an MDP optimization defined by a tuple $K = (S, A, R, \gamma)$, in which S, A, R, γ represent state space, action space, reward and discount factor (allowing to control the influence of future rewards). We adopt MADDPG to resolve the problem and handle competitive and cooperative multi-slice environments by allocating a DRL agent to each slice.

In addition, we incorporate incremental learning into the algorithm to further facilitate the performance of MADDPG in edge computing networks. With the assistance of incremental learning, MADDPG can quickly adapt to changes in the number of agents without needing to be retrained, thus effectively solving the optimization of dynamically changing network slices in 5G applications. In addition, incremental MADDPG can retain the previously acquired knowledge and continuously update the model to achieve better optimization performance.

MADDPG extends the Deep Deterministic Policy Gradient (DDPG) framework to a multi-agent setting [32] and is constructed based on the actor-critic paradigm. In previous works, the state information for each actor-critic pair comprises the associated local observations, and each critic's input includes the actions of all actors, enabling it to offer more precise evaluations for the corresponding actor [33, 34]. This methodology is recognized as the centralized training and distributed execution framework [32]. However, within this framework, the critic's input is influenced by fluctuations in the number of agents, which renders it *unsuitable* for incremental learning in environments with variable slices. Therefore, as shown in Figure 2, instead of sharing all actions between critics, we adopt an alternative by providing each critic with global information, including remaining resources and request details over the network. This modification retains the benefits of the centralized learning approach of empowering critics with the global view and keeping coordinated decision-making property between agents and also overcomes the limitation of integrating incremental learning technologies into MADDPG when dealing with dynamic environments and changing agent quantities. Specifically, the key elements of MDP are summarized as follows:

State of Actor and Critic S**:** Each critic has a global view, accessing the resource information and the requests of the entire network. the state of a critic can be written as s_t

$$s_t = [J_{Mt}, B_{Lt}, D_{It}, E_{It}] \tag{7}$$

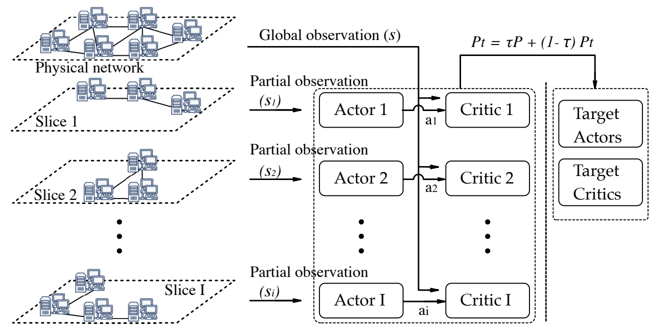


Fig. 2. Actor-critic structure of the proposed MADDPG with incremental learning.

where J_{Mt} and B_{Lt} stand for remaining resource of all MECs and links over the network at time slot t, respectively. D_{It} and E_{It} represent computing resource requirement and workload size of all network slices at time slot t. The actor, on the other hand, has a partial observation of the state s_{it}, which includes remaining resource information of its utilized MECs and links, as well as the request information of the network slice.

$$s_{it} = [J_{mt}, B_{lt}, D_{it}, E_{it}] \quad \forall m \in M_{it}, \forall l \in L_{it} \tag{8}$$

For MEC without request, related parameters are set to 0. In addition, it it worth noticing that the "time slot" concept in DRL refers to a discrete decision point in the modelled environment. To resolve P, it is consistent with the time interval for processing requests by the network slice operator.

Action of Actor A**:** Action a_{it} can be represents as follows

$$a_{it} = [a_{itm} * 2/5J, \forall m \in M_{it}; a_{itl} * 2/5B, \forall l \in L_{it}] \tag{9}$$

a_{itm} and a_{itl} denote the allocated computing to VNFs and bandwidth resources between MECs in network slice i, respectively. They can be written as

$$a_{itm} = clip(a_{itm} + \epsilon\mathcal{O}(\mu, \sigma, \beta), 0, 1) \tag{10}$$

$$a_{itl} = clip(a_{itl} + \epsilon\mathcal{O}(\mu, \sigma, \beta), 0, 1) \tag{11}$$

where $\mathcal{O}$ is the Ornstein–Uhlenbeck (OU) noise and ϵ is the noise scale. μ, σ and β are the long-term mean, standard deviation and speed of mean reversion of $\mathcal{O}$. The activation function of output is set to $sigmod$ with an output range between $0 \sim 1$, aligning with constraints (6c)–(6f). $2/5J$ and $2/5B$ are used to constrain the maximum resources that can be allocated to each slice.

Reward R**:** Consistent with the objective function of formula (6a), the reward generated by each actor is set to

$$r_{it} = (C_1/C_{it} + E_1/(\sum_{m=1}^{M_{it}} f(U_{mt})\ c_{imbt}))/2I \tag{12}$$

C_1 and E_1 are achieved by a dynamic slicing window that records C_i and $\sum_{m=1}^{M_i} f(U_{mt}) c_{imb}$ in previous 500 time slots. If any MEC or link used by slice i runs out of resources, this implies that the request cannot be served by the network. In such cases, the reward of this agent is

$$r_{it} = -1/I \tag{13}$$

with a *penalty* aimed at improving the performance of resource scheduling and preventing the over-allocation of resources that could negatively impact the rewards of subsequent requests. By aggregating the rewards from all agents, we employ a shared reward approach for DRL training, commonly referred to as the "fully-cooperative" model [32]. Therefore,

$$r_t = \sum_{i}^{I} r_{it} \tag{14}$$

Last, notice I in the denominator of Equations 12 and 13. Its role is to normalize r_t.

Based on these elements, MADDPG operates by optimizing the neural networks of both actors and critics so as to emulate the optimal policy and value functions, respectively. In terms of the critic, the optimization unfolds by minimizing the temporal-difference (TD) error between the current and the target Q-value. The target Q-value y_{it} can be calculated by

$$y_{it} = r_{it} + \gamma Q_{\phi_i}\left(s_{t+1}, a'_{1t}, \ldots, a'_{It}\right) \tag{15}$$

where $a'_{i,t}$ is the action selected by the target actor network for agent i at time t. ϕ_i represents the parameters of the critic network for agent i to estimate the Q-value Q_{ϕ_i}. Therefore, the loss function for the critic update is

$$V\left(\phi_i\right) = \mathbb{E}_{s_t, a_{it}, r_{it}, s_{t+1} \sim \mathcal{D}}\left(\left(Q_{\phi_i}\left(s_t, a_{1t}, \ldots, a_{It}\right) - y_{it}\right)^2\right) \tag{16}$$

The optimization of the critic network is carried out using gradient descent, specifically, by updating the parameters ϕ_i to minimize this loss function. The gradient descent can be represented as

$$\phi_i \leftarrow \phi_i - \alpha \nabla_{\phi_i} V(\phi_i) \tag{17}$$

Here, α represents the learning rate, which controls how much the parameters are updated at each step, and $\nabla_{\phi_i} V(\phi_i)$ is the gradient of the loss function with respect to the parameters ϕ_i. In practice, the gradient $\nabla_{\phi_i} V(\phi_i)$ is estimated by sampling a batch of experiences from the replay buffer $\mathcal{D}$ and averaging the gradients for these experiences.

To maximize the expected return, the actor update step focuses on finding an optimal policy. Policy gradient methods are utilized for this purpose, specifically by performing gradient ascent on the expected return derived from the critic network with respect to the actor's parameters. The gradient of the expected return J and update rule for actor's parameter θ_i can be expressed as follows

$$\nabla_{\theta_i} J\left(\theta_i\right) = \mathbb{E}_{s_t \sim \mathcal{D}}\left(\nabla_{\theta_i} Q_{\phi_i}\left(s_t, a_{1t}, \ldots, a_{It}\right)\big|_{a_{jt} = \pi(s_t; \theta_j)}\right) \tag{18}$$

$$\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} J\left(\theta_i\right) \tag{19}$$

where π refers to the policy of an actor agent and j stands for all the agent in I.

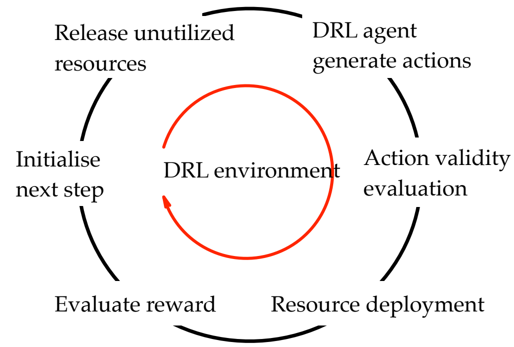


Fig. 3. Iteration of Dynamic DRL Environment in MEC Network Resource Management

Algorithm 1 Multi-agent Deep Deterministic Policy Gradient (MADDPG) Algorithm

1: Initialize replay memory $\mathcal{D}$
2: Initialize the actor, target actor, critic and target critic with parameter $\theta_{1\sim I}, \theta'_{1\sim I}, \phi_{1\sim I}, \phi'_{1\sim I}$
3: Initialize OU noise for action exploration
4: Initialise time step t and network resource state
5: Initialise observation step k_1, exploration step k_2, training step k_3
6: **while** $t < k_1$ **do**
7: Get critic state s_t and actor state $s_{1\sim I,t}$
8: Estimate $a_{1\sim I,t}$ by $\pi(s_{1\sim I,t}; \theta_{1\sim I,t})$
9: Execute $a_{1\sim I,t}$, get reward $r_{1\sim I,t}$, next critic state $s_{n+1,t}$ and actor state $s_{n+1,i\sim I,t}$
10: Store transition in $\mathcal{D}$
11: $t = t + 1$
12: **end while**
13: Initialize $f = 1$
14: **while** $k_1 < t < k_1 + k_2$ **do**
15: Repeat code $7 - 10$
16: Get a batch from $\mathcal{D}$
17: **for** m in batch **do**
18: Compute TD target $y(i)$ using 15
19: **end for**
20: Update critic network based on 17
21: Update actor network based on 19
22: Update target networks using 20 and 21
23: $\epsilon = (k_2 - f)/k_2$, $t = t + 1$, $f = f + 1$
24: **end while**
25: **while** $k_1 + k_2 < t < k_1 + k_2 + k_3$ **do**
26: Repeat code $15 - 22$
27: $t = t + 1$
28: **end while**

In addition, MADDPG incorporates a set of target networks to enhance the stability of learning. These target networks serve as duplicates of the actor and critic networks, however, their parameters are updated gradually using a soft update strategy as shown in Equation 20 and 21. The updating process involves smoothly blending the parameters of the target networks with those of the main networks.

$$\theta'_i \leftarrow \tau \theta_i + (1 - \tau)\theta'_i \tag{20}$$

$$\phi'_i \leftarrow \tau \phi_i + (1 - \tau)\phi'_i \tag{21}$$

where $\tau \ll 1$ is a small factor controlling the rate of the update.

Figure 3 illustrates the iteration of dynamic DRL environment in MEC network resource management. In the beginning of each DRL step, all resources occupied by completed requests are first released back to the network. Then the DRL decides on actions based on the latest network resource information and request details. Subsequently, in consistency with Equations (6e) and (6f), the action gets verified for its network execution feasibility, i.e. the resource allocation cannot exceed the remaining resources. Finally, the environment generates corresponding rewards for the action based on the delay and energy consumption in processing the slice. The pseudo-code of MADDPG is summarized in Algorithm 1. As shown in algorithm, the training of MADDPG consists of three stages. These include (i) the observation stage, where actions are chosen randomly to populate the experience replay buffer; (ii) the exploration stage, where actions are selected by the models with added noise that decreases over time, allowing the model to explore different outcomes and update its understanding of the environment; and finally (iii) the training stage, where the models are further refined using noise-free training based on the experiences stored in the buffer.

An incremental learning approach [35] was devised to accommodate varying numbers of agents in MADDPG. Inspired by federated learning and transfer learning [36–38], this approach involves averaging the parameters of the existing agents' models using formulas (22) and (23), resulting in generalized models

$$\phi_g = \sum_i^I \phi_i / I \tag{22}$$

$$\theta_g = \sum_i^I \theta_i / I \tag{23}$$

When the number of agents increases, the generalized model is loaded into the new agent while preserving the models of the existing agents. In the opposite case where the number of agents decreases (i.e., decremental learning), the newly derived model is assigned to all agents. This method ensures that the previously acquired features are retained in the neural networks, requiring only minor additional training to continually update and enhance the model by incorporating new data.

IV. Performance Evaluation

A. Simulation setup

The experimental scenario is set as follows:

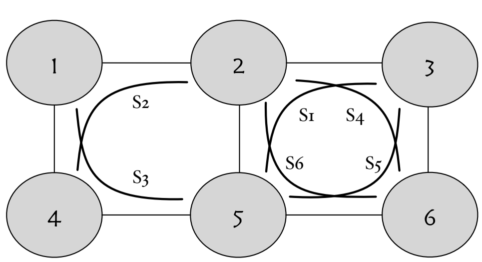


Fig. 4. Simulated edge computing network setup.

TABLE II
SUMMARY OF NEURAL NETWORK CONFIGURATIONS IN MADDPG.

	Layer	Shape	Activation
Actor	Input	5 (Request info) + 5 (Resource of MECs and links)	-
	Fully connected	32	*ReLU*
	Fully connected	32	*ReLU*
	Output	5	*Sigmod*
Critic	Input	6 (All MEC resource) + 7 (All link resource) + All request information (5 * 6)	-
	Fully connected	64	*ReLU*
	Fully connected	64	*ReLU*
	Output	1	*Tanh*

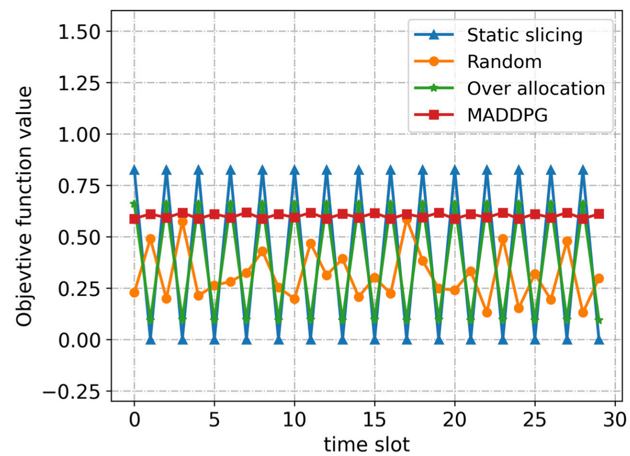
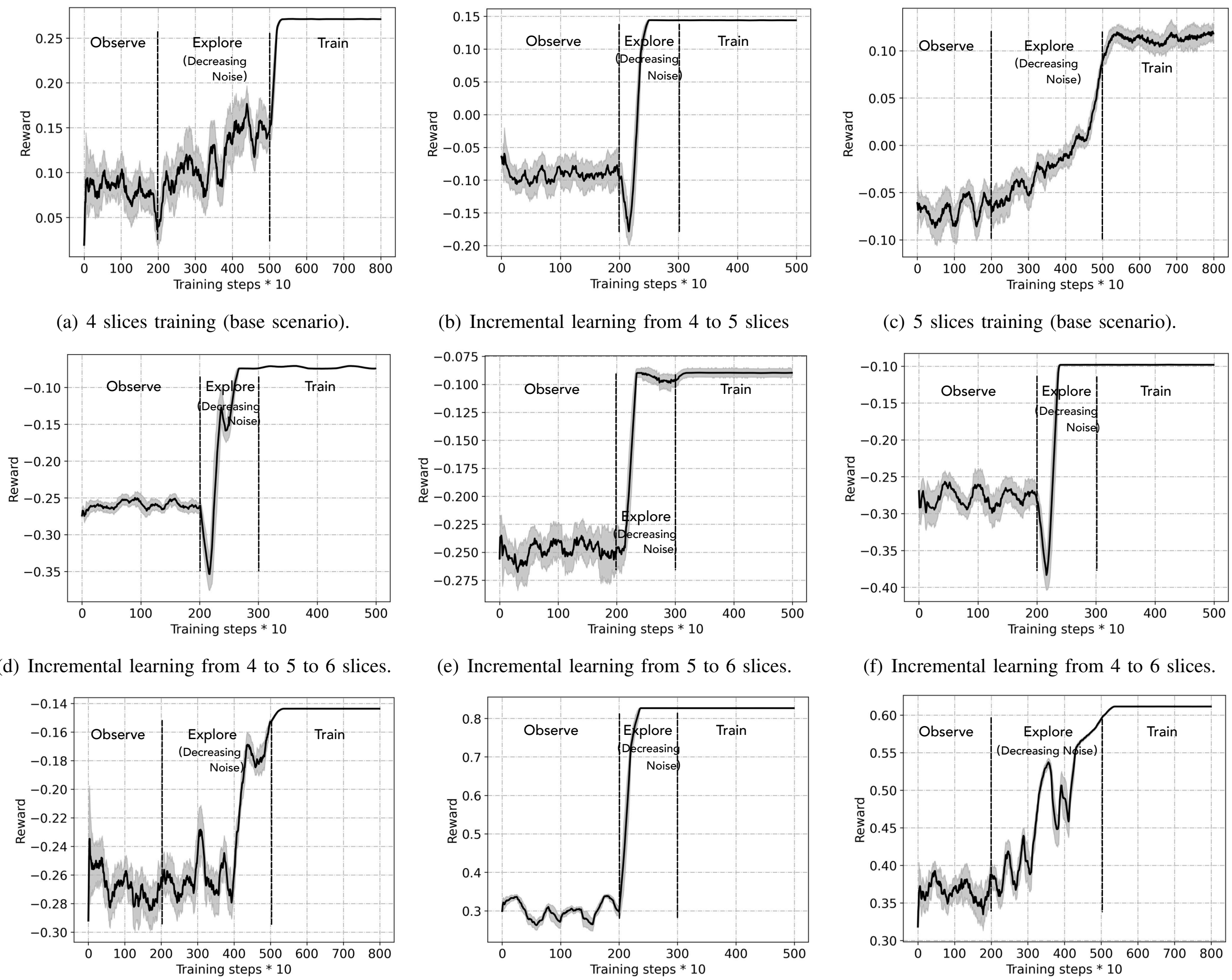


Fig. 5. Objective function values for various resource management solutions in a 4-slice edge networking scenario spanning 30-time slots.

- In the MEC network, there are 6 servers with the same amount of computing resources, 30 Intel i7-1195G7 CPU modules with a maximum 150GHz processing capability. The computing capacity J and bandwidth capacity B reserved for URLLC and the noise-to-signal ratio of each link are set to 100 GHz, 10 Gbps and 10 dB, respectively. The topology of this network is shown in Figure 4. Each network slice consists of three MECs hosting VNFs for computing purposes. Additionally, the processing interval (i.e. time slot length) is set to 1s. Within each slot, the bandwidth requirement for requesting slices has been set to 1 - 2 Gb, while the computational demand for these slices on MEC has been specified as 10 - 20 Gcycles.
- In the DRL part, batch sizes in the cases of training from scratch solution and incremental learning solution are set to 300 and 200, respectively. The learning rate is set to 0.001. The target network update coefficient τ of MADDPG and the discount factor γ of DRLs are set to 0.1 and 0.99, respectively. The neural network components of MADDPG algorithm are summarized in Table II. In addition, the initial scale ϵ, long-term mean μ, standard deviation σ and the speed of mean reversion β of noise $\mathcal{O}$ are set to 1, 0, 0.1 and 0.9, respectively.



(a) 4 slices training (base scenario).
(b) Incremental learning from 4 to 5 slices
(c) 5 slices training (base scenario).
(d) Incremental learning from 4 to 5 to 6 slices.
(e) Incremental learning from 5 to 6 slices.
(f) Incremental learning from 4 to 6 slices.
(g) 6 slices training (base scenario).
(h) Decremental learning from 4 to 3 slices.
(i) 3 slices training (base scenario).

Fig. 6. Performance evaluation of the proposed MADDPG with incremental learning. Notice the 95-percentile confidence intervals marked over the mean performance curves, and the differences in Y-axes scale between the different graphs to accommodate results' readability.

TABLE III
SUMMARY OF FIGURE 5.

	Random	**Over allocation**	**MADDPG**	**Static slicing**
Maximum	0.5855	0.6597	0.6175	0.8249
Minimum	0.1306	0.0928	0.5864	0
Average	0.3115	0.3771	0.6015	0.4124
Variance	0.0161	0.0825	0.0002	0.1759

B. Long-term performance of MADDPG

Figure 5 compares the objective function value of various network slice management solutions in a 4 network slice scenario (S_1 - S_4) over a 30 time slot span. In specific, random allocation is to distribute network resources randomly between network slices. Over allocation is to allocate $2/5B$ and $2/5J$ to each network slice at each time slot. In addition, a static slicing-based solution proposed by [14] that refers to a complete partitioning of resources based on the network shares. The key information is summarized in Table III.

As can be seen in Figure 5 and Table III, random allocation under performs, averaging a mere utility of 0.3115. In comparison, static slicing and over allocation exhibit superior average utility. However, the higher instant returns come at the expense of resource shortages in the subsequent time slots and resulting lowest minimum utility. This is evidenced by their highest variances of 0.1759 and 0.0825, respectively. Such fluctuations in slice management strategies cannot meet the stability requirements in 5G networks. Therefore, compared to the rest solutions, MADDPG not only achieves the highest average value of 0.6015 but also maintains relatively low fluctuations, demonstrating its effectiveness in solving resource allocation and scheduling in the proposed optimization problem.

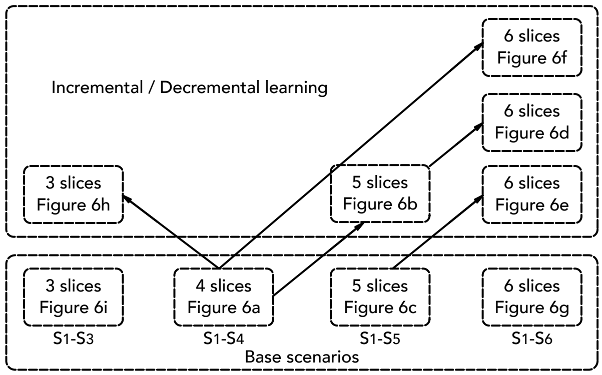


Fig. 7. Dependency map of evaluation scenarios portrayed in Figure 6. Notice the arrows. They denote a slice increment or decrement transition between learning scenarios. Base scenarios appear at the bottom of the figure, while Incremental/Decremental learning scenarios on the top.

C. Performance of incremental learning

To validate the effectiveness of incremental learning, we demonstrated the training process of both incremental learning and training from scratch under various numbers of agents. In detail, the results include *base scenarios*, i.e. scenarios that imply training from scratch with 3, 4, 5, and 6 slices without a prior model increment or decrement. They also include scenarios based on incremental learning from 4 slices through 5, and eventually to 6 slices, as well as increasing from 4 to 6 slices directly. Finally, we tested the performance when the number of agents *decreased* from 4 to 3 (i.e., a case of *decremental* learning). The relevant results are shown in Figure 6, while Figure 7 provides a dependency mapping for learning scenario transitions in Figure 6.

Firstly, a comparison between the Graphs of Figure 6(b) and 6(c), and those of Figure 6(d), 6(e), 6(f) and 6(g) reveals that regardless of employing a "step-by-step" or "skipping-step" strategy, incremental MADDPG necessitates *only* approximately *12%* of the training steps relatively to base scenarios. This implies a big leap of almost 90% in training time savings[1].

Furthermore, the post-convergence average reward in Graph 6(b) outperforms that in Graph 6(c) by 0.03, and similarly, the mean reward following convergence in Graph 6(d), 6(e) and 6(f) exceeds that in Graph 6(g) by 0.05. Therefore, besides the faster convergence, the latter suggests that the previously trained model retains and exploits the learnt network structure and information, hence allowing for improving network performance.

Moreover, the *drastic reward reduction* observed in the graphs of Figure 6(b), 6(d) and 6(f) after the end of the observation period may be attributed to the previously unseen training data from networks assuming a different number of slices. Nevertheless, the incremental MADDPG demonstrates an impressive responsiveness and capability of adapting to changes in the new network and rebounding the reward within 120 steps. Finally, Graphs 6(h) and 6(i) illustrate the performance of *decremental* learning. Due to the minimal resource competition when assuming three slices, MADDPG with decremental learning can *avoid* a reward reduction as a penalty and achieve an average reward closer to 1.

It is worth noting that the reward gradually decreases with the number of slices. This is due to the limited resources struggling to meet the demands of a growing number of slices. Besides this, an increased number of slices may increase competition for resources significantly, thus making resource starvation likely to happen and eventually a slice service failure or outage to happen. The latter likelihood is captured and expressed by the exhibited negative reward values.

[1]As a positive side implication, the reduction in training time achieved by our DRL solution leads to a *significant decrease in energy* consumption for *maintaining the RL* model itself when incrementing or decrementing the number of slices. Existing literature has both theoretically [39] and empirically [40] (specific to GPU or CPU architectures) demonstrated a proportional relationship between training time and energy consumption costs.

V. Conclusion and Future Work

We investigate the resource allocation and scheduling of network slices in edge computing networks to provide service for emerging computing-intensive, latency-sensitive URLLC applications. The absence of effective resource management in network slicing can lead to service failures and performance fluctuations. To mitigate these issues, our approach aims to minimize both latency and energy consumption by establishing a mathematical model that captures the resource-sharing relationships among network slices. The problem is then formulated as an MINLP and addressed using a MADDPG-based solution. Furthermore, we augment the MADDPG algorithm with incremental learning features that enable to capture network slice dynamics in terms of the number of slices and to reduce training costs. Simulation results against benchmark solutions indicate that MADDPG can gain great improvements compared to other methods in the literature that focus on instant profits, and *incremental learning* can capture varying slice number dynamics, *achieving better* returns with lower time complexity compared to pre-trained base models.

Regarding future work, there is room to improve the performance of incremental MADDPG against dynamic changes in the 5G network structure. The extension of MECs with more nodes and/or types of resources can disrupt existing management strategies. In light of these challenges, we plan to investigate the development of *generalized models* that can effectively accommodate *varying network structures* and *diverse types* of network slices beyond URLLC. This forward-looking approach seeks to ensure the adaptability and scalability of MADDPG-based solutions for optimal resource allocation and scheduling in complex and evolving 5G and beyond 5G environments.

Acknowledgments

The authors would like to express their gratitude for the support from the China Scholarship Council and the UK-funded project REASON under the Future Open Networks Research Challenge sponsored DSIT.

References

[1] A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane, "Resource provisioning in edge computing for latency-sensitive applications," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 088–11 099, 2021.

[2] G. Premsankar and B. Ghaddar, "Energy-efficient service placement for latency-sensitive applications in edge computing," *IEEE internet of things journal*, vol. 9, no. 18, pp. 17 926–17 937, 2022.

[3] G. Sriram, "Edge computing vs. cloud computing: an overview of big data challenges and opportunities for large enterprises," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 1, pp. 1331–1337, 2022.

[4] H. Li, K. D. R. Assis, S. Yan, and D. Simeonidou, "Drl-based long-term resource planning for task offloading policies in multiserver edge computing networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4151–4164, 2022.

[5] X. Zhao, M. Liu, and M. Li, "Task offloading strategy and scheduling optimization for internet of vehicles based on deep reinforcement learning," *Ad Hoc Networks*, vol. 147, p. 103193, 2023.

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

[7] A. Thantharate and C. Beard, "Adaptive6g: Adaptive resource management for network slicing architectures in current 5g and future 6g systems," *Journal of Network and Systems Management*, vol. 31, no. 1, p. 9, 2023.

[8] J. Jin, R. Li, X. Yang, M. Jin, and F. Hu, "A network slicing algorithm for cloud-edge collaboration hybrid computing in 5g and beyond networks," *Computers and Electrical Engineering*, vol. 109, p. 108750, 2023.

[9] A. Domeke, B. Cimoli, and I. T. Monroy, "Integration of network slicing and machine learning into edge networks for low-latency services in 5g and beyond systems," *Applied Sciences*, vol. 12, no. 13, p. 6617, 2022.

[10] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[11] S. D'Oro, L. Bonati, F. Restuccia, M. Polese, M. Zorzi, and T. Melodia, "Sl-edge: Network slicing at the edge," in *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2020, pp. 1–10.

[12] Q. Liu and T. Han, "Direct: Distributed cross-domain resource orchestration in cellular edge computing," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 181–190.

[13] T. Ma, Y. Zhang, F. Wang, D. Wang, and D. Guo, "Slicing resource allocation for embb and urllc in 5g ran," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–11, 2020.

[14] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Pérez, "Network slicing games: Enabling customization in multi-tenant mobile networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 662–675, 2019.

[15] M. Datar and E. Altman, "Strategic resource management in 5g network slicing," in *2021 33th International Teletraffic Congress (ITC-33)*. IEEE, 2021, pp. 1–9.

[16] T. D. Tran and L. B. Le, "Resource allocation for multi-tenant network slicing: A multi-leader multi-follower stackelberg game approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8886–8899, 2020.

[17] F. Fossati, S. Moretti, P. Perny, and S. Secci, "Multi-resource allocation for network slicing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1311–1324, 2020.

[18] J. A. Hurtado Sánchez, K. Casilimas, and O. M. Caicedo Rendon, "Deep reinforcement learning for resource management on network slicing: A survey," *Sensors*, vol. 22, no. 8, p. 3031, 2022.

[19] I. Vilà, J. Pérez-Romero, O. Sallent, and A. Umbert, "A novel approach for dynamic capacity sharing in multi-tenant scenarios," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2020, pp. 1–6.

[20] G. Sun, Z. T. Gebrekidan, G. O. Boateng, D. Ayepah-Mensah, and W. Jiang, "Dynamic reservation and deep reinforcement learning based autonomous resource slicing for virtualized radio access networks," *Ieee Access*, vol. 7, pp. 45 758–45 772, 2019.

[21] W. Wu, J. Dong, Y. Sun, and F. R. Yu, "Heterogeneous markov decision process model for joint resource allocation and task scheduling in network slicing enabled internet of vehicles," *IEEE Wireless Communications Letters*, vol. 11, no. 6, pp. 1118–1122, 2022.

[22] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and fast real-time resource slicing with deep dueling neural networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[23] C. Qi, Y. Hua, R. Li, Z. Zhao, and H. Zhang, "Deep reinforcement learning with discrete normalized advantage functions for resource management in network slicing," *IEEE Communications Letters*, vol. 23, no. 8, pp. 1337–1341, 2019.

[24] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource allocation for network slicing in 5g telecommunication networks: A survey of principles and models," *IEEE Network*, vol. 33, no. 6, pp. 172–179, 2019.

[25] K. Suh, S. Kim, Y. Ahn, S. Kim, H. Ju, and B. Shim, "Deep reinforcement learning-based network slicing for beyond 5g," *IEEE Access*, vol. 10, pp. 7384–7395, 2022.

[26] J. Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu, "Dynamic network slicing and resource allocation in mobile edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7863–7878, 2020.

[27] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5g: Challenges and opportunities," *IEEE Internet Computing*, vol. 21, no. 5, pp. 20–27, 2017.

[28] N. Alliance, "Description of network slicing concept," *NGMN 5G P*, vol. 1, no. 1, pp. 1–11, 2016.

[29] M. Schwartz, *Mobile wireless communications*. IET, 2005, vol. 25.

[30] N. Kiran, X. Liu, S. Wang, and C. Yin, "Vnf placement and resource allocation in sdn/nfv-enabled mec networks," in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2020, pp. 1–6.

[31] R. Rahmani, I. Moser, and M. Seyedmahmoudian, "A complete model for modular simulation of data centre power load," *arXiv preprint arXiv:1804.00703*, 2018.

[32] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2022.

[33] J. Tian, Q. Liu, H. Zhang, and D. Wu, "Multiagent deep-reinforcement-learning-based resource allocation for heterogeneous qos guarantees for vehicular networks," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1683–1695, 2021.

[34] Z. Cheng, M. Min, Z. Gao, and L. Huang, "Joint task offloading and resource allocation for mobile edge computing in ultra-dense network," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[35] Y. Luo, L. Yin, W. Bai, and K. Mao, "An appraisal of incremental learning methods," *Entropy*, vol. 22, no. 11, p. 1190, 2020.

[36] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[37] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[38] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[39] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *arXiv preprint arXiv:1906.02243*, 2019.

[40] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

PLACE PHOTO HERE

Haiyuan Li received the B.Sc degree in communication engineering in Central South University, China, in 2019, and the M.Sc degree in communication networks and signal processing from University of Bristol, U.K., in 2020. He is currently pursuing his Ph.D. degree with the school of electrical and electronic engineering in University of Bristol. His research interests include Mobile Edge Computing, Deep Reinforcement Learning, Parallel Computing, Game Theory, Beyond 5G and Network Optimization.

PLACE PHOTO HERE

Yuelin Liu is currently pursuing PhD degree at University of Bristol. He received his BSc degree in Electronic Information Engineering from Harbin Institute of Technology, China, in 2021 and his MSc degree in Optical Communications and Signal Processing from the University of Bristol, UK, in 2022. His research interests include Deep Reinforcement Learning and Edge Computing in the context of 5G and future 6G network architectures.

PLACE PHOTO HERE

Xueqing Zhou is a PhD student at the Smart Internet Lab, University of Bristol. She received her B.S. degree in Communication Engineering from the Jilin University in 2018, and her M.S. degrees in Communication Network and Signal Processing from University of Bristol in 2019. Her research interests encompass deep reinforcement learning, multi-access networks, and network optimization.

PLACE PHOTO HERE

Xenofon Vasilakos is a Lecturer in AI for Digital Infrastructures at the University of Bristol, affiliated with the Bristol Digital Futures Institute (BDFI) and Smart Internet Lab. He holds an MSc degree in Parallel and Distributed Computer Systems from VU Amsterdam and a PhD degree in informatics from AUEB Athens. He has actively contributed to several EU, national, and industry-funded research projects. His current research focuses on natively intelligent 6G architectures, emphasizing Zero-touch networking. His work aims to seamlessly integrate artificial intelligence into software-defined communication networks, advancing the efficiency and automation of network operations for future digital infrastructures.

PLACE PHOTO HERE

Reza Nejabati is currently a chair professor of intelligent networks and head of the High-Performance Network Group in the Department of Electrical and Electronic Engineering in the University of Bristol, UK. He is also a visiting professor and Cisco chair in the Cisco centre for Intent Based Networking in the Curtin University, Australia. He has established successful and internationally recognised experimental research activities in “Autonomous and Quantum Networks”. Building on his research, He co-founded a successful start-up company (Zeetta Networks Ltd) with 25 employees and £6m funding.

PLACE PHOTO HERE

Shuangyi Yan is a Senior Lecturer in the High Performance Networks Group in the Smart Internet Lab at the University of Bristol. He received the B.E degree in information engineering from Tianjin University, Tianjin, China in 2004. In 2009, he got the PhD degree in Optical Engineering from Xi’an Institute of Optics and Precision Mechanics, CAS, Xi’an, China. From 2011 to 2013, Dr Yan worked in the Hong Kong Polytechnic University, Hong Kong, as a postdoctoral researcher, investigating on the spectra-efficient long-haul optical transmission system and low-cost short-range transmission system. In July 2013, he joined the University of Bristol. His research focuses on machine-learning applications in dynamic optical networks and 5G Beyond networks, programmable optical networks, and data centre networks.

PLACE PHOTO HERE

Dimitra Simeonidou (FREng, FIEEE) is a Full Professor at the University of Bristol, the Co-Director of the Bristol Digital Futures Institute, and the Director of the Smart Internet Lab. Her research focuses in the fields of high performance networks, programmable networks, wireless-optical convergence, 5G/B5G and smart city infrastructures. She is increasingly working with Social Sciences on topics of digital transformation for society and businesses. Dimitra has also been the Technical Architect and the CTO of the Smart City project Bristol Is Open. She is currently leading the Bristol City/Region 5G urban pilots. She is the author and co-author of over 500 publications, numerous patents and several major contributions to standards. Dimitra is a Fellow of the Royal Academy of Engineering, a Fellow of the Institute of Electrical and Electronic Engineers, and a Royal Society Wolfson Scholar.