Received 31 December 2023; accepted 29 January 2024. Date of publication 7 February 2024; date of current version 26 February 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3363132

AoI-Aware Energy-Efficient SFC in UAV-Aided Smart Agriculture Using Asynchronous Federated Learning

MOHAMMAD AKBARI[1], AISHA SYED[2] (Member, IEEE), W. SEAN KENNEDY[2] (Member, IEEE), AND MELIKE EROL-KANTARCI[1] (Senior Member, IEEE)

[1]School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada

[2]Nokia Bell Labs, Ottawa, ON K2K 2T6, Canada

CORRESPONDING AUTHOR: M. EROL-KANTARCI (e-mail: melike.erolkantarci@uottawa.ca)

This work was supported in part by MITACS Canada and in part by Nokia Bell labs.

ABSTRACT In the midst of rising global population and environmental challenges, smart agriculture emerges as a vital solution by integrating advanced technologies to optimize agricultural practices. Through data-driven insights and automation, it tackles the necessity for sustainable resource management, enhancing productivity and resilience in the face of complex food security and ecological concerns. The prospects of utilizing the Internet of Things (IoT) for smart agriculture are tremendous, where many IoT devices can be deployed for local environment monitoring, precision farming, autonomous irrigation, and, soil management. In some use cases like smart monitoring and agrochemical applications, UAV-enabled mobile-edge computing (MEC) is proposed as an enabler to provide IoT nodes with additional resources by hosting their computation functions. From the implementation perspective, to flexibly manage the computation functions in UAVs and/or MEC server, the emerging network function virtualization (NFV) can be utilized. However, efficient orchestration of the virtualized functions would be a challenge. In this paper, we consider a decentralized UAV-aided MEC system for smart agricultural applications in which the processing nodes benefit from the NFV technology. We aim to propose a method for efficiently orchestrating the NFVs while some important metrics are minimized, i.e., the age of information (AoI) and total network energy consumption. Especially, we consider the case in which the network state is not fully observable to the orchestrator or the observations are exposed to uncertainties. Consequently, the problem is formulated as a decentralized partially observable Markov decision process (DEC-POMDP). As the formulated problem is NP-complete, we exploit some structural features of the proposed scheme to introduce the concept of symmetry and simplify the problem. Then, a novel decentralized federated learning-based solution is proposed to solve the problem. Simulation results show the effectiveness of the proposed approach in minimizing the total network energy consumption and achieving AoI values less than 200 $msec$ to support demanding real-time applications.

INDEX TERMS Internet of Things, UAV-aided mobile edge computing (UAV-aided MEC), age of information, network function virtualization, federated reinforcement learning.

I. INTRODUCTION

SMART farming in the contemporary world has gained paramount importance while playing a pivotal role in revolutionizing agricultural practices [1]. Conversely, the increasing global population along with environmental challenges have made the adoption of smart farming techniques indispensable. By leveraging data-driven insights, automation, and precision agriculture, smart farming not only enhances productivity but also addresses the pressing need for sustainable resource management [2]. On the other hand, the Internet of Things (IoT) has been a promising technology to provide connection among a large volume of devices that are deployed to provide a specific service for smart agriculture. A wide range of different use cases such as smart greenhouse monitoring, pest control, and irrigation/soil management can be considered [1], [3]. These use cases are mostly computation intensive and delay-sensitive [1], [4], [5]. A large volume of connected devices implies a large volume of data that must be processed accurately in a timely manner [6]. Such data computation and analysis demands a significant amount of processing and storage resources which put a constraint on energy-limited IoT devices. Depending on the specific application, the aggregated data from monitoring sensors, and captured images/videos from installed cameras in the field need to be processed in real-time to turn raw data into usable information. Introducing IoT and its applications in modern agriculture provides this industry with suitable tools to support farmers for better productivity, quality, and profitability [1]. Therefore, the scope of IoT extends beyond just agricultural land, encompassing a broader spectrum that includes the supply chain as well [7], [8].

Battery-operated IoT devices lack the capability to perform energy-intensive computations independently. Additionally, they face challenges related to limited processing power and storage capacity [9]. Therefore, MEC is proposed as a promising technology to address these limitations by offering additional computation and storage resources for IoT systems. This is achieved by providing servers where IoT devices can offload their computation tasks [10], [11]. On the other hand, large-scale agricultural operations often involve the distribution of IoT devices across expansive areas. In such scenarios, challenges may emerge concerning communication, particularly with issues related to the limited reachability between IoT devices and MEC infrastructure. To address these challenges, a proposed solution involves the integration of a network comprising unmanned aerial vehicles (UAVs) to enhance the capabilities of MEC [12], [13]. This approach aims to mitigate communication limitations stemming from the constrained transmission power of IoT devices and simultaneously improve network coverage. Due to the easy-to-deploy, cost-effective, and strong capabilities of the UAVs, UAV-aided MEC has attracted much attention and is widely utilized in smart agriculture to provide a high-quality line-of-sight (LoS) link to IoT devices [1], [3], [5], [14], [15], [16], [17]. The versatility and mobility exhibited by UAVs in response to changing weather conditions, alongside their straightforward deployment and economically viable maintenance costs, collectively establish UAVs as a proficient solution for supplying IoT devices with the necessary resources. Nevertheless, the UAVs themselves are quite often battery-powered which means their available energy is limited [6].

1) AGE OF INFORMATION

Besides energy, the freshness of information is another important aspect that needs to be considered in environmental monitoring and smart agriculture applications in which rapid protective and/or recovery actions are needed. Being more specific, within the realm of environmental monitoring applications and precision farming [18] in a UAV-aided MEC network, the IoT devices are strategically dispersed throughout specified regions to seamlessly gather real-time environmental data. Then the collected and pre-processed data by the UAVs finds its transmission route toward a localized MEC server; where a comprehensive analysis is conducted to facilitate the extraction of pertinent insights. These insights, in turn, play a pivotal role in fostering *prompt* agricultural *decision-making* and actions. The objectives can be the refinement of operations, the performance optimization, or the reduction of expenditure [1], [14], [19]. This expansion of smart agriculture's scope ensures a secure and sustainable food supply chain, underpinned by contextual and situational awareness derived from real-time event processing [20]. Accordingly, such applications are characterized by their intensive computational demands and *time-sensitive* nature [1], [3], [5], [14], [18]. Concisely, the freshness of information becomes a critical factor that demands careful consideration. Packet delay and inter-delivery times, as two exemplary metrics that are commonly used to quantify the performance of real-time applications, are not adequate to represent the freshness of information received at the destination. Recently, age of information (AoI) has been proposed as a novel criterion to quantitatively evaluate the freshness of information [6], [17], [21]. For a flow of data packets, and with emphasis on the freshness of data at the destination, AoI is defined as the time elapsed from receiving the most recent packet belonging to that data-packet flow [22], [23].

2) NETWORK FUNCTION VIRTUALIZATION (NFV)

From the above discussion, in the context of UAV-enabled smart agriculture paradigm – which constitutes the focal point of this paper – UAVs are confronted with a substantial amount of data necessitating prompt processing. This scenario embodies a dynamic computational framework wherein a bunch of processing functions demands seamless implementation across both the UAVs and the local server [20], [24], [25].

NFV is a key technology for implementing and managing computing machines in a reliable, efficient, and robust manner [26]. The NFV virtualizes the network functions (NFs) and abstracts them from the physical hardware, which enables rapid service function chaining (SFC) and service provisioning in UAV-aided MEC applications [27]. Considering the data-intensive and computation-based application of smart agriculture, multiple computing functions in the form of virtual network functions (VNF) should be deployed *sequentially* and *orderly* to provide the processed data for the final decision-making at the local MEC-server. Utilizing NFV significantly enhances the agility in deploying and managing network components and improves the robustness and scalability of networks [27], [28].

Therefore, a critical challenge to address is the optimal and efficient placement of Virtual Network Functions (VNFs) and determining how to route information packets among VNF components over the available NFV infrastructure, i.e., UAVs and the MEC server. The decision to distribute and allocate VNFs between both UAVs and the server, rather than solely on one of them, aligns with the primary goal of the proposed scheme—to minimize the Age of Information (AoI) while maintaining energy efficiency. Network traffic and the workload on both UAVs and MEC servers fluctuate over time, while changing channel conditions between the parties necessitate adjustments in the required communication resources for packet forwarding. Furthermore, certain processing functions, such as compression, may alter the packet size, so the trade-off of performing these functions locally and sending the smaller packets with spending fewer communication resources versus doing the entire processing locally becomes pivotal. Consequently, the placement must dynamically adjust to new conditions to ensure the minimization of AoI and energy efficiency [28]. In light of this, a general condition has been considered where the VNFO can, depending on network conditions (processing node resources, service type, and channel conditions), decide on the optimal placement to minimize AoI and energy consumption.

The work presented in [24] is a use-case of the practical implementation of smart agriculture in real-world contexts; where, the authors leverage a confluence of cloud computing, edge computing, and NFV technology to conceive a comprehensive framework tailored to the essential demands of soilless precision farming practiced within a fully-recirculating greenhouse [24].

3) FEDERATED REINFORCEMENT LEARNING (FRL)

The VNF placement and scheduling in our network settings can be expressed as integer programming with some constraints that reflect the service requirements and the network infrastructure's restrictions. Nevertheless, this problem is NP-complete and there is no standard solver that can solve such problems in polynomial time [29], especially for large-scale networks where the required computation to find the optimal solution increases exponentially. Recently, machine learning algorithms and artificial intelligent (AI) based solutions appear as a viable way to solve such complex problems in polynomial time [17], [29], [30]. Since its inception in 2017 [31], Federated Learning (FL) has reshaped many emerging intelligent IoT systems toward advanced FL architecture. The distributed nature of FL, where some agents cooperatively train a global ML model without directly sharing the local data, makes FL an attractive alternative to traditional centralized ML schemes. To be more specific, FL by pushing intelligent ML functions to the network edge enhances the privacy and scalability of IoT applications and networks [30].

In this paper, our focus is on use cases in smart agriculture that require live streaming and analysis, such as surveillance and environmental monitoring. Specifically, we address the flexible dynamic orchestration of NFV-enabled SFCs within the context of delay-sensitive services. The approach involves distributing VNFs across processing nodes, utilizing UAVs and local MEC server in a UAV-aided MEC network. The objective is to perform SFC while ensuring the freshness of information by jointly minimizing AoI and total energy consumption throughout the network. Condensing the system model and the definition of the problem, we present the following key insights:

- In the realm of smart agriculture applications, real-time information is collected by IoT devices on a smart farm and transmitted through hovering UAVs to the local server.
- VNFs must be executed sequentially on the raw packets, as they represent split functionalities of a single processing job. Meanwhile, certain VNFs, such as compression, may potentially alter packet sizes.
- Various service types are assumed, each with its specific VNF chain.
- The challenge involves determining the optimal placement and scheduling of VNF chains on processing nodes (UAVs and the local server), accounting for processing time, transmission delay, and power consumption (both transmit and processing power) in a distributed manner.
- We will demonstrate analytically that the problem:
  - Exhibits circular symmetry, wherein the optimal policies of the agents (UAVs) are identical.
  - The local observations of the agents serve as sufficient statistics for determining the optimal policy,

  We will show how these two features will simplify the problem significantly. Subsequently, we present a novel solution for solving the modeled problem.

The main contributions of our paper are summarized as follows:

- To the best of our knowledge this is the first time that the problem of dynamic orchestration of NFV-enabled SFCs in a multi-hop UAV-aided MEC network for smart agriculture is considered, while the problem is formulated as a joint AoI and Energy minimization.
- We formulate this joint optimization problem as a decentralized partially observable MDP (DEC-POMDP), where the parties are not aware of the true state and just make decisions based on their local observations.
- We adopted the structural feature of the problem and have analytically shown that under the satisfaction of certain symmetry conditions, the local observation of the parties (agents) would be a sufficient statistic for determining the optimal solution.
- As the formulated problem is NP-complete, we proposed a novel FL-based algorithm called Asynchronous FL Deep Q-Network (AFDQN) in which a set of distributed parties learn in parallel and aggregate their own experience through a coordinator.
- A Multi-hop network is considered, where the UAVs can offload their computing tasks to the other UAVs as well as the local MEC server.

The rest of the paper is organized as follows: In Section II some state-of-the-art studies will be reviewed. Section III describes the system model and the main components of the system in detail. Section IV presents the problem definition and formulation. In Section V, the problem is expressed as a DEC-POMDP, and some analytical results are given that support our proposed FL-based solution presented in Section VI. The complexity analysis of the proposed algorithm is presented in Section VII. The effectiveness and performance of the proposed scheme is demonstrated in Section VIII. Finally, Section IX concludes the paper.

II. RELATED WORKS

In this section, we review some state-of-the-art studies on AoI and energy-aware UAV-aided MEC for smart agriculture. The prospects of using UAVs for smart agriculture are immense. Moreover, UAVs are easy to deploy and cost-effective which motivates their use in smart agriculture [1], [6], [32]. For a comprehensive survey on IoT-based smart agriculture and the emerging technologies mentioned in the previous section refer to [1]. In [13], Mozaffari et al. have considered the reliable design of IoT's uplink communication in a scenario in which multiple UAVs are deployed to collect data from ground IoT devices. In particular, a framework for jointly optimizing the trajectory of the UAVs, IoT-to-UAV association, and IoT's uplink power is proposed with the aim of minimizing the total energy consumption and mobility of the UAVs. However, in the formulated problem, the delay of the forwarded data across the UAV network is not considered. Nguyen et al. [5] have considered this issue as the problem of processing deadline-critical tasks which are fed to a network of hovering UAVs that support the IoT devices deployed in a smart farm. It is assumed that the smart farm is equipped with a multi-access MEC infrastructure. In such a circumstance, the energy-efficient monitoring problem is modeled as a multi-objective maximization problem which aims to maximize the number of tasks that are successfully processed before their deadline. Then, a Q-Learning-based solution is proposed to solve the problem. The same authors in [5] have extended their proposed scheme to a DQN-based solution [33] and to a multi-actor-based risk-sensitive RL approach [32]. Although, the goal of the aforementioned studies is to minimize the energy consumption in the network, however, the proposed solutions are basically centralized and the communication overhead of the centralized approaches is itself a source of energy waste.

The AoI as a metric for determining the freshness of information has been used in some recent works On UAV-aided IoT networks [6], [17], [23], [34], [35]. Buyukates and Ulukus [34] examined a status update system where update packets require processing to extract embedded useful information. The source node sends information to a computation unit (CU) with a master node and worker nodes. The master node assigns tasks, aggregates results, and sends them back to the source node for updating. The analysis focuses on the *age performance* of various schemes in the presence of stragglers, considering i.i.d. exponential transmission delays and i.i.d. shifted exponential computation times. Then, the best scheme that minimizes the average age is presented. In [35], the authors analyzed the average age of information (AoI) and average peak AoI (PAoI) in a multiuser Mobile Edge Computing (MEC) system. The system considers three computing schemes: local computing, edge computing, and partial computing (where the computational tasks are partially performed at the edge and the remaining is performed by the local server). To address the complexity, upper and lower bounds on average AoI are provided, enabling an examination of optimal offloading decisions based on MEC system parameters.

In [6], Han et al. modeled a UAV-aided IoT network using a Markov chain. The freshness of data packets is defined using AoI and they analyzed the IoT devices as first-come–first-served (FCFS) model and M/M/1 queue. Sun et al. [17], employed AoI to propose an AoI-energy-aware data collection scheme for IoT networks in which the UAVs are used to collect data. Here, AoI is used to quantify the temporal correlation among data packets. Then, an algorithm for determining the UAV's flight speed, hovering locations, and allocated bandwidth to IoT devices is proposed that jointly minimizes energy consumption and the weighted sum of expected average AoI in the network. In [23], a UAV-aided wireless powered IoT scheme is proposed, where a UAV flies from a data center toward IoT sensory nodes to transfer energy and collect their information and then it returns back to the data center. The goal is to minimize the average AoI of the collected data from sensor modes. For such circumstances, an optimization problem is formulated, and then a suboptimal method is proposed that first decomposes the problem into two subproblems. The solution to the first subproblem is the input for the second subproblem. It is worth mentioning that the AoI is basically an end-to-end metric; Hence, even though the aforementioned works try to minimize the AoI, for the use cases in smart agriculture that the captured data needs some live processing before being turned into useful information, these approaches are not effective as they just consider the problem of finding the best data flow path.

In the context of UAV-aided MEC for IoT networks, each service can be represented as a service function chain (SFC) consisting of ordered processing functions in the form of VNFs that can be geographically placed on to local MEC-server or the UAVs. However, in a network with numerous IoT devices and dynamic network load, the placement of VNF instances and routing among them in an optimal and efficient manner is a challenging problem [27], [28]. In the literature, this problem is referred to as the SFC dynamic orchestration problem (SFC-DOP) [27]. In [27], Liu et al. presented a DRL-based framework for dynamic SFC orchestration in IoT networks. Huang et al. in [36] dealt with the problem of scalability and flexibility of static orchestration of NFV-enabled SFCs. Then, a FL-based SFC

TABLE 1. Summary of related works.

	Application	MDP type		MEC architecture		Objective Function (Minimization of)			ML solution	
Ref.	Smart agriculture	Multi-agent	Partially observable	UAV-aided	NFV enabled	Energy efficient	AoI awareness	SFC	Fedetared RL	Asynchronous
[5]	✓	×	×	✓	×	✓	×	×	×	×
[13]	×	×	×	✓	×	✓	×	×	×	×
[33]	✓	×	×	✓	×	✓	×	×	×	×
[18]	✓	×	×	✓	×	✓	×	×	×	×
[6]	✓	×	×	✓	×	✓	✓	×	×	×
[23]	×	×	×	✓	×	✓	✓	×	×	×
[27]	×	×	×	×	✓	×	×	✓	×	×
[36]	×	✓	×	×	✓	×	×	✓	✓	×
[28]	×	×	×	×	✓	×	×	✓	×	×
[17]	×	✓	×	✓	×	✓	✓	×	×	×
[29]	×	×	×	✓	×	✓	×	×	×	×
[32]	✓	✓	×	✓	×	✓	×	×	×	×
[37]	×	✓	×	✓	×	×	✓	×	✓	×
Our paper	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

orchestration is proposed which is scalable and benefits from low communication cost.

Table 1 presents an overview of the previously mentioned studies, emphasizing the main topics they concentrated on. Although, in the literature, the UAV-aided MEC architecture is mainly proposed as a technique to compensate for the energy and computational limitations of IoT networks, however, existing solutions for NFV-enabled SFC require IoT nodes to exchange large volumes of local data with a centralized server or among the distributed agents. Considering the energy limitation of IoT devices and battery-powered UAVs, this significantly causes waste of the network energy. In this paper, we deal with this inconsistency and propose a novel FL-based solution for the dynamic orchestration of SFCs in a multi-hop and UAV-aided MEC network. To the best of our knowledge, this is the first time that the problem of dynamic orchestration of NFV-enabled SFCs in a multi-hop UAV-aided MEC network is considered. What makes our approach unique is the adoption of the inherent structural aspects of the problem, typical in most scenarios, to introduce a decentralized solution that is analytically demonstrated to be valid. The proposed method is *asynchronous* FL-based, enabling distributed parties to independently learn locally and subsequently contribute to the training of the global model asynchronously [38], [39]. In other words, the parties are allowed to directly share gradients with the coordinator (here, the MEC server) after every local update and asynchronous of the other parties. This further enhances the training speed and efficiency of our proposed approach [39]. The coordinator in turn can perform the aggregation and update the global model whenever an update from one of the distributed parties is received. This approach improves the whole system's scalability and alleviates the straggler impact, i.e., users who may have slower performance [38].

The following notations are used throughout the remainder of the paper. Matrices and sets are denoted by Bold uppercase characters, and vectors are denoted by bold lower-case characters. The cardinality of a set $\mathcal{A}$ is represented by $|\mathcal{A}|$. The expected value of random variable X is denoted by $\mathbf{E}[X]$. The indicator function $\mathbb{1}_{\mathcal{A}}(a)$ is defined as $\mathbb{1}_{\mathcal{A}}(a) = 1$ if the element a belongs to $\mathcal{A}$, and $\mathbb{1}_{\mathcal{A}}(a) = 0$ if the element a does not belong to $\mathcal{A}$.

III. SYSTEM MODEL

A. GENERAL DESCRIPTION

We consider a real-time IoT network for smart agriculture applications, where, the IoT network provides real-time monitoring and visibility to network operators by video/image streaming. For such circumstances, several use cases from remote monitoring to security can be considered. As it is depicted in Fig. 1, a set $\mathcal{N}$ of N IoT devices collect real-time information from a smart farm and send the packets to a local server. A UAV-aided MEC architecture is considered, where packets are forwarded through an *Aerial Network* consisting of U hovering UAVs toward the local server. We consider applications in which some processing functions, from primary processes (e.g., compression) to advanced ones (e.g., object recognition) must be *sequentially* performed on the raw packets. Each IoT device is associated with one of the UAVs in its range. $\mathcal{U}$ denote the set of all U UAVs in the network. The local server is indeed a MEC-server[1] denoted by M. A set $\mathcal{S} = \{S^k\}_{k=1}^{K}$ with $|\mathcal{S}| = K$ different service

[1]Unless it makes ambiguity, in this paper the terms *local server* and *MEC* will be used interchangeably.

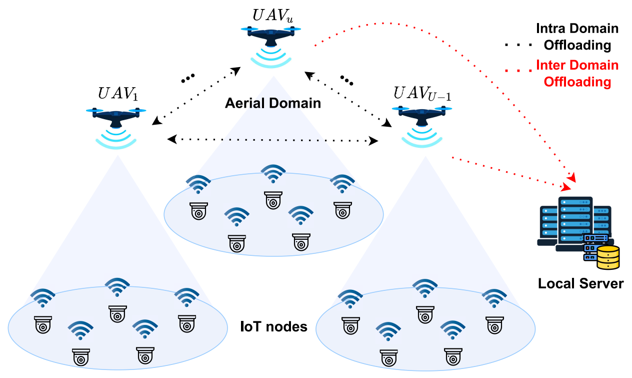


FIGURE 1. System model.

types is assumed. Each service type S_k itself consists of a set $\mathcal{F}^k = \{F_f^k\}_{f=1}^{F^k}$ including F^k different processing functions that should be performed on the packets of that service and $\mathcal{F}$ denotes a set of all processing functions of all services, $\mathcal{F} = \bigcup_{k\in\mathcal{S}}(\mathcal{F}^k)$. The input and output packet sizes of the function f of service type k are ρ_f^k and ϱ_f^k, respectively.

Let $s_n^k \in \{0, 1\}$ denote the service type $k \in \mathcal{S}$ in IoT node $n \in \mathcal{N}$ is active, $s_n^k = 1$, or not, $s_n^k = 0$. In each IoT node one of the K different services is running, $\sum_{k\in\mathcal{S}} s_n^k = 1, \forall n \in \mathcal{N}$. To perform the processing functions, the MEC-server M and each UAV u is able to run $F = \sum_{k\in\mathcal{S}} F^k$ different VNF types on their physical computing machine.[2] The proposed architecture is based on ETSI-NFV standard [40] which is a globally accepted architecture for implementing the NFV. According to ETSI-NFV, in each physical machine (processing node) $p \in \mathcal{U} \cup M$ *(All U UAVs and the local server M)*, the VNF manager (VNFM) is responsible to manage its computing and storage resources among the VNFs it hosts. The total available resources at processing node p is indicated by C_p, the computing capacity in Hz, and B_p, the memory capacity in Byte. The VNF orchestrator (VNFO), hosted by the local server, places and schedules the chain of VNFs through *Aerial Domain* and *Local (MEC) Server*.

For a summary of the key symbols and variables used in the system model and problem formulation, refer to Table 2.

B. VNF PLACEMENT AND SCHEDULING

We consider a discrete-time system with two hierarchical timing levels. First, the time is divided into equal time slots *TS* with duration T indexed by $t = 1, 2, \ldots$. On top of that, we have the VNF-scheduling time slots $\tilde{t}$ with duration $\widetilde{T}$ that is multiples of T, and it is a single round of VNF placement and scheduling update. At the beginning of each time slot $\tilde{t}$, the VNFO updates the VNF placements. Each UAV u segregates data packets from IoT nodes sharing the same service type, say k, into a packet-flow $\Upsilon_u^k(t)$, which it then transmits across the aerial network to the local server. The set $\mathcal{F}^k = \{\mathcal{V}^{fk}\}_{f=1}^{F^k}$ of F^k different VNFs (processing functions) must be performed on data packets of the IoT nodes with service type k.

[2]Depending on the load level of a service type, the processing node may run more than one instance of a VNF type.

TABLE 2. Key symbols and variables.

Parameter	Description
$\mathcal{U}$	The set of UAVs with a cardinality of U
$\mathcal{N}$	The set of IoT nodes with a cardinality of N
$\mathcal{S} = \{S^k\}_{k=1}^{K}$	The set of services with a cardinality of $\lvert\mathcal{S}\rvert = K$
$\mathcal{F}^k = \{F_f^k\}_{f=1}^{F^k}$	A set of F^k processing functions belongs to service type k
ρ_f^k	Input packet size of function f of service type k
ϱ_f^k	Output packet size of function f of service type k
C_p	Total computing capacity of processing node p in Hz
B_p	Total memory capacity of processing node p in Byte
T	Duration of each network-wide communication time slot
$\widetilde{T}$	Duration of each VNFO-level timing
$\Upsilon_u^k(t)$	Packet flow of service type k at UAV u
$\Delta_u^k(\tilde{t})$	The set of processing nodes participate in serving packet-flow $\Upsilon_u^k(t)$
$\mathcal{B}_{up}^k(\tilde{t})$	A subset of $\mathcal{F}^k$ that is assigned to be processed by node p
$\mathcal{P}_p^{to-do}(\tilde{t})$	All VNFs that is assigned to be processed by node p
$R_{nu}(t) \in \mathbb{C}$	Achievable bit rate of node n in up-link direction
$R_{u\acute{u}}(t) \in \mathbb{C}$	Achievable bit rate of the link between UAV u and $\acute{u}$
$R_{uM}(t)$	Achievable bit rate of UAV u in downlink direction
$\chi_{pu}^{fk}(\tilde{t}) \in \{0,1\}$	Denotes if node p is selected to run VNF f on the packet of service type k of UAV u
c^{fk}	Computing (CPU) resource, required by service flows
b^{fk}	storage capacity, required by service flows
$\Theta_u^k(t)$	AoI of the packet-flow of service type k in UAV u
$\Delta_u^k(\tilde{t})$	Sequence of UAVs that will sequentially do the chain of VNFs on packets of service type k
$\Phi_u^k(t)$	Total processing time of packets of service type k
$\Pi_u^k(t)$	AoI at the local server
Λ^k	Service packet length
$\Lambda^k(f)$	Function I/O packet size
$P^{UL}(t)$	Energy consumption of the network in the uplink direction
$P^{Aerial}(t)$	Energy consumption in the aerial domain
P^{NFV}	Energy consumption for performing the VNFs across a single packet of all service type

At the beginning of each VNFO-level time slot $\tilde{t}$, the VNFO determines the set of processing nodes $\Delta_u^k(\tilde{t})$ that participate in serving the kth service packet-flow $\Upsilon_u^k(t)$ of UAV u at tth TS. Each selected processing node $p \in \mathcal{U} \cup M$ that belongs to $\Delta_u^k(\tilde{t})$ performs a subset $\mathcal{B}_{up}^k(\tilde{t})$ of all functions $\mathcal{F}^k$ that is supposed to be performed on the packet

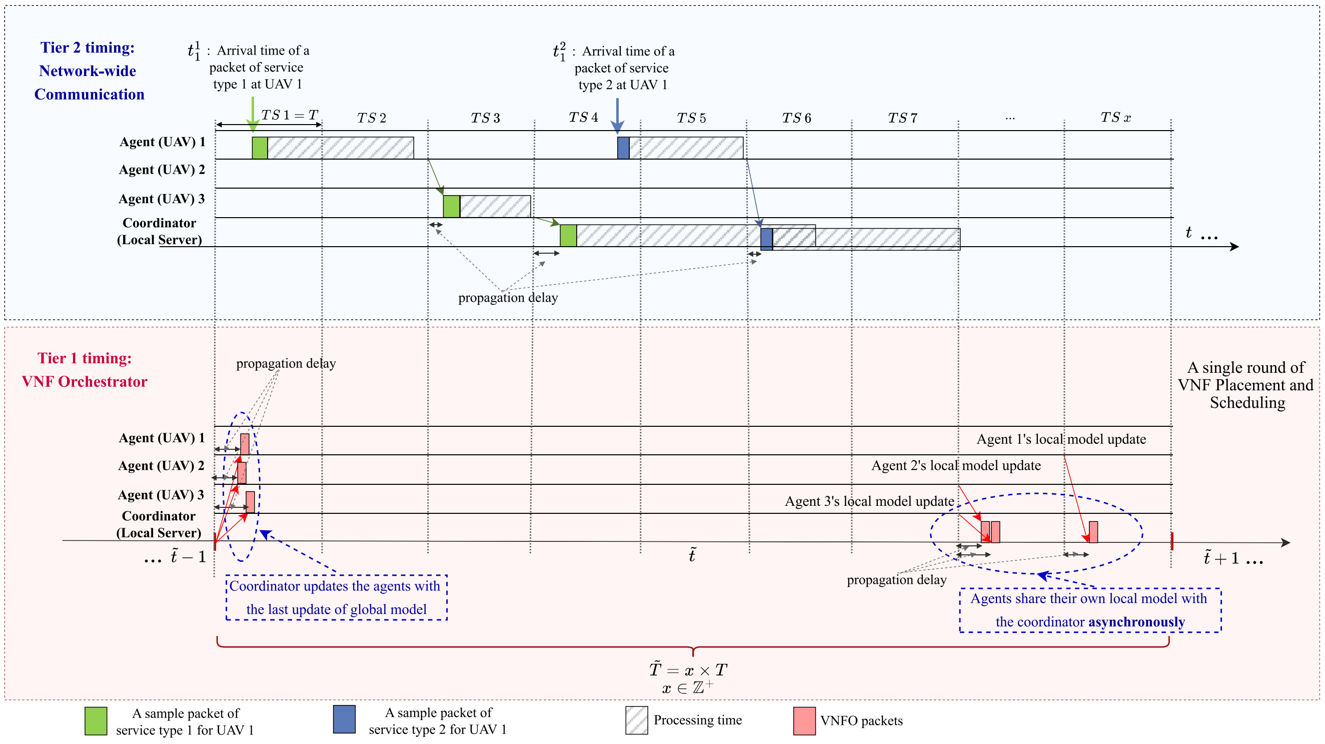

Tier 2 timing:
Network-wide
Communication
t_1^1 : Arrival time of a packet of service type 1 at UAV 1
t_1^2 : Arrival time of a packet of service type 2 at UAV 1
TS 1 = T
TS 2
TS 3
TS 4
TS 5
TS 6
TS 7
...
TS x
Agent (UAV) 1
Agent (UAV) 2
Agent (UAV) 3
Coordinator (Local Server)
t ...
propagation delay
Tier 1 timing:
VNF Orchestrator
A single round of VNF Placement and Scheduling
Agent 1's local model update
Agent 2's local model update
Agent 3's local model update
... $\tilde{t}-1$
$\tilde{t}$
$\tilde{t}+1$ ...
Coordinator updates the agents with the last update of global model
Agents share their own local model with the coordinator asynchronously
$\tilde{T} = x \times T$
$x \in \mathbb{Z}^+$
A sample packet of service type 1 for UAV 1
A sample packet of service type 2 for UAV 1
Processing time
VNFO packets

FIGURE 2. Network-wide timing model.

flow of service type k belongs to UAV u:

$$\mathcal{B}_{up}^k(\tilde{t}) = \left\{F_f^k : \dot{f}_{up}^k \le f \le \ddot{f}_{up}^k\right\},$$
$$\mathcal{B}_{up}^k(\tilde{t}) \subseteq \mathcal{F}^k, \tag{1}$$

where $\dot{f}_{up}^k(S)$ and $\ddot{f}_{up}^k(E)$ are the first and the last function (VNF) that p performs on the packet flow of service type k of UAV u. In summary, for each packet-flow $\Upsilon_u^k(t)$, the VNFO selects and schedules the processing nodes that handle $\mathcal{F}^k$. These nodes can be each one of any other UAVs (intra-domain offloading) or local server (inter-domain offloading).

Remark 1: Let $\mathcal{P}_p^{to-do}(\tilde{t}) = \{(u,k,f) : f \in \mathcal{F}_u^k$ is running on $p\}$ indicate a set of all assigned VNFs to the processing node $p \in \mathcal{U} \cup M$. We assume all assigned VNFs $\mathcal{P}_p^{to-do}(\tilde{t})$ to processing node p should be finished in a single round of VNF scheduling $\tilde{t}$.

Fig. 2 provides a comprehensive representation of crucial elements related to the proposed orchestration solution and the communication dynamics within the network. Operating at two distinct timing levels, the figure elucidates: 1) The VNFO timing (Tier 1), emphasizing interactions pivotal for establishing the proposed orchestration solution, and 2) the network-wide communication (Tier 2), encompassing all entities within the network.

The upper segment of the figure illustrates how UAV 1 manages received packet flows from IoT nodes, accommodating two distinct service types. Following the assumed VNFO policy in a single round of VNF placement and scheduling, it is evident that for packets of service type 1, UAV1 processes them and subsequently forwards them to UAV3. UAV3, after performing the assigned VNFs, then forwards the packets to the local server, i.e., the final destination. Conversely, for packets of service type 2, UAV1 initiates some initial processing and then directly forwards them to the local server, where all remaining VNFs are executed.

The lower section of the figure is dedicated to illustrating VNFO level operation, depicting control packets exchanged between the local server and agents for the training of the VNFO model. Further clarification on this VNFO communication is postponed to subsequent sections within the article.

C. COMMUNICATION (ACCESS) NETWORK

There are three communication links among the network nodes: the wireless links between IoT nodes and an aerial network consisting of UAVs, the UAV-to-UAV wireless links, and the wireless link between the UAVs and the local terrestrial server. Let random process $g_{nu}(t) \in \mathbb{C}$ denote the channel loss between IoT node $n \in \mathcal{N}$ and UAV $u \in \mathcal{U}$, then the achievable bit rate of node n in up-link direction at time instant t will be $R_{nu}(t) = w_n \log_2(1 + \frac{p_{nu}|g_{nu}(t)|}{\sigma^2})$, $\forall n \in \mathcal{N}, u \in \mathcal{U}$, where w_n and σ^2 denote the channel bandwidth of IoT device n and the noise variance, respectively, and p_{nu} is the transmission power level. The channel between IoT nodes and UAVs and between UAVs and the local server can be modeled as an air-to-ground channel model [41]. According to this model, the path loss, g_{nu} can be calculated as [37],

$$g_{nu}(t) = \left(\frac{4\pi f}{c}\right)^2 d^2(t)\eta_e, \tag{2}$$

where f, c, and d are frequency of operation, speed of light, and distance between the transmitter and receiver, respectively; and η_e is the average of excessive path loss in two cases of existing a LoS path, η_e^{LoS}, and non-LoS case, η_e^{nLoS},

$$\eta_e = p_{LoS} \times \eta_e^{LoS} + (1 - p_{LoS}) \times \eta_e^{nLoS}, \tag{3}$$

where p_{LoS} is the probability that a LoS path exists and can be closely approximated as [37],

$$p_{LoS} = \frac{1}{1 + a \exp -b(\psi - a)}, \tag{4}$$

where, a and b are environment-related parameters.

Similarly, in the downlink direction, the achievable bit rate of the link between UAV $u \in \mathcal{U}$ and MEC server M at time instant t will be $R_{uM}(t) = w_{uM} \log_2(1 + \frac{p_{uM}|g_{uM}(t)|}{\sigma^2}), \forall u \in \mathcal{U}$, where w_{uM} denotes the channel bandwidth, σ^2 is the noise variance, p_{uM} is the transmission power level, and random process $g_{uM}(t) \in \mathbb{C}$ denotes the channel Loss at time t. Finally, the UAV-to-UAV wireless channel also follows the same mathematical model, however, the only difference is that the probability of existing LoS is equal to 1, $p_{LoS} = 1$.

For each UAV u and service type k, $\tau_u^k(t)$ is defined as the expectation value of IoT access network delay with respect to transmission rate $R_{nu}(t)$,

$$\tau_u^k(t) = \boldsymbol{E}_{R_{nu}}\left[\tau_{nu}^k(t)\right],$$
$$\tau_{nu}^k(t) = D_{nu}/C + \Lambda_{nu}^k/R_{nu}(t), \tag{5}$$

where, $\boldsymbol{E}_{R_{nu}}$ denotes the expectation with respect to R_{nu}, D_{nu} is distance between IoT node $n \in \mathcal{N}$ and UAV $u \in \mathcal{U}$ and Λ_{nu}^k is the packet length of service type $S_k \in \mathcal{S}$. For the aerial radio links, Let random process $g_{u\acute{u}}(t) \in \mathbb{C}$ denote the channel power gain between UAV $u \in \mathcal{U}$ and UAV (or local server M) $\acute{u} \in \mathcal{U} \cup \mathcal{M}$, then the achievable bit rate of the link at time instant t will be $R_{u\acute{u}} = w_{u\acute{u}} \log_2(1 + \frac{p_{u\acute{u}}|g_{u\acute{u}}(t)|}{\sigma^2}), \forall n \in \mathcal{N}, u \in \mathcal{U} \cup M$, where $w_{u\acute{u}}$ denote the channel bandwidth between UAV u, and UAV (or local server) $\acute{u}$, σ^2 is the noise variance, and $p_{u\acute{u}}$ is the transmission power level. In the following sections, we will focus on the VNFO's functionality, and resource allocation of the radio access part is beyond the scope of this paper; hence, without loss of generality, we assume a fixed power and bandwidth allocated to all the participating nodes.

IV. PROBLEM FORMULATION

Let $\chi_{pu}^{fk}(\tilde{t}) \in \{0, 1\}$ denote whether the processing node p at time slot t is selected to run VNF function f on the packet of the kth service type of UAV u:

$$\chi_{pu}^{fk}(\tilde{t}) = \begin{cases} 1, \text{ if } p \in \Delta_u^k(\tilde{t}) \text{ and } f \in \mathcal{B}_{pu}^k(\tilde{t}) \\ 0, \text{ otherwise} \end{cases}$$
$$\forall k \in \mathcal{S}, u \in \mathcal{U}. \tag{6}$$

For each service type $k \in \mathcal{S}$ of UAV u, only one of the processing nodes (UAVs or MEC-server) must be selected for serving each function $f \in \mathcal{F}^k$:

$$\sum_{p \in \mathcal{U} \cup M} \chi_{pu}^{fk}(\tilde{t}) = 1, \forall f \in \mathcal{F}^k, \sum_{p \in \mathcal{U} \cup M} \chi_{pu}^{fk}(\tilde{t}) = 0, \forall f \nexists \mathcal{F}^k. \tag{7}$$

The relation (7), both left and right expressions together, implies that the packets belonging to the service packet-flows travel a loop-free route. Each packet belongs to packet-flow $\Upsilon_u^k(t)$ needs a specific computational capacity c^{fk} in CPU cycle. Assuming that all the processing capacity of processing node p in a single time slot with duration $\widetilde{T}$ is $C_p \widetilde{T}$, to be assured that the computing capacity of the selected processing node is enough to serve the assigned VNFs and the scheduler does not exceed the processing node's budget, the following condition at each VNF-scheduling time slot $\tilde{t}$ should be satisfied:

$$\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{S}} \sum_{f \in \mathcal{F}_k} \mathbb{1}_{\mathcal{P}_p^{to-do}(\tilde{t})}(u, k, f) c^{fk} \leq C_p \widetilde{T}, \forall p \in \mathcal{U} \cup M. \tag{8}$$

The same condition also needs to be fulfilled regarding the storage capacity requirement b^{fk} (in Bytes):

$$\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{S}} \sum_{f \in \mathcal{F}_k} \mathbb{1}_{\mathcal{P}_p^{to-do}(\tilde{t})}(u, k, f) b^{fk} \leq B_p, \forall p \in \mathcal{U} \cup M \tag{9}$$

where B_p is the total amount of available storage capacity of the processing node p.

A. AGE OF INFORMATION

In order to quantify the freshness of the received packet at the destination, the AoI metric is adopted. As soon as an IoT node has a packet to send, it connects to its serving UAV and sends packets.[3] For each packet flow of service type k that UAV $u \in \mathcal{U}$ receives directly from the IoT nodes that are connected to it and running this service type, the arrival time t_u^k is defined as the time elapsed from the beginning of the time slot t in which any packet has arrived in. Let $\Theta_u^k(t)$ denote the AoI of the packet-flow of service type k in UAV u, it can be calculated as,

$$\Theta_u^k(t) = \begin{cases} \tau_u^k + T - t_u^k, & \text{if } \alpha_u^k(t) = 1 \\ \Theta_u^k(t-1) + T, & \text{if } \alpha_u^k(t) = 0 \end{cases} \forall k \in \mathcal{S}, u \in \mathcal{U} \tag{10}$$

where binary variable $\alpha_u^k(t) \in \{0, 1\}$ indicates whether any new packet of service flow k at TS t is received, $\alpha_u^k(t) = 1$, or not, $\alpha_u^k(t) = 0$. By definition, for every time slot t in which the UAV does not receive a new packet from a service packet flow, the AoI of that service packet flow increases by T. On each received packet from IoT node n with service type k, the set $\mathcal{F}_k^n$ of VNFs (processing functions) should be performed. As stated above, the VNFO determines the set of processing nodes $p \in \Delta_u^k(\tilde{t})$ that participate in performing the VNFs on packet-flow $\Upsilon_u^k(t)$ by performing the set $\mathcal{B}_{up}^k(\tilde{t})$ of VNFs. $\mathcal{B}_{up}^k(\tilde{t}) = \emptyset$ means p hosts none of the required VNFs

[3]Communications between the nodes throughout the network are based on the smaller time scale t.

of $\Upsilon_u^k(t)$. Let $\Delta_u^k(\tilde{t}) = \{p_1, p_2, \ldots, p_L\}$ denote the sequence of all UAVs that are already selected to sequentially do the chain of VNFs on the received packets of service type k. The processing time of every packet of this flow will be,

$$\begin{aligned}\Phi_u^k(t) = &\sum_{p_v \in \Delta_u^k(\tilde{t})} \left(\left\lceil \frac{\tau^{fk}_{p_{v-i}p_v}(t)}{T} \right\rceil T + \sum_{\vartheta \in \mathcal{B}^k_{up_v}(\tilde{t})} \left(\left\lceil \frac{\theta^k_\vartheta}{T} \right\rceil T \right) \right),\\ &\forall k \in \mathcal{S}, u \in \mathcal{U}\\ &\tau^{fk}_{p_{v-i}p_v}(t) = D_{p_{v-i}p_v}/C + \Lambda_{p_{v-i}p_v}\left(F_f^k\right)/R_{p_{v-i}p_v}(t)\\ &F_f^k = \ddot{f}^k_{up_{v-i}}, \qquad (11)\end{aligned}$$

where θ^k_ϑ is the run time of ϑth VNF for service type k, $\Lambda_{p_{v-i}p_v}(F_f^k)$ is the packet length of service type $S^k \in \mathcal{S}$ after doing the last VNF $\ddot{f}^k_{up_{v-1}}$ of the chain in p_{v-1}, and, Assuming that the queueing delay is negligible, $\tau^{fk}_{p_{v-1}p_v}$ is total transmission delay between p_{v-1} and p_v consists of propagation delay and transmission delay.

If the binary variable $\beta_u^k(\tilde{t}) \in \{0, 1\}$ indicates whether the VNF scheduling (at VNF-scheduling time slots $\tilde{t}$) for the flow of packets belong to service type k of UAV u was successful, then, the AoI at the Local server will be,

$$\Pi_u^k(t) = \begin{cases} \Theta_u^k(t) + \Phi_u^k(t), \text{ if } \beta_u^k(\tilde{t}) = 1\\ \Pi_u^k(t-1) + \widetilde{T}, \text{ if } \beta_u^k(\tilde{t}) = 0 \end{cases} \forall k \in \mathcal{S}, u \in \mathcal{U}. \quad (12)$$

Note that $\widetilde{T}$ is the duration of a single round of VNF placement and scheduling.

B. ENERGY CONSUMPTION

The energy consumption of the network in the uplink direction can be calculated as,

$$P^{UL}(t) = \sum_{n \in \mathcal{N}} \sum_{k \in \mathcal{S}} \sum_{u \in \mathcal{U}} s_n^k p_{nu} \Lambda^k_{nu}/R_{nu}(t), \qquad (13)$$

where $\Lambda^k_{nu}/R_{nu}(t)$ is the transmission time between IoT node n, with service type k, and UAV u at TS t. The other energy-consuming part of the access network is transmission among the processing nodes. If we represent the energy consumption in the aerial domain (including UAV-to-UAV and UAV-to-MEC links) at TS t with $P^{Aerial}(t)$, then it will be as,

$$\begin{aligned}P^{Aerial}(t) &= \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{S}} \sum_{p_v \in \Delta_u^k} p_{p_{v-i}p_v} \tau^{fk}_{p_{v-i}p_v}(t),\\ \tau^{fk}_{p_{v-i}p_v}(t) &= D_{p_{v-i}p_v}/C + \Lambda_{p_{v-i}p_v}\left(F_f^k\right)/R_{p_{v-i}p_v}(t),\\ F_f^k &= \ddot{f}^k_{up_{v-i}}, \qquad (14)\end{aligned}$$

where $\Lambda_{p_{v-i}p_v}(F_f^k)$, $\ddot{f}^k_{up_{v-1}}$ and $\tau^{fk}_{p_{v-1}p_v}$ are defined like (11). Finally, if Ψ^k_ϑ denotes the power each machine, that hosts the ϑth VNF of service type k, consumes to run this VNF on each packet of this service type, then, the total network-wide required energy for performing the VNFs across a single packet of all service type can be calculated as follows:

$$P^{NFV} = \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{S}} \sum_{p_v \in \Delta_u^k} \sum_{\vartheta \in \mathcal{B}^k_{up_v}} \Psi^k_\vartheta \theta^k_\vartheta. \qquad (15)$$

In a single term, P^{NFV} represents the energy consumption resulting from VNFs processing.

Using (13)-(15), the total energy consumption of the network to process a single packet across all the service types belonging to all UAVs would be,

$$P^{Total}(t) = P^{UL}(t) + P^{Aerial}(t) + P^{NFV}. \qquad (16)$$

The UAVs use a battery, hence their available energy to do the processes and perform the required communications is limited. Therefore, we need an energy-efficient VNFO solution with minimum communication overhead. A centralized ML method will be optimal, but it requires a large communication transaction to share the local observation with the central controller. Another drawback of adopting centralized architecture is that the centralized coordinator is not scalable and from the processing and communication viewpoint is a single point of failure. Therefore, in this paper, we deal with proposing a solution for the following problem of distributed NFV orchestration through the UAVs as distributed VNFO agents.

Problem 1 (Distributed VNFO for Joint AoI and Energy Minimization): Considering the service requirements of IoT nodes, UAVs/MEC-server available resources, and the condition of access networks, what is the optimal strategy of VNF placement and scheduling in each UAV to jointly minimize the average AoI and total energy consumption at the Local Server:

$$\begin{aligned}\underset{\chi^{fk}_{pu}(\tilde{t})}{\textbf{\textit{Minimize}}} \quad & \gamma^{AoI}\left[\frac{1}{UK}\sum_{u \in \mathcal{U}}\sum_{k \in \mathcal{S}} \Pi_u^k(t)\right] + \gamma^E P^{total}(t),\\ \textbf{\textit{s.t.}} \quad & \sum_{p \in \mathcal{U} \cup M} \chi^{fk}_{pu}(\tilde{t}) = 1, \forall f \in \mathcal{F}^k,\\ & \sum_{p \in \mathcal{U} \cup M} \chi^{fk}_{pu}(\tilde{t}) = 0, \forall f \nexists \mathcal{F}^k,\\ & \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{S}} \sum_{f \in \mathcal{F}_k} \mathbb{1}_{\mathcal{P}_p^{to-do}(\tilde{t})}(u,k,f) c^{fk} \le C_p T,\\ & \forall p \in \mathcal{U} \cup M,\\ & \sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{S}} \sum_{f \in \mathcal{F}_k} \mathbb{1}_{\mathcal{P}_p^{to-do}(\tilde{t})}(u,k,f) b^{fk} \le B_p,\\ & \forall p \in \mathcal{U} \cup M. \qquad (17)\end{aligned}$$

In each VNFO-level time slot $\widetilde{T}$ the orchestrator sequentially decides on the chain of VNFs of the service flows belonging to UAVs. For a class of stochastic sequential decision-making problems, the Markov Decision Process (MDP) has been a powerful framework for the mathematical formulation and study of this type of problems. Another point that is worth mentioning is that our proposed method in Section VI is FL-based where the UAVs follow the same model trained cooperatively. Therefore, minimizing the average AoI over UAVs or minimizing the maximum value of AoI over UAVs are basically the same.

Depending on the environment state, the MDP output will be the best action (or at least the best upon the history of the observations and actions) which maximizes a specific utility function [42]. For the case that the state is not fully observable to the deciding agent or the agent's observations are exposed to noise or some source (sort) of uncertainties, another extended class of decision-making processes called Partially Observable MDP (POMDP) is adopted [43]. Both MDP and POMDP in their original scope are defined centralized [42], [43]. Partially Observable Stochastic Game (POSG) is the extended version of POMDP in which a set of distributed agents are involved in the decision-making process [44], [45]. By definition, if every agent has the same individual reward function, the POSG model becomes the Decentralized POMDP (DEC-POMDP) [45]. In the following two sections, first, we show how the problem can be modeled as a DEC-POMDP and then we will present our proposed method to solve the developed DEC-POMDP.

V. DEC-POMDP FORMULATION

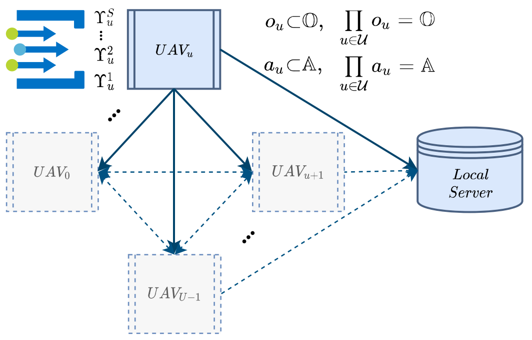
In a multi-agent MDP with state uncertainty, a DEC-POMDP is formally defined as a tuple with the following definition.

Definition 1 (DEC-POMDP Model): DEC-POMDP $\mathbb{G}$ with a set $\mathcal{U}$ of U agents is defined as a tuple $\mathbb{G} = \langle \mathcal{U}, \mathbb{S}, \mathbf{b}_0, \mathbb{A}, \mathbb{O}, T, O, \mathbb{R} \rangle$, where

- $\mathbb{S}$ is the finite set of global environment states,
- $\mathbf{b}_0$ is the probability of the environment initially being in state $s \in \mathbb{S}$,
- $\mathbb{A} = \prod_{u \in \mathcal{U}} A_u$ is the joint action of all agents, where A_u is the set of actions available to agent u,
- $\mathbb{O} = \prod_{u \in \mathcal{U}} O_u$ is joint observation, where O_u is the observations available to agent u,
- T is the transition function $T : \bigcup_{s \in \mathbb{S}} (s \times \mathbb{A}(s)) \times \mathbb{S} \rightarrow [0, 1]$, where $T(\acute{s}|s, \mathbb{A}(s))$ is defined as the transition probability from state s to $\acute{s}$ by doing joint action $\mathbb{A}(s)$,
- O is the observation function $O : \bigcup_{s \in \mathbb{S}} (s \times \mathbb{A}(s)) \times \mathbb{O} \rightarrow [0, 1]$, where $O(\mathbb{O}|s, \mathbb{A}(s))$ is defined as the joint observation at state s by doing joint action $\mathbb{A}(s)$,
- $\mathbb{R} = \{r_u\}_{u=0}^{U-1}$ is a set of reward functions $r_u : \bigcup_{s \in \mathbb{S}} (s \times \mathbb{A}(s)) \rightarrow \mathbb{R}$, where $r_u(s, \mathbb{A}(s))$ is defined as the reward received by u when $\mathbb{A}(s)$ is executed at the global state s.

In a DEC-POMDP, each agent $u \in \mathcal{U}$ based on its local observation $o_u \in O_u$ and a local policy π_u performs an independent action $a_u \in A_u$. In each partially observable state s, the joint action $\mathbb{A}(s) = \prod_{u \in \mathcal{U}} a_u(s)$ from the joint policy $\mathbb{P} = \prod_{u \in \mathcal{U}} \pi_u$ determines the next global state $\acute{s}$ and joint observations $\mathbb{O}$ according to transition and observation probability of $T(\acute{s}|s, \mathbb{A}(s))$ and $O(\mathbb{O}|s, \mathbb{A}(s))$, respectively. According to *Problem 1*, our purpose is to find the best choice for the sets $\Delta_u^k(t)$ of processing nodes p and $\mathcal{B}_{up}^k(t)$ of the VNF chains for serving packet-flow $\Upsilon_u^k(t)$ of any service type $k \in \mathcal{S}$ of UAVs $u \in \mathcal{U}$ in a distributed manner.

Remark 2 (Source of Uncertainties): We have considered a multi-hop network architecture, as depicted in Fig. 3, in which, 1) each agent u is not aware of the other agents' observation $\mathbf{o}_{-u}$ ($-u$ refers to all the agents except u, $\mathbf{o}_{-u} = \prod_{\acute{u} \neq u} o_{\acute{u}}$), nor the action $\mathbf{a}_{-u}$,[4] 2) each agent observes the *result* of doing its own action and the actions of the other agents, while it is not aware of the global state nor the action of the other agents. Therefore, according to *Definition 1*, it can be inferred that *Problem 1* in its decentralized form is a DEC-POMDP.



Υ_u^S
Υ_u^2
Υ_u^1
UAV_u
$o_u \subset \mathbb{O}, \quad \prod_{u \in \mathcal{U}} o_u = \mathbb{O}$
$a_u \subset \mathbb{A}, \quad \prod_{u \in \mathcal{U}} a_u = \mathbb{A}$
UAV_0
UAV_{u+1}
Local Server
UAV_{U-1}

FIGURE 3. System model as a DEC-POMDP.

In DEC-POMDP, each agent based on its actions and observations creates a local database that in time t can be represented as $h_u(\tilde{t}) = \{a_u(\tilde{t}-1), o_u(\tilde{t})\}_{\tilde{t}=1}^{\tilde{t}}, \forall u \in \mathcal{U}$, where $a_u(\tilde{t}-1)$ and $o_u(\tilde{t})$ are the action of u at time slot $\tilde{t}-1$ and its corresponding observation at $\tilde{t}$. All the information is available to all the U agents at time $\tilde{t}$ defines as the joint history $\mathbb{H}(\tilde{t}) = \prod_{u \in \mathcal{U}} h_u(\tilde{t}) = \{\mathbb{A}(\tilde{t}-1), \mathbb{O}(\tilde{t})\}_{\tilde{t}=1}^{\tilde{t}}$. The POMDP state is hidden from the agents. Hence, the agents would not be able to choose their actions based on knowing the true state. The standard approach for dealing with POMDPs is to find a solution to the fully observable equivalent belief-MDP [43]. Where, belief $\mathbb{B}(s, \tilde{t}) = P(s(\tilde{t}) = s|\mathbb{H}(\tilde{t}), \mathbb{B}(0)), \forall s \in \mathbb{S}$ defines as a probability distribution over the state space of the original POMDP knowing the joint history $\mathbb{H}(\tilde{t})$, i.e., all the available information from the sequence of interactions that the agents have had until now,where $\mathbb{B}(0)$ is the initial value the belief state function. Belief state function $\mathbb{B}(s, \tilde{t})$ is known as sufficient statistic for the history $\mathbb{H}(\tilde{t})$ [45]. Upon performing a new interaction at $\tilde{t}+1$, the belief value is updated from the belief point at time t considering the new interaction $\{\mathbb{A}(\tilde{t}-1), \mathbb{O}(\tilde{t})\}$. Despite the current hidden state, in DEC-POMDP the agents need to infer the action (the policy) of the other agents. This leads to the definition of multi-agent belief function [46] with the following definition,

$$
\begin{aligned}
& b_u : \bigcup_{s \in \mathbb{S}} (s \times \pi_{-u}(s)) \rightarrow [0, 1], \forall u \in \mathcal{U} \qquad (18)\\
& b_{u,\tilde{t}}(s, \pi_{-u}) = P\big(s(\tilde{t}) = s, \pi_{-u}|h_u(\tilde{t}), b_u(0)\big),\\
& \pi_{-u} = \prod_{\acute{u} \in \mathcal{U}, \acute{u} \neq u} \pi_{\acute{u}}.
\end{aligned}
$$

As it is evident, the multi-agent belief function is defined in a space that is a combination of the hidden global

[4]It should be mentioned that the local observation o_u of agent u is the result of the action of all agents, i.e., the joint action $\mathbb{A}$.

state s and joint policy π_{-u} of the other agents. From Bellman expectation equation [47], the action-value function $Q^{\pi}_{u,\tilde{t}}[(s,\pi_{-u}),a^{\pi}_{u}]$ is the expected return starting from state hidden state s, taking action a^{π}_{u} according to policy π_u, while the other agents follow the joint policy π_{-u},

$$Q^{\pi}_{u,\tilde{t}}\Big[(s,\pi_{-u}),a^{\pi}_{u,\tilde{t}}\Big] = \mathbf{E}\Big\{R_u\Big[(s,\pi_{-u}),a^{\pi}_{u,\tilde{t}}\Big] + \gamma_u Q^{\pi}_{u,\tilde{t}+1}\Big[\big(\acute{s},\pi_{-u}\big),a^{\pi}_{u,\tilde{t}+1}\Big]\Big\}. \quad (19)$$

where the action-value function decomposed into immediate reward plus discounted action-value of the successor state, and γ_u is the discount factor of agent u. With some mathematical manipulation, (19) can be written as,

$$\begin{aligned} Q^{\pi}_{u,\tilde{t}}\Big[(s,\pi_{-u}),a^{\pi}_{u,\tilde{t}}\Big] &= R_u\Big[(s,\pi_{-u}),a^{\pi}_{u,\tilde{t}}\Big] \\ &+ \gamma_u \sum_{(o_u,\mathbf{o}_{-u})\in\mathbb{O}} O\big[o_u,\mathbf{o}_{-u}|s,\big(a^{\pi}_u,\mathbf{a}^{-\pi}_{-u}\big)\big]\sum_{\acute{s}\in\mathcal{S}} T\big[\acute{s}|s,\big(\mathbf{a}^{-\pi}_{-u}\big)\big] \\ &\times \sum_{(a^{\pi}_u,\mathbf{a}^{-\pi}_{-u})\in\mathbb{A}} \pi_u\big(a^{\pi}_u|\acute{s}\big)\, Q^{\pi}_{u,t}\Big[\big(\acute{s},\pi_{-u}(\mathbf{o}_{-u})\big),\acute{a}^{\pi}_{u,\tilde{t}+1}(o_u)\Big]. \end{aligned} \quad (20)$$

Using (18), for a given belief state function $b_{u,\tilde{t}}(s,\pi_{-u})$, the action-value function $Q^{\pi}_{u,\tilde{t}}(b_{u,\tilde{t}},a^{\pi}_{u,\tilde{t}})$ will be,

$$Q^{\pi}_{u,\tilde{t}}\Big(b_{u,\tilde{t}},a^{\pi}_{u,\tilde{t}}\Big) = \sum_{s\in\mathcal{S}}\sum_{\pi_{-u}} b_{u,\tilde{t}}(s,\pi_{-u})\, Q^{\pi}_{u,\tilde{t}}\Big[(s,\pi_{-u}),a^{\pi}_{u,\tilde{t}}\Big]. \quad (21)$$

For an enough large value of $\tilde{t}$ ($\tilde{t}\to\infty$), the goal is to find the optimal policy π^*_u among available policies π_u which leads to the optimal Q-value (action-value) function,

$$Q^{*}_{u,\tilde{t}}\big(b_{u,\tilde{t}},a_{u,\tilde{t}}\big) = \arg\max_{\pi} Q^{\pi}_{u,\tilde{t}}\Big(b_{u,\tilde{t}},a^{\pi}_{u,\tilde{t}}\Big). \quad (22)$$

There is no straightforward solution for the aforementioned DEC-POMDP problem. Among decentralized methods, multi-agent solutions also need a large volume of communication overhead between the agents to share their local observations to converge. FL does not have the communication overhead of the centralized techniques and also does not necessitate the agents to share all of the data and local observations to converge. Although this specification is for providing privacy, in our problem it provides us with the gain of energy efficiency that arises because the agents (UAVs) do not need to share all of their observations. A few efforts, [45], [48], [49], have been made in the literature to capture and exploit some structural specifications of the understudied system (application) to find or at least simplify the problem of finding optimal policy (22). One of these works is one presented by Yongacoglu et al. [48], in which the authors have developed a class of POSGs that is characterized by symmetry across players in terms of cost and state dynamics. In view of this research, within the APPENDIX, we introduce a class of *Symmetric DEC-POMDP* and prove that *Problem 1* belongs to this class and is *Circularly Symmetric*. This implies

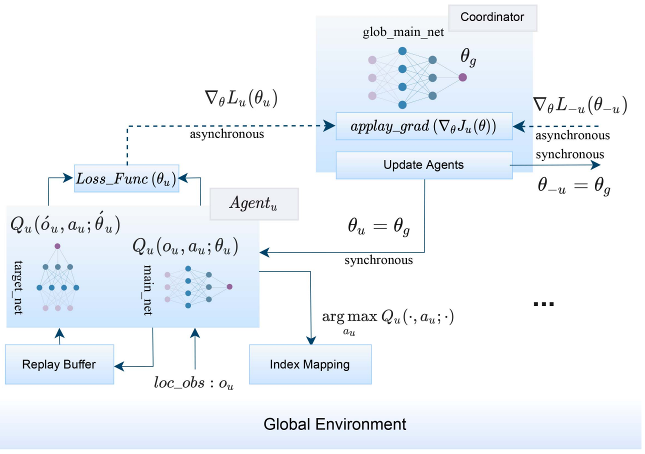


FIGURE 4. Block diagram of the proposed Asynchronous Federated-DQN (AFDQN).

that the best agents' policy, π_* are the same, $\{\pi_u\}_{u=0}^{U-1} = \pi^*$. In essence, this necessitates a distributed solution to determine the best policy while ensuring uniformity across all agents. Furthermore, in accordance with the circular symmetry characteristic of the problem, in *Corollary* 2 we prove that local observations serve as sufficient statistics for each agent to ascertain the best policy. *Corollary* 2 guarantees that the local observation $h_u(\tilde{t})$ serves as a sufficient statistic to determine the optimal policy π^* for the agents (UAVs). This implies that the information encapsulated in the local observations of the individual agents is comprehensive enough to determine the optimal policy π^*. While this corollary does not suggest a particular method for identifying the optimal policy, it is promising and justifies our proposed FL-based algorithm in which the agents collaboratively engage in training a globally shared model. This approach aligns with the notion that leveraging decentralized insights from each agent's local observations can contribute to the derivation of an effective and globally optimal policy because their observations are sufficient statistics.

Capitalizing on the promising findings in this section, the subsequent section introduces our proposed approach to address *Problem 1*.

VI. SYMMETRY-AIDED ASYNCHRONOUS FEDERATED DQN FRAMEWORK

In this section, we introduce our proposed Asynchronous Federated Deep Q-Network (AFDQN) algorithm and the components we have used in the proposed model as depicted in Fig. 4. In traditional RL, the problem is modeled as an MDP consisting of a tuple $\{s(\tilde{t}),a(\tilde{t}),r(\tilde{t}),s(\tilde{t}+1)\}$. At each decision-making time $\tilde{t}$, the agent is at state $s(\tilde{t})$ and takes an action $a(\tilde{t})$ based on a policy π that causes a state transition from $s(\tilde{t})$ to $s(\tilde{t}+1)$ while an immediate reward $r(\tilde{t})$ incurred. For our POMDP case in which the true state of the network is hidden, the true state $s(\tilde{t})$ is replaced with belief-state $b(\tilde{t})$. Mainly, the RL aims to guide the agent to find the best policy π^* defined as the best mapping from observation $o(\tilde{t})$ to action $a(\tilde{t})$ that maximizes the expected cumulative discounted future rewards $R(\tilde{t}) = \mathbf{E}_{\pi}\{\sum_{l=t}^{\infty} \gamma^{(l-\tilde{t})} r(l)\}$, where $\gamma \in [0, 1]$ is a discount factor indicating how much future rewards is important. For our DEC-POMDP problem, this relation maps to (19).

A. DEEP Q-NETWORK (DQN) PART

To estimate Q-value functions (21), deep reinforcement learning (DRL) is deployed, where Q-values are predicted using deep neural networks (DNNs) as function approximators. The estimated Q-functions are represented by $\{Q_u(o_u(\tilde{t}), a_u(\tilde{t}); \boldsymbol{\theta}_u(\tilde{t}))\}_{u=0}^{U-1}$, where the parameter $\boldsymbol{\theta}_u(\tilde{t})$ represents the weights of the agent u's neural network (NN). The updated value of $\boldsymbol{\theta}_u(\tilde{t})$ is used to approximate the true values of $Q_u(\tilde{t})$ [21], [50]. Let's define the loss function $L(\boldsymbol{\theta}_u(\tilde{t}))$ as the expectation value of the mean squared error of the estimated Q-value $Q_u(o_u(\tilde{t}), a_u(\tilde{t}); \boldsymbol{\theta}_u(\tilde{t}))$ from the target value $y(\tilde{t})$ [21],

$$L\big(\boldsymbol{\theta}_u(\tilde{t})\big) = \mathbf{E}\Big[\big(y(\tilde{t}) - Q_u\big(o_u(\tilde{t}), a_u(\tilde{t}); \boldsymbol{\theta}_u(\tilde{t})\big)\big)^2\Big], \quad (23)$$

where, $y(\tilde{t}) = r_u(\tilde{t}) + \gamma \max_{a_u(\tilde{t}+1)} Q_u(o_u(\tilde{t}+1), a_u(\tilde{t}+1); \boldsymbol{\theta}_u(\tilde{t}))$ and $a_u(\tilde{t}+1)$ indicates the agent's action generated by the DNN at $\tilde{t}+1$, given the observation $o_u(\tilde{t}+1)$.

At each iteration, the deep Q-function approximator is trained to learn the best estimate of the Q-function by minimizing the loss function $L(\boldsymbol{\theta}_u(\tilde{t}))$. To improve the stability of the algorithm and cope with samples correlation, as depicted in Fig. 4, two novel techniques, namely Fixed Target Network [51] and Experience Replay Buffer [52] are deployed, respectively. Utilizing these two techniques, the loss function $L(\boldsymbol{\theta}_u(\tilde{t}))$ can be written as

$$\begin{aligned} &L\big(\boldsymbol{\theta}_u(\tilde{t})\big) \\ &= \mathbf{E}_D\Bigg[\Bigg(r_u(\tilde{t}) + \gamma \max_{a_u(\tilde{t}+1)} Q_u\big(b(\tilde{t}+1), a_u(\tilde{t}+1); \acute{\boldsymbol{\theta}}_u(\tilde{t})\big) \\ &\qquad - Q_u\big(o_u(\tilde{t}), a_u(\tilde{t}); \boldsymbol{\theta}_u(\tilde{t})\big)\Bigg)^2\Bigg], \end{aligned} \quad (24)$$

where $\acute{\boldsymbol{\theta}}_u(\tilde{t})$ denotes the target network parameters, and the expectation $\mathbf{E}_D$ is taken over the randomly selected mini-batches of samples from the replay buffer D.

B. FEDERATED LEARNING PART

As it is illustrated in Fig. 4, we have two main entities, the set $\mathcal{U} = \{0, 1, \ldots, U-1\}$ of UAVs that are our distributed agents or in FL terminology, the agents, and the coordinator that in our model is local server (MEC-server). FL allows the UAVs (agents) to train a shared global model parameterized by $\boldsymbol{\theta}_g$ that is an exact copy of the agents' local model $\{\boldsymbol{\theta}_u\}_{u=0}^{u=U-1}$ using their own dataset $\{D_u\}_{u=0}^{u=U-1}$, while the original data remains in UAVs. After local training, agents share their local model updates with the coordinator. The coordinator then aggregates the received updates to build the global model $\boldsymbol{\theta}_g$. As a result, relying on the distributed data training at the agents, the local server is able to enhance the training performance without significant communication overhead as it just needs an update of the local model parameters, not the agents' local data. The federated learning procedure of our proposed method includes the following key steps.

1) DISTRIBUTED LOCAL TRAINING

Primarily, the local server initializes the global model, $\boldsymbol{\theta}_{g,0}$, and transmit it to the agents. Upon receiving $\boldsymbol{\theta}_{g,0}$, during VNFO time slots $\tilde{t}$ the agents interact with environment and train their local model $\{\boldsymbol{\theta}_u(\tilde{t})\}_{u=0}^{u=U-1}$ using their own data set $\{D_u(\tilde{t})\}_{u=0}^{u=U-1}$ by minimizing a loss function $\{L_u(\boldsymbol{\theta}_u(\tilde{t}))\}_{u=0}^{u=U-1}$,

$$\boldsymbol{\theta}_u^* = \arg\min_{\pi_u} L_u\big(\boldsymbol{\theta}_u\big((\tilde{t})\big)\big), \ \forall u \in \mathcal{U}. \quad (25)$$

Then, the agents upload their local update on $\{\boldsymbol{\theta}_u(\tilde{t})\}_{u=0}^{u=U-1}$ to the coordinator for aggregation.

2) MODEL AGGREGATION

After collecting the agents' local model updates, the next step is aggregating them into a new version of the global model which is performed by the coordinator by solving the following optimization problem.

Problem 2 (Model Aggregation): Given the local model updates $\{\boldsymbol{\theta}_u(\tilde{t})\}_{u=0}^{u=U-1}$ of all agents, and knowing the local loss functions $\{L_u(\cdot)\}_{u=0}^{u=U-1}$, what is the optimal strategy for aggregating the local model that minimizes the global loss,

$$\boldsymbol{\theta}_g^*, \ L_g^* = \arg\min_{L_g, \boldsymbol{\theta}_g} L_g\Big(\prod_{u\in\mathcal{U}} L_u\big(\boldsymbol{\theta}_g\big((\tilde{t})\big)\big)\Big). \quad (26)$$

According to *Problem* 2, $\boldsymbol{\theta}_g^*$ is the optimal value for $\boldsymbol{\theta}_g$ with having the local updates in hand; and L_g^* is the best function (the best method) for aggregating the local loss function. The loss-aggregation function L_g^* indicates the relative contribution of each agent on the global model, however, there is not a fixed method for determining this function and it depends mostly on the structure and specific features of the problem.

Corollary 1: According to *Lemma* 2,

1) The best setting for loss functions is (24) and $\{L_u(\cdot)\}_{u=0}^{u=U-1} = L(\cdot)$,
2) The optimal way for aggregating the local updates is averaging among the agent's contributions, thus (25) can be rewritten as, $\boldsymbol{\theta}_g^* = \sum_{u\in\mathcal{U}} \omega_u \boldsymbol{\theta}_u$, where ω_u represents the relative contribution of each agent on the global model.

After the derivation of a new update of $\boldsymbol{\theta}_g$ the coordinator broadcasts it to all agents. Upon receiving the update from the coordinator, the agents upgrade their local model accordingly. Until achieving a predefined level of accuracy or convergence of the global loss function, this process is continued.

C. ASYNCHRONOUS NETWORKING

The communication among the network entities follows the time slots t, but slot scheduling in which the agents share their own local model with the coordinator is distributed and *asynchronous*. Upon receiving an update, the coordinator aggregates it with the global model and updates the agents with the newly updated global model. In this way, we do not impose a strict constraint on synchronous communication with the local server (the coordinator). This significantly decreases the networking overhead in comparison with synchronous federation among distributed agents. During the training phase, we consider episodes in which the agent presets to a random initialization setting and starts interaction with the environment, and learns from its experience. Each episode contains $\widehat{T}$ VNF scheduling round $\tilde{t}$. In the deployment phase, the stream of packets that belong to different services is assigned based on an optimally determined VNF placement/scheduling policy. To avoid service interruption any fine-tuning and policy adaption to environmental changes, including the time the algorithm spends on fine-tuning and finding the optimal solution by coordinating multiple agents to train the global model, can be done in parallel.

Definition 2 (Global Update Period): global update period $1 \leq \eta \leq \widehat{T}$ is defined as the period of updating the coordinator by the agents. We have two special settings,

1) *AFDQN-SGD ($\eta = 1$):* In this case, every VNF scheduling slot $\tilde{t}$, the agent sends the locally calculated SGD, $\nabla_{\boldsymbol{\theta}} L_u(\boldsymbol{\theta}_u(\tilde{t}))$ to the coordinator. Then, the coordinator uses the received local data to perform one step of gradient descent:

$$\boldsymbol{\theta}_g\big(\tilde{t}+1\big) = \boldsymbol{\theta}_g\big(\tilde{t}\big) - \dot{\gamma}\nabla_{\boldsymbol{\theta}} L_u\big(\boldsymbol{\theta}_u(\tilde{t})\big). \tag{27}$$

2) *AFDQN-Avg ($\eta = \widehat{T}$):* In this case, only one time during each episode, the agent sends the whole parameter $\boldsymbol{\theta}_u(\tilde{t})$ to the coordinator. Then, the coordinator updates $\boldsymbol{\theta}_g(\tilde{t})$ accordingly,

$$\boldsymbol{\theta}_g\big(\tilde{t}+1\big) = (1-\ddot{\gamma})\boldsymbol{\theta}_g\big(\tilde{t}\big) + \ddot{\gamma}\boldsymbol{\theta}_u\big(\tilde{t}\big), \tag{28}$$

where, $\dot{\gamma}$ and $\ddot{\gamma}$ are AFDQN-SGD and AFDQN-Avg forgetting factors, respectively.

Introducing the forgetting factors $\dot{\gamma}$ and $\ddot{\gamma}$ allows for the adjustment of learning rates during model updates. Rather than updating the global model instantly upon receiving a local update, it is beneficial to employ a weighted average approach, considering both the most recent update and the previous value of the Agent's NN weight $\boldsymbol{\theta_g}$. This approach, known as *Asynchronous Weight Averaging* [53] in the literature, proves advantageous in alleviating the impact of outdated updates, commonly referred to as stale weights, and consequently, it enhances overall stability. The only point is that the forgetting factors should be chosen small enough.

Another worth mentioning point is that the AFDQN-SGD and a centralized approach doing mini-batch SGD in the local server are essentially different as the former is asynchronous, distributed, and fully based on local data. The details of the proposed AFDQN algorithm are described in *Algorithm1*.

Algorithm 1: AFDQN-η Algorithm

```
Coordinator:
- Initialize main Q-network parameter $\boldsymbol{\theta}_g(\tilde{t})$
- Initialize target Q-network parameter $\acute{\boldsymbol{\theta}}_g(\tilde{t})$
agent:
- Initialize $Q_u\big(o_u(\tilde{t}), a_u(\tilde{t}); \boldsymbol{\theta}_u(\tilde{t})\big)$ and $Q_u\big(o_u(\tilde{t}), a_u(\tilde{t}); \acute{\boldsymbol{\theta}}_u(\tilde{t})\big)$
- Initialize reply buffer
Environment: Initialize environment state $\mathbf{s}(0)$
Repeat for each agent $u$
for Episode 1 : total-number-of-episodes do
    for $\tilde{t} = 1 : \widehat{T}$ do
        - Get local observation $o_u(\tilde{t})$
        - Select action $a_u(\tilde{t}) = \pi_u(a_u(\tilde{t})|o_u(\tilde{t}); \boldsymbol{\theta}_u)$
        - Execute $a_u(\tilde{t})$, calculate $r_u(\tilde{t})$ from (31),
        - Save $\big\{o_u(\tilde{t}), a_u(\tilde{t}), o_u(\tilde{t}+1), r_u(\tilde{t})\big\}$ to replay buffer $D$
        if $|D| \geq$ batch-size then
            Sample a mini-batch of $D$, randomly
            for each sample do
                - calculate $\nabla_{\boldsymbol{\theta}} L_u(\boldsymbol{\theta}_u(\tilde{t}))$, cumulatively
                - Update main Q-network:
                  $Q_u\big(o_u(\tilde{t}+1), a_u(\tilde{t}+1); \boldsymbol{\theta}_u(\tilde{t}+1)\big)$
                - Update target Q-network:
                  $Q_u\big(o_u(\tilde{t}+1), a_u(\tilde{t}+1); \acute{\boldsymbol{\theta}}_u(\tilde{t}+1)\big)$
                if $\tilde{t}$ mod $\eta = 0$ then
                    if $\eta = 1$ then
                        Update $\acute{\boldsymbol{\theta}}_g(\tilde{t})$ from (27)
                    else
                        Update $\acute{\boldsymbol{\theta}}_g(\tilde{t})$ from (28)
                    end
                end
                Coordinator: Update the other agents
            end
        end
    end
end
```

Considering our optimization problem, for each agent $u \in \mathcal{U}$ at VNF scheduling round $\tilde{t}$, we define the observation space O_u, the action space A_u, and the reward function R_u as follows:

- **Observation:** We define the observation space as a vector of: 1) CPU and storage requested by the local service flows $\{\Upsilon_u^k(t)|t = \tilde{t}\}_k$ as $\{c^{fk}(t)|t = \tilde{t}\}_{f,k}$ and $\{b^{fk}(t)|t = \tilde{t}\}_{f,k}$, respectively, 2) available CPU $\{C_p\}_p$ and storage $\{B_p\}_p$ of the processing nodes, 3) service arrival time $\{t_u^k\}_k$, and 4) the transmission rate $\{R_{u\acute{p}}\}_{\mathcal{U}\backslash\acute{u}\cup M}$ of the links between agent u and the other processing nodes. Therefore, the observation $o_u(\tilde{t})$ can be written as

$$O_u\big(\tilde{t}\big) = \Big\{\big\{c^{fk}(t)\big\}_{f,k}, \big\{b^{fk}(t)\big\}_{f,k}, \big\{C_p\big\}_p, \big\{B_p\big\}_p, \big\{t_u^k\big\}_k, \big\{R_{u\acute{p}}\big\}_{\mathcal{U}\backslash u \bigcup M}\Big\}_{t=\tilde{t}}. \tag{29}$$

- **Action:** The action space is defined as all possible placement of the required VNFs for service flows $\{\Upsilon_u^k(t)|t = \tilde{t}\}_k$ as

$$A_u\big(\tilde{t}\big) = \Big\{\chi_{pu}^{fk}\big(\tilde{t}\big)\Big\}, \forall p \in \mathcal{U} \cup M,\ k \in \mathcal{S},\ f \in \mathcal{F}_k. \tag{30}$$

- **Reward:** Our objective is to orchestrate the VNFs in a way that jointly minimizes the average AoI and total energy consumption over the network. So we define the reward as a linear combination of three terms as follows:

$$R_u(\tilde{t}) = \delta^{NFV}\overline{\zeta}_u(\tilde{t}) + \delta^{AoI}\overline{\Pi}(\tilde{t}) + \delta^{E}P^{total}(\tilde{t}),$$
$$\overline{\zeta}_u(\tilde{t}) = \frac{1}{K}\sum_{k\in S}\zeta_u^k(\tilde{t})$$
$$\overline{\Pi}(\tilde{t}) = \frac{1}{UK}\sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{S}}\Pi_u^k(\tilde{t})$$
$$\zeta_u^k(\tilde{t}) = \begin{cases} +1, & \text{if (7)-(9) are satisfied} \\ -10, & \text{otherwise.}\end{cases} \quad (31)$$

where δ^{NFV}, δ^{AoI} and δ^{E} are the normalization factors for NFV scheduling result, AoI, and the energy consumption, respectively; $\zeta_u^k(\tilde{t})$ is defined as the reward assigned to the result of NFV placement for service flow $\Upsilon_u^k(t)|t=\tilde{t}$.

VII. COMPLEXITY ANALYSIS

In this section, we determine the computational complexity of the proposed *Algorithm 1*. We analyze the algorithm's complexity through two distinct phases: Model Training and Action Selection, which occur during the deployment of the trained model. During each iteration of the global update period η, the process involves the training of local models by the agents (UAVs) and subsequently, the asynchronous aggregation of these local models by the MEC server to form the global model. The complexity of the local model training conducted by the agents can be expressed as the sum of two components: the complexity of *action selection* and the complexity of the *back-propagation* algorithm for each sample within the replay buffer. This sum is then further multiplied by the mini-batch size and η. It is worth noting that the multiplication by the mini-batch size accounts for the fact that, in each training iteration, the local agent randomly selects a mini-batch of samples from its own local replay buffer.

The computational complexity of action selection in a fully connected neural network with a fixed number of hidden layers and neurons in each hidden layer is directly proportional to the sum of the input size and the output size of the neural network being used [54], [55]. The input size of the neural network is equivalent to the size of the state space, which from (29) is given by $2KF + 2(U+1) + K + (U-1) + 1$; and, the output size of the neural network is equal to the size of the action space. which from (30) is given by $KF(U+1)$. It's worth recalling that $F = \sum_{k\in\mathcal{S}} F^k$ represents the overall count of distinct VNFs required to be executed across all service types. Hence, the computational complexity associated with action selection is represented by $\mathcal{O}(KFU)$. Conversely, as indicated in equation [54], [55], for a specific sample extracted from the replay buffer, the computational complexity of the back-propagation process is directly proportional to the product of the neural network's input and output sizes. Consequently, in our specific scenario, derived from the preceding computations, the overall algorithmic complexity for each agent can be expressed as $\mathcal{O}(\eta OK^2F^2U)$, where, O denotes the batch size. From Algorithm 1, each agent repeats the whole process of local model training η times preceding sharing the results with the local server.

The last step is the aggregation process. While our algorithm conducts asynchronous aggregation, it's important to note that the processing load escalates proportionally with the number of UAVs, U. As a result, the overall computational complexity for the training phase across the entire network is given by $\mathcal{O}(\eta OF^2K^2U^2)$.

Based on the prior discussion, the complexity associated with the action selection for all U agents during the deployment phase is denoted as $\mathcal{O}(KFU^2)$.

VIII. PERFORMANCE EVALUATION

In this section, the performance of the proposed algorithm is evaluated. The performance results are compared for four different methods of AFDQN-SGD, AFDQN-Avg, AFDQN with parameter $\eta \neq \{1, \widehat{T}\}$, and a centralized case. In the centralized DQN method, all the observations are forwarded to the local server and the local server performs the VNF scheduling. Hence, its performance can be considered as an upper bound for the proposed AFDQN method.

A. SIMULATION SETUP

The simulation is implemented by Python using OpenAI gym [56], a widely used tool for developing RL algorithms, and conducted in a computer with Intel Core i7-10700 CPU 2.90 GHz and 64 GB RAM. Using simulation, the impact of parameter η, the number of IoT nodes, and the effect of the received load by the agents are evaluated. To this purpose, while $\widehat{T} = 60$, five different values for the parameter of η consisting of 1 (corresponds to AFDQN-SGD), 10, 20, 30, and 60 (corresponds to AFDQN-Avg) are considered. The other model parameters and simulation settings are summarized in Table 3 and Table 4. The proposed system model entirely matches none of the existing related works. However, the simulation parameters have been chosen to be in line with the typical values commonly used in similar studies within this context [6], [32], [33], [36], [37].

B. SIMULATION RESULTS

In Fig. 5, the convergence behavior of the AFDQN method is compared with the centralized method as a baseline. The two extreme cases of AFDQN-SGD and AFDQN-Avg are considered. As is evident from this figure, for the AFDQN method we have some variations around the value where it converged. The absolute value of these variations in AFDQN-Avg is more than AFDQ-SGD, and in AFDQN-SGD's case, they are around the value achieved by the centralized method. Also, the AFDQN-Avg converged to a smaller value for total return, we expect this result will also be reflected in the performance of AFDQN-Avg against AFDQN-SGD.

TABLE 3. Model parameters and simulation settings part I.

Parameter	Description	Value
U	Number of UAVs	3
N	Number of IoT devices	30, 60, 90, 120, 180
K	Number of service types	2
F^k	Number of processing functions (VNFs)	2
C^{fk}	Computing (CPU) resource, required by service flows	Randomly between 0.8 to 1.2 (normalized)
C_p	Computing (CPU) resource in processing nodes	1.8 (for UAVs), 2.2 (for local server) (normalized)
$b_f k$	Storage capacity, required by service flows	Randomly between 0.8 to 1.2 (normalized)
B_p	Storage capacity in processing nodes	1.8 (for UAVs), 2.2 (for local server)
t	VNFO-level time slot duration	500 msec
τ	Network-level time slot duration	5, 10, 15, 20
Λ^k	Service packet length	$k = 1$: 16KB; $k = 2$: 8KB
$\Lambda^k(f)$	Function I/O packet size	$k = 1 \rightarrow f_1$: 16KB, f_2 : 8KB, $k = 2 \rightarrow f_1$: 8KB, f_2 : 4KB.
p	Transmission power	IoT-to-UAV: 3 W UAV-to-UAV: 10 W
R	Transmission rate	IoT-to-UAV: 2 Mbps UAV-to-UAV: 4 Mbps
α	DQN learning rate	0.001
$\gamma^A oI$	AoI factor in objective function	0.5
γ^E	Energy consumption factor in objective function	0.5
γ	DQN discount factor	0.9
$\dot{\gamma}$	AFDQN-SGD forgetting factor	0.9
$\ddot{\gamma}$	AFDQN-Avg forgetting factor	0.1
ϵ	Exploration rate for DQN	0.001

TABLE 4. Model parameters and simulation settings part II.

Configuration Parameter	Value
Batch-size	8
Activation function	ReLU
AFDQN DNN hidden layers	23, 64, 128, 256
Centralized method DNN hidden layers	64, 128, 256, 512
Number of episodes	1000
Number of time steps	50
Target network update frequency	15
Experience replay buffer size	1000
Replay buffer size	1000
DNN optimizer	Adam (LR=0.001)
DNN kernel initializer	He-Uniform

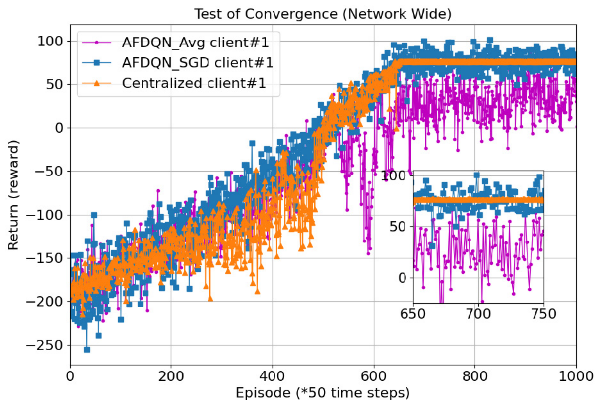
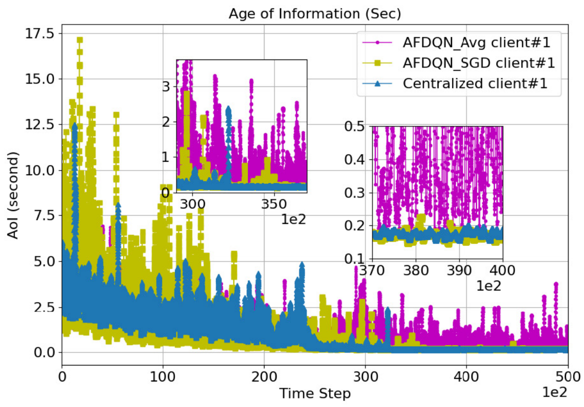
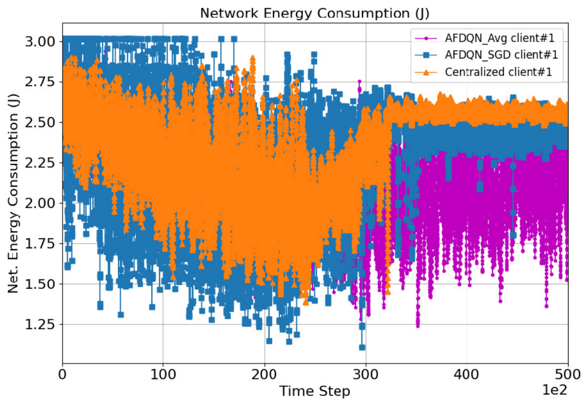


FIGURE 5. Episodic reward of agent 1 for the proposed method compared with centralized architecture. All agents follow the same policy, which is a copy of the cooperatively-trained global model.

This will be investigated in the following experiments. In Fig. 6(a) and Fig. 6(b), the minimum achievable average AoI and total energy consumption are investigated. The AoI value is averaged over all the agents and the total energy consumption is defined as the total energy consumption for sending a single packet of each service flow through the network from IoT nodes toward the local server. As can be seen, the maximum energy consumption of the three methods is limited to a maximum value corresponding to the maximum value of resources that the processing nodes can allocate to the requests. As a result, the negative side effect of the methods' drop in performance is mostly reflected in the value of the averaged AoI. For the AFDQN-Avg, this point is evidenced in Fig. 6(a), where we can see that at several points the values of the averaged AoI are more than 1 second. However, this is the worst-case choice for selecting the global update period η. Form Fig. 6(a), for the proposed AFDQN, we could achieve an average AoI of less than 200 msec, which is acceptable for most real-time applications. Therefore, performance close to the centralized case can be achieved by the proposed method in a distributed manner, relying solely on the local observations of the UAVs

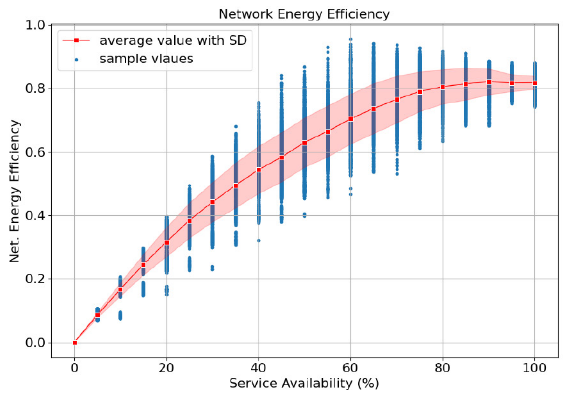


(a)

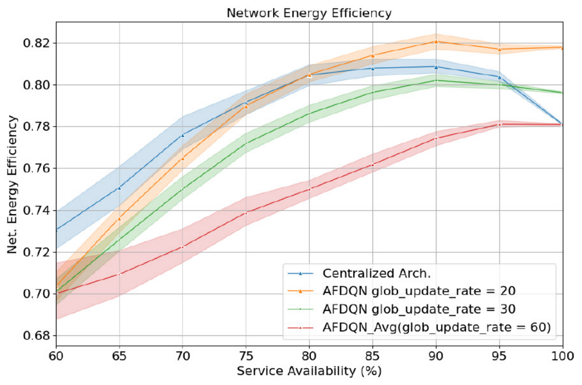


(b)

FIGURE 6. **(a) Average episodic AoI. (b) Average episodic total energy consumption of the proposed method compared with centralized architecture. All agents follow the same policy, which is a copy of the cooperatively-trained global model.**



(a)



(b)

FIGURE 7. **Network-wide energy efficiency vs. percentage of service availability. The shaded area represents the standard deviation (SD) from the average value: (a) For AFDQN with $\eta = 20$. Sample values are shown for more illustration. (b) For AFDQN with $\eta = 20$, $\eta = 30$, AFDQN-Avg and centralized architecture.**

and the sharing of trained local models, thus eliminating the need for exchanging the local observations, which can be time and energy-consuming.

In Fig. 7(a) and Fig. 7(b), the network energy efficiency versus service availability is drawn. The network-wide energy efficiency is defined as the total number of *successfully-supported* service flows divided by the total energy consumption throughout the network to send a single packet from each one of the services flows. Also, service availability is defined as the percentage of the service flows that are supported successfully. In Fig. 7(a), the average network energy efficiency and the sample values in our simulation, as well as the standard deviation of the value, for AFDQN-20 ($\eta = 20$) are shown. The averaged network efficiency with increasing the value of the service availability increases in a way that for the service availability of more than 80 percent, it converges to around 82 percent. This convergence for large value of service availability shows that we were eventually able to provide energy efficiency as one of our objective functions. In Fig. 7(b), where the proposed AFDQN method with different values of $\eta \in \{20, 30, 60\}$ is compared with the centralized approach, it can be seen that the energy efficiency of the centralized approach degrades as the availability reaches 100%. This behavior is because the centralized method has a single NN for all UAVs. For the availability values near 100%, the centralized agent is not able to do the job as efficiently as the AFDQN method where each agent has its own NN.

As illustrated in Table 3, the load of services (in terms of requested CPU and storage resources) is modeled as a normalized uniform random variable. The standard deviation (SD) of the values for both CPU request values and storage is 1.3 percent. In Fig. 8, the impact of increasing this value, up to 6 times the primary value of 1.3, on the network's performance is examined. The network is first trained with the primary input-load SD value of 1.3, and then, we incrementally raised the SD of the input load. This can be implicitly considered as a test under non-stationary load. It is evident that in this situation the AoI will increase, as illustrated in Fig. 8. To be more specific, we considered a threshold of one second for the acceptable AoI. Hence, we have also drawn a graph of AoI violation (in percentage) in conjunction with the bar chart of averaged AoI to better reflect the effect of the load SD. The results demonstrate that the AFDQN-Avg method has better performance in

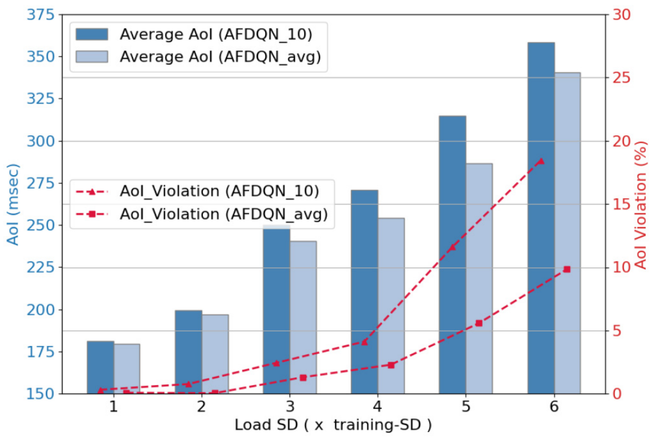


FIGURE 8. Averaged-AoI and AoI-violation percentage for AFDQN-10 and AFDQN-Avg versus standard deviation (SD) of the load normalized by SD of the load in the training phase.

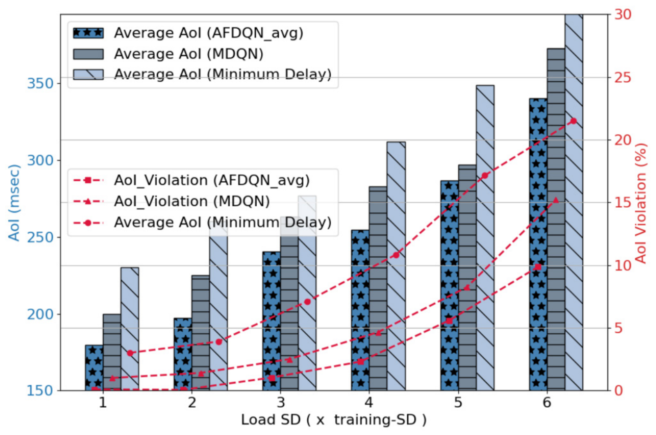


FIGURE 10. Averaged-AoI and AoI-violation percentage for AFDQN-avg compared with MDQN and Minimum-Delay versus standard deviation (SD) of the load normalized by SD of the load in the training phase.

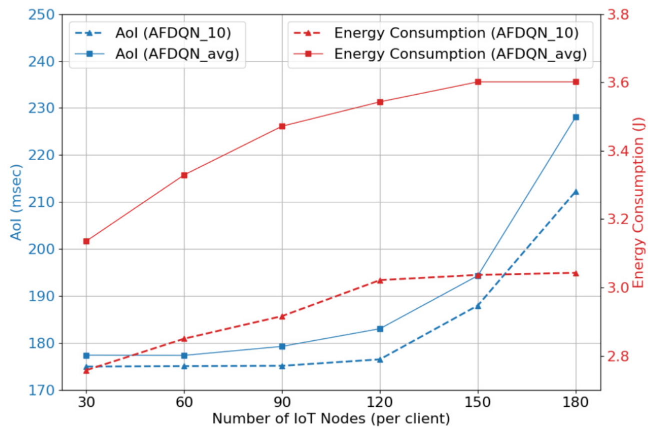


FIGURE 9. The effect of increasing the number of IoT nodes per agent (UAV) on AoI and energy consumption.

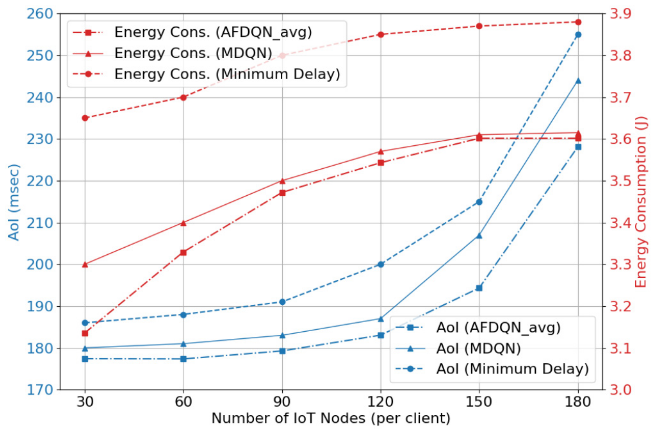


FIGURE 11. The effect of increasing the number of IoT nodes per agent (UAV) on AoI and energy consumption for AFDQN-avg compared with MDQN and Minimum-Delay.

this situation, which appears to contradict our previous observations where we concluded that decreasing the global update period improves the performance. This observation can be explained using the non-stationary behavior of the input load. In this case, the AFDQN-Avg lets the agents learn the variations and input model better, in comparison with AFDQN-10, which was chosen as an example for a case between two extreme cases, AFDQN-SGD and AFDQN-Avg.

The effect of increasing the number of IoT nodes per agent (UAV) on AoI and Energy consumption is investigated in Fig. 9. For this experiment, two AFDQN-10 and AFDQN-Avg cases are considered. The results show that AFDQN-10 had better performance than AFDQ-Avg. Another important point that should be mentioned relates to the behavior of the network for a large number of IoT nodes, e.g., 120 or more. As the number of IoT nodes increases, the load of the agents increases directly, this means that there are more CPU and storage requests. Because the resources in the processing nodes are limited, there comes a point where an increase in the number of IoT nodes will lead to more request rejection responses. In this situation, the fresh packets will drop, and that causes an increase in the AoI value. However, the network energy consumption does not change because the volume of the active processes does not change and remains equal to the maximum capacity of the processing nodes.

For further exploration, Fig. 10 and Fig. 11 compare the performance of the proposed algorithm against two baseline algorithms: Multi-agent DQN (MDQN) [57] and the adapted version of heuristic *Minimum-Delay* algorithm [58]. In MDQN the agents operate independently based on locally trained models without coordination. The agents are equipped with an identical neural network featuring the same specifications as AFDQN. Conversely, the Minimum Delay algorithm selects actions at each VNF scheduling round $\acute{t}$ to minimize end-to-end delay (not AoI) between IoT nodes and the local server.

In Fig. 10, a comparison similar to Fig. 8 assesses the impact of increasing load standard deviation (SD) on network performance in terms of AoI and AoI violation. Generally, an increase in SD raises the average AoI and AoI violation percentage for all methods, as larger SD values force agents to find the best action across a larger state space. As it is evident, the proposed AFDQN method exhibits the smallest average AoI and AoI violation percentage compared to baseline methods. However, the Minimum Delay method performs the worst due to its localized short-term target.

Finally in Fig. 11, the impact of increasing the number of IoT nodes per agent (UAV) on the proposed method's performance in terms of AoI and energy consumption is explored and compared with MDQN and Minimum Delay algorithms. Results demonstrate the superior performance of the proposed AFDQN method compared to baselines. Additionally, as the number of IoT nodes reaches 120 and beyond, there is an increased demand for CPU and storage; limited resources in processing nodes eventually lead to increased rejection of requests, causing a rise in AoI. However, network energy consumption remains unchanged, as the volume of active processes remains equal to the maximum capacity of processing nodes.

IX. CONCLUSION

We considered the problem of dynamic orchestration of NFV-enabled SFCs in smart agriculture applications to jointly minimize the AoI and energy consumption throughout the network. Especially, the problem is formulated as a decentralized POMDP. Then, adopting the symmetric structure of the network, we analytically proved that the optimal policy among the agents is behaving similarly. Accordingly, a novel federated-based DQN method was proposed to solve the problem efficiently. The proposed method is distributed and energy efficient, as the local agents just need to share the parameters of their locally trained model with each other. Although the primary goal of this method is to provide privacy among the agents, in our problem, this significantly decreased the communication overhead, and additionally, the total energy consumption of the network. In terms of freshness of information, the AoI is minimized jointly, and the achieved value for the AoI, while the parameter setting is set to be close to real values, is appropriate for most real-time applications.

APPENDIX

In this section, we aim to simplify the problem of determining the optimal policy (22) by utilizing some structural specifications of the developed system and problem. Inspiring by the work presented by Yongacoglu et al. [48], we introduce a class of Symmetric DEC-POMDP based on the Definition and Lemma presented below.

Definition 3 (Symmetric DEC-POMDP): A DEC-POMDP is called symmetric if the following conditions hold:

(i) $A_u = A_{\acute{u}}$ and $\gamma_u = \gamma_{\acute{u}}$, $\forall u, \acute{u} \in \mathcal{U}$

(ii) $\forall u \in \mathcal{U}$, $\forall \mathbf{a} \in \mathbb{A}$, and an arbitrary permutation function $\sigma(.)$:

a) $r_u(s, \sigma(\mathbf{a})) = r_{\sigma(u)}(s, \mathbf{a})$, and

b) $T(\cdot|s, \sigma(\mathbf{a})) = T(\cdot|s, \mathbf{a})$.

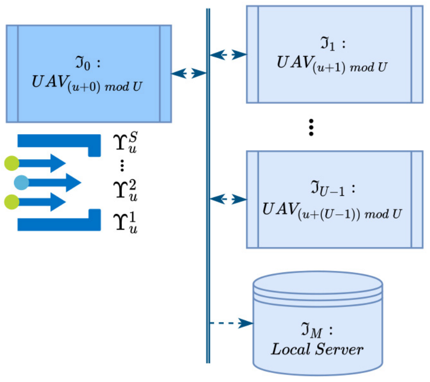

$\Im_0$: $UAV_{(u+0)\ mod\ U}$
$\Im_1$: $UAV_{(u+1)\ mod\ U}$
$\Im_{U-1}$: $UAV_{(u+(U-1))\ mod\ U}$
$\Im_M$: Local Server
Υ_u^S
Υ_u^2
Υ_u^1

FIGURE 12. Index mapping mechanism.

Lemma 1: Let $\mathbb{G}$ be a symmetric DEC-POMDP, for any $u, \acute{u} \in \mathcal{U}$, if $\pi_u = \pi_{\acute{u}}$, then, π_u is ϵ-best-response to π_{-u} **if and only if** $\pi_{\acute{u}}$ is ϵ-best-response to $\pi_{-\acute{u}}$, where ϵ-best-response (for an arbitrary $\epsilon \geq 0$) defines as a policy that achieves (reach) a reward (cost) within ϵ of the maximum (minimum) value.

In our VNF scheduling problem, as it is depicted in Fig. 3, at each VNF scheduling round $\tilde{t}$, each distributed agent $u \in \mathcal{U}$, has K packet-flow $\{\Upsilon_u^k(t)|t = \tilde{t}\}_{k=1}^K$ belong to different services each of which requires running F^k different VNFs on their packets. The agents decide on how to place these VNFs into available processing nodes, i.e., other $U - 1$ UAVs, the local server, or itself, in a way that jointly minimizes the average AoI and energy consumption according to (17). All the agents follow the same goal and the priorities among the agents are the same. Without loss of generality, we assume that the agents have chosen the same discount factor, $\{\gamma_u\}_{u \in \mathcal{U}} = \gamma$. Thus, it can be found that the agents conceptually have the same model; an intuitive inference that can be figured out better using the following *Index Mapping* rule.

Definition 4 (Index Mapping): For each function $f \in \mathcal{F}^k$ of packet-flow $\Upsilon_u^k(t)|t = \tilde{t}$, by definition we assume that the policy $\pi_u(\tilde{t})$ outputs $\Im_u^{fk} \in \{0, 1, \ldots, U - 1, M\}$ is the ith candidate processing node, $\Im_i$, as depicted in Fig. 12. The actual selected processing node among $p \in \{u \cup \{\acute{u}\}_{\acute{u} \neq u} \cup M\}$, will be,

$$p_u(\tilde{t}) = \begin{cases} [(u+i)\ mod\ U]^{th}\ UAV, & \text{if } \Im_i \neq \Im_M \\ M, & \text{if } \Im_i = \Im_M, \end{cases} \quad \forall\ \Im_u^{fk} = \Im_i. \tag{32}$$

Then, $\chi_{pu}^{fk}(|t = \tilde{t}) = 1$, and X_u^k, B_{pu}^k will be updated accordingly.

Now, according to *Definition 3* and *Definition 4*, the following lemma can be driven.

Lemma 2: **Problem 1** is a *symmetric* DEC-POMDP.

Proof: According to *Definition 4*, the agents have the same set of actions $\{A_u\}_{u \in \mathcal{U}} = \{\Im_i\}_{i \in \mathcal{U} \cup M}$, so the condition *(i)* of *Definition 3* is met. Let define $\mathcal{C}_\ell$ as circular shift operand with arbitrary value of ℓ, $\mathcal{C}_\ell(\cdot) = [\cdot + \ell]\ mod\ U$. Considering a circular shift of ℓ over joint action $\mathbb{A}$, $\mathcal{C}_\ell(\mathbb{A})$, we will have, $a_{\mathcal{C}_\ell(u)} = a_u$, thus from (32), for each $\Im^{fk}_{\mathcal{C}_\ell(u)} = \Im^{fk}_u = \Im_i$ the selected processing node $p_{\mathcal{C}_\ell(u)}(t)$ will be,

$$p_{\mathcal{C}_\ell(u)}(\tilde{t}) = \begin{cases} [(\mathcal{C}_\ell(u) + i) \; mod \; U]^{th} \; UAV, & \text{if } \Im_i \neq \Im_M \\ M, & \text{if } \Im_i = \Im_M, \end{cases} \quad \forall \, \Im^{fk}_{\mathcal{C}_\ell(u)} = \Im_i.$$

Or,

$$p_{\mathcal{C}_\ell(u)}(\tilde{t}) = \mathcal{C}_\ell\big(p_u(\tilde{t})\big). \tag{33}$$

Equation (33) means *Problem 1* is circularly symmetric. To be more specific, we do not have any special dependency on the identity of the agents and the state transitions depend only on the profile of joint actions performed by all agents. Accordingly, permuting the agents' actions does not change the conditional transition probabilities, $T(\cdot|s, \mathcal{C}_\ell(\mathbb{A})) = T(\cdot|s, \mathbb{A})$, and this permutation will lead to a corresponding permutation of rewards, $r^{\mathcal{C}_\ell(u)}(s, \mathbb{A}) = r^u(s, \mathcal{C}_\ell(\mathbb{A}))$. Hence, the second condition of *Definition 3* is also satisfied and the proof is complete. ■

According to *Lemma 2*, being circularly symmetric means that the best agents' policy, π_* are the same, $\{\pi_u\}_{u=0}^{U-1} = \pi^*$. For such a condition, *Lemma 1* implies that the problem of finding the best policy can be reduced to finding π^*_u, the best response to $\prod_{\acute{u}\neq u} \pi_{\acute{u}}$, while $\pi_{\acute{u}} = \pi_u$, $\forall \acute{u} \neq u$. Although this result is promising, from an implementation viewpoint, proposing a distributed solution for finding the best policy while requiring the same policy for all the agents is challenging. Recalling (18), in addition to the current hidden state, in DEC-POMDP the agents need to infer the action (the policy) of the other agents. The subsequent Corollary establishes a connection between this inference and solely relying on the local observations of the agents, thereby rendering it feasible.

Corollary 2: With the same initialization of the belief function, $\{b_u(0)\}_{u=0}^{U-1} = b(0)$, for a circularly symmetric DEC-POMDP, the local observation $h_u(\tilde{t})$ is a sufficient statistic for determining the optimal policy π^*.

Proof: Assume that using a mechanism, the agents peruse the same policy $\pi_u(\tilde{t})$, while they are learning the optimal policy π^*. Then, the multi-agent belief state $b_{u,\tilde{t}}(s, \pi_{-u})$ would be,

$$\begin{aligned} b_{u,\tilde{t}}(s, \pi_{-u}) &= b_{u,\tilde{t}}\left(s, \prod_{\acute{u}\neq u} \pi_u\right) \\ &= P\big(s(\tilde{t}) = s | h_u(\tilde{t}), b_u(0), \pi_u\big) \\ &= b^{\pi_u}_u\big(s, \tilde{t}\big). \forall u \in \mathcal{U} \end{aligned} \tag{34}$$

So, for each agent u, the local observation $h_u(\tilde{t})$ and knowing its local policy π_u is enough for determining the belief-state $b^{\pi_u}_u(s, \tilde{t})$. In a DEC-POMDP, the set $\{b^{\pi_u}_u(s, \tilde{t})\}_{u=0}^{U-1}$ is sufficient statistic for the joint history $\mathbb{H}$, and the proof is complete. ■

REFERENCES

[1] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, and X. Wang, "Internet of Things for the future of smart agriculture: A comprehensive survey of emerging technologies," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 4, pp. 718–752, Apr. 2021.

[2] X. Yang et al., "A survey on smart agriculture: Development modes, technologies, and security and privacy challenges," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 2, pp. 273–302, Feb. 2021.

[3] P. K. Reddy Maddikunta et al., "Unmanned aerial vehicles in smart agriculture: Applications, requirements, and challenges," *IEEE Sensors J.*, vol. 21, no. 16, pp. 17608–17619, Aug. 2021.

[4] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "Enabling massive IoT toward 6G: A comprehensive survey," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 11891–11915, Aug. 2021.

[5] A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Reinforcement learning-based deadline and battery-aware offloading in smart farm IoT-UAV networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2022, pp. 189–194.

[6] R. Han, J. Wang, L. Bai, J. Liu, and J. Choi, "Age of information and performance analysis for UAV-aided IoT systems," *IEEE Internet Things J.*, vol. 8, no. 19, pp. 14447–14457, Oct. 2021.

[7] K. Gupta and N. Rakesh, "IoT-based solution for food adulteration," in *Proc. 1st Int. Conf. Smart System, Innovat. Comput.*, Singapore, 2018, pp. 9–18.

[8] S. Nirenjena, D. Subramanian, and M. Monisha, "Advancement in monitoring the food supply chain management using IoT," *Int. J. Pure Appl. Math.*, vol. 119, pp. 1193–1196, Jan. 2018.

[9] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green Industrial Internet of Things architecture: An energy-efficient perspective," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 48–54, Dec. 2016.

[10] L. Lei, H. Xu, X. Xiong, K. Zheng, and W. Xiang, "Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-IoT edge computing system," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5345–5362, Jun. 2019.

[11] L. Dong, W. Wu, Q. Guo, M. N. Satpute, T. Znati, and D. Z. Du, "Reliability-aware offloading and allocation in multilevel edge computing system," *IEEE Trans. Rel.*, vol. 70, no. 1, pp. 200–211, Mar. 2021.

[12] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2334–2360, 3rd Quart., 2019.

[13] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient Internet of Things communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7574–7589, Nov. 2017.

[14] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A compilation of UAV applications for precision agriculture," *Comput. Netw.*, vol. 172, May 2020, Art. no. 107148.

[15] M. Bacco, A. Berton, A. Gotta, and L. Caviglione, "IEEE 802.15.4 air-ground UAV communications in smart farming scenarios," *IEEE Commun. Lett.*, vol. 22, no. 9, pp. 1910–1913, Sep. 2018.

[16] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36–42, May 2016.

[17] M. Sun, X. Xu, X. Qin, and P. Zhang, "AoI-energy-aware UAV-assisted data collection for IoT networks: A deep reinforcement learning method," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17275–17289, Dec. 2021.

[18] K.-V. Nguyen, C.-H. Nguyen, T. V. Do, and C. Rotter, "Efficient multi-UAV assisted data gathering schemes for maximizing the operation time of wireless sensor networks in precision farming," *IEEE Trans. Ind. Informat.*, vol. 19, no. 12, pp. 11664–11674, Dec. 2023.

[19] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, "An overview of Internet of Things (IoT) and data Analytics in agriculture: Benefits and challenges," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3758–3773, Oct. 2018.

[20] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big data in smart farming–a review," *Agricultural Syst.*, vol. 153, pp. 69–80, May 2017.

[21] F. Wu, H. Zhang, J. Wu, Z. Han, H. V. Poor, and L. Song, "UAV-to-device underlay communications: Age of information minimization by multi-agent deep reinforcement learning," *IEEE Trans. Commun.*, vol. 69, no. 7, pp. 4461–4475, Jul. 2021.

[22] Y. Sun, I. Kadota, R. Talak, and E. Modiano, "Age of information: A new metric for information freshness," in *Synthesis Lectures Communication Network*, vol. 12, San Rafael, CA, USA: Morgan Claypool Publ., Dec. 2019, pp. 1–224.

[23] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1211–1223, Jan. 2021.

[24] M. A. Zamora-Izquierdo, J. Santa, J. A. Martínez, V. Martínez, and A. F. Skarmeta, "Smart farming IoT platform based on edge and cloud computing," *Biosyst. Eng.*, vol. 177, pp. 4–17, Jan. 2019.

[25] A. Pretto et al., "Building an aerial–ground robotics system for precision farming: An adaptable solution," *IEEE Robot. Autom. Mag.*, vol. 28, no. 3, pp. 29–49, Sep. 2021.

[26] S. T. Arzo, C. Naiga, F. Granelli, R. Bassoli, M. Devetsikiotis, and F. H. P. Fitzek, "A theoretical discussion and survey of network automation for IoT: Challenges and opportunity," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12021–12045, Aug. 2021.

[27] Y. Liu, H. Lu, X. Li, Y. Zhang, L. Xi, and D. Zhao, "Dynamic service function chain orchestration for NFV/MEC-enabled IoT networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 9, pp. 7450–7465, May 2021.

[28] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, Nov. 2018.

[29] B. Khamidehi and E. S. Sousa, "Reinforcement learning-aided safe planning for aerial robots to collect data in dynamic environments," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13901–13912, Aug. 2022.

[30] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated learning for Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1622–1658, 3rd Quart., 2021.

[31] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from Decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[32] T. Pamuklu, A. C. Nguyen, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "IoT-aerial base station task offloading with risk-sensitive reinforcement learning for smart agriculture," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 1, pp. 171–182, Mar. 2023.

[33] A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Deep reinforcement learning for task offloading in UAV-aided smart farm networks," in *Proc. IEEE Future Netw. World Forum (FNWF)*, 2022, pp. 270–275.

[34] B. Buyukates and S. Ulukus, "Timely distributed computation with stragglers," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5273–5282, Sep. 2020.

[35] Z. Tang, Z. Sun, N. Yang, and X. Zhou, "Age of information of multi-user mobile-edge computing systems," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 1600–1614, 2023.

[36] H. Huang et al., "Scalable orchestration of service function chains in NFV-enabled networks: A federated reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2558–2571, Aug. 2021.

[37] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, "Federated Multiagent actor–critic learning for age sensitive mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1053–1067, Jan. 2022.

[38] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[39] S. Zheng et al., "Asynchronous stochastic gradient descent with delay compensation," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 4120–4129.

[40] "NFV release 2015 definition," Eur. Telecommun. Stand. Inst. (ETSI), Sophia Antipolis, France, document GS NFV-REL 002 V1.1.1, Sep. 2015.

[41] D. W. Matolak and R. Sun, "Unmanned aircraft systems: Air-ground channel characterization for future applications," *IEEE Veh. Technol. Mag.*, vol. 10, no. 2, pp. 79–85, Jun. 2015.

[42] R. Bellman, "A Markovian decision process," *J. Math. Mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[43] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1, pp. 99–134, 1998.

[44] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Math. Oper. Res.*, vol. 27, no. 4, pp. 819–840, 2002.

[45] B. K. Kang and K.-E. Kim, "Exploiting symmetries for single-and multi-agent partially observable stochastic domains," *Artif. Intell.*, vols. 182-183, pp. 32–57, May 2012.

[46] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2003, pp. 705–711.

[47] R. Sutton and A. Barto, *Reinforcement Learning, Second Edition: An Introduction* (Adaptive Computation and Machine Learning series). Cambridge, MA, USA: MIT Press, 2018.

[48] B. Yongacoglu, G. Arslan, and S. Yüksel, "Satisficing paths and independent multi-agent reinforcement learning in stochastic games," 2021, *arXiv:2110.04638*.

[49] M. Zinkevich and T. R. Balch, "Symmetry in Markov decision processes and its implications for single agent and Multiagent learning," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, p. 632.

[50] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Service function chain embedding for NFV-enabled IoT based on deep reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 102–108, Nov. 2019.

[51] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[52] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Banff, AB, Canada, 2017, pp. 316–321.

[53] Q. Wang, Q. Yang, S. He, Z. Shi, and J. Chen, "AsyncFedED: Asynchronous federated learning with euclidean distance based adaptive weight aggregation," 2022, *arXiv:2205.13797*.

[54] M. Sipper, "A serial complexity measure of neural networks," in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, pp. 962–966.

[55] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative Internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6807–6821, Nov. 2020.

[56] "Getting started with gym." openai. Accessed: Feb. 18, 2024. [Online]. Available: https://www.gymlibrary.dev/

[57] E. A. O. Diallo, A. Sugiyama, and T. Sugawara, "Learning to coordinate with deep reinforcement learning in doubles pong game," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2017, pp. 14–19.

[58] J. Cai, Z. Huang, J. Luo, Y. Liu, H. Zhao, and L. Liao, "Composing and deploying parallelized service function chains," *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102637.

MOHAMMAD AKBARI received the B.Sc. degree in electrical engineering from Tabriz University, Tabriz, Iran, in 2008, and the M.Sc. and Ph.D. degrees from the Iran University of Science and Technology, Tehran, Iran, in 2010 and 2016, respectively. From 2010 to 2017, he was a Senior System Designer with Afratab R&D Group, Tehran. In 2018, he joined as a Research Assistant Professor with the Department of Communication Technology, ICT Research Institute, Tehran. Since September 2021, he has been a Postdoctoral Fellow with the University of Ottawa, ON, Canada. His current research interests span topics in telecommunication systems and networks, including self-organizing networks, 5G and 6G networks, and the application of machine learning techniques in wireless communication.

AISHA SYED (Member, IEEE) is an Augmented Dynamic Networks Researcher with the Modelling and Optimization Group, Nokia Bell Labs. Her current research interests lie broadly in automating network and service management and evolution in the presence of challenges and opportunities created by increasing adoption of soft technologies and machine learning.

W. SEAN KENNEDY (Member, IEEE) received the joint Ph.D. degree in mathematics and computer science from McGill University in Montreal, Canada. In 2011, he joined Bell Labs as a Postdoctoral Researcher, before becoming a Technical Staff Member with the Mathematics of Network Systems Department. He currently heads Nokia Bell Labs' Artificial Intelligence Research Lab and is focused on creating solutions for critically hard and important industry problems through disruptive research into algorithms, machine learning fundamentals and applications, and real-time analytics. He applies his unique depth in both mathematics and computing technologies to envision the evolution and future effects of artificial intelligence ultimately building disruptive technologies to realize this vision. His current research focuses on moving beyond existing machine learning systems towards systems that mimic the human capacity for analytical thinking.

MELIKE EROL-KANTARCI (Senior Member, IEEE) is the Canada Research Chair in AI-enabled Next-Generation Wireless Networks and a Full Professor with the School of Electrical Engineering and Computer Science, University of Ottawa. She is the Founding Director of the Networked Systems and Communications Research Laboratory. She is the Co-Editor of three books on smart grids, smart cities, and intelligent transportation. She has over 200+ peer-reviewed publications. She has delivered 70+ keynotes, plenary talks, and tutorials around the globe. Her main research interests are AI-enabled wireless networks, 5G and 6G wireless communications, smart grid, and Internet of Things. She is an IEEE ComSoc Distinguished Lecturer and an ACM Senior Member. She has received numerous awards and recognitions including a Women in AI North America Award in 2023. She is on the editorial board of the IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, and IEEE NETWORKING LETTERS. She has acted as the general chair and the technical program chair for many international conferences and workshops.