IEEE ComSoc | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 23 March 2023; revised 24 July 2023; accepted 26 August 2023. Date of publication 4 September 2023; date of current version 6 October 2023.

The associate editor coordinating the review of this article and approving it for publication was M. Chen.

Digital Object Identifier 10.1109/TMLCN.2023.3311749

Constrained Federated Learning for AoI-Limited SFC in UAV-Aided MEC for Smart Agriculture

MOHAMMAD AKBARI[1], AISHA SYED[2], (Member, IEEE), W. SEAN KENNEDY[2], 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 (melike.erolkantarci@uottawa.ca)

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

ABSTRACT For a wide range of smart agriculture use cases, the prospects of utilizing the Internet of Things (IoT) are immense. Many IoT devices can be deployed for precision farming, soil management, automated irritation, information gathering, or performing some local processing to provide various services. Due to the computational capacity limitation of IoT devices and their limited power, UAV-aided mobile-edge computing (MEC) is proposed to provide IoT nodes with additional resources by hosting their computation functions and making smart agriculture use cases more operational. On the other hand, from the implementation viewpoint, Network Function Virtualization (NFV) is an emerging approach recently proposed for flexible management of such computation functions in UAVs and MEC-server. However, efficient orchestration of the virtualized functions is a challenge. In this paper, we consider a decentralized UAV-aided MEC system in which the NFV-enabled processing nodes manage the computational tasks. To be more specific, we consider the smart agriculture use cases that need live streaming/analysis, such as surveillance or environmental monitoring. In such a network, we propose a method for orchestrating the NFVs efficiently, while the network energy consumption throughout the network is minimized. This problem becomes even more crucial when considering a strict condition on the *instantaneous* AoI values. Therefore, the problem is first formulated as a Decentralized *Constrained* Multi-agent Markov Decision Process (Dec-CMMDP). As the formulated problem is NEXP, in the next step, we exploit some structural features of the considered network and introduce the concept of symmetry to simplify the problem. Then, inspired by Augmented Lagrangian dual optimization, 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, minimizing the average AoI, and satisfying the strict condition of $AoI < 100\ msec$ for supporting real-time applications in our network parameter settings.

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

I. INTRODUCTION

AGRICULTURE, as the main source of food that faces ever-increasing global demand, requires a major step forward in quality and productivity. The introduction of the Internet of Things (IoT) and its applications in *smart agriculture* provide this industry with effective tools to support farmers not only for better productivity but for greater profitability [1]. IoT provides connections among numerous devices that, in harmony with each other, are deployed to provide a specified service. The use cases cover a wide range of services such as crop field monitoring, pest control, smart autonomous irrigation, and soil management [1], [2]. The applicability of IoT is not limited to agricultural land itself but on a much wider scale includes the supply chain as well [3], [4]. Therefore, a huge amount of data provided by IoT nodes expanded throughout the agriculture and supply

chain needs to be processed and analyzed to support real-time and non-real-time decision making processes [1], [5]. That is while volatile weather conditions besides the vastness of agricultural fields will increase the risk and maintenance costs [2]. Therefore, such a huge data computation and analysis demand will put a strain on resource-limited IoT devices. To improve the capability of resource-constrained IoT devices and network coverage, UAV-aided MEC [6], [7] is proposed. The flexibility, and mobility in changing weather situations, as well as being easy to deploy and having reasonable maintenance costs make UAVs an effective solution to provide IoT devices with the required resources. This goal is realized by enabling the IoT nodes to offload their processing task to hovering UAVs which is equipped with the required storage, processing, and communication resources [2].

Therefore, IoT in conjunction with UAVs, has attracted much attention and is widely utilized in smart agriculture and smart farming [1], [2], [8], [9]. For instance, in environment monitoring applications or precision farming [10], the IoT devices are distributed throughout the intended area and collect real-time data from their surroundings. Next, the collected data is forwarded to the UAVs. Finally, the pre-processed data is forwarded to the local server for further processing and information extraction to support *timely* farming-related decisions and actions. The purpose of precision farming is to improve the accuracy of operations and maximize the overall performance while reducing cost by taking the field's variables such as weather conditions, and information on moisture level and soil water requirements (for example for smart autonomous irrigation) into account [1], [5], [8]. This extends smart agriculture scope to guarantee a secure and sustainable food supply chain provided by context and situational awareness through processing the real-time events where rapid protective and/or recovery actions are needed [11]. Such applications are computationally intensive, delay-sensitive [1], [2], [8], [10], [12], and *freshness of information* is an important aspect that needs to be considered.

With emphasizing information freshness, AoI is recently proposed as a metric to quantify the timeliness and freshness of the collected data [13], [14]. This metric is widely used in the context of IoT networks [15], [16] to evaluate the freshness of the data *at the destination node*. AoI is the time elapsed from the generation of the last-received update packet [14]. AoI increases linearly over time until the next fresh packet has arrived, and this is the main difference between the AoI and the other traditional metrics that quantify the timeliness of a designed system.

Before diving into another important aspect of smart farming realization, in the following, we will provide two use cases of real-world realization of smart agriculture. The first use-case is the work presented in [17] which exploits cloud and edge computing in conjunction with NFV technology to develop a system covering excessive requirements of smart precision farming. A platform of three layers is developed. The first layer is a local cyber-physical system that mainly gathers data by interaction with crop devices in a real-time manner. The second layer is an edge computing plane composed of VNFs where task offloading is accomplished. And finally, the third layer is a cloud platform to collect and record data. This 3-tier platform is able to cope with the requirements of soilless agriculture in full recirculation greenhouses [17]. The second case is the Flourish research project [18], an adaptable robotic solution that combines the capabilities of a small UAV network with another small network of autonomous unmanned multipurpose ground vehicles. Flourish is aimed at monitoring crop density, crop nitrogen level, and weed pressure to precisely classify weeds by developing multi-spectral perception algorithms. In the next step, the developed navigation and mapping system is able to locate weeds and perform selective spraying. All the above multi-function processes are performed without human intervention [18]. Therefore, in a UAV-aided smart agriculture scheme (which is the case of interest in this paper), the UAVs are exposed to a huge amount of data to process in a timely manner. In other words, we are faced with a dynamic computing environment where different processing/computing functions need to be implemented on the UAVs and the MEC local server in a scalable, flexible, easy-to-launch, and cost-effective manner [11], [17], [18]. From this point of view, virtualization of the network element functions (NFs), called NFV, is a key technology for reliably implementing and intelligently managing the NFs [19]. The NFV virtualizes the NFs and abstracts them from the physical hardware, which enables rapid service function chaining (SFC), and service provisioning in UAV-aided MEC applications [20]. Considering the data-intensive and computational-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. One of the most important problems that should be optimally and efficiently solved is the placement of VNFs, managing the resources among different VNFs, and determining how to route the packets of information among VNF components over the available NFV infrastructure. VNF Orchestrator (VNFO) performs this operation [21]. As the network traffic and VNF's load change over time, the placement needs to be dynamically adjusted to the new conditions as well [22]. Utilizing NFV enhances the agility in deploying and managing network components and improves the robustness and scalability of networks significantly [20], [22].

In this paper, we will focus on the smart agriculture use cases that require live streaming and analysis, where there is a strict condition on the AoI values. Surveillance and environmental monitoring, in general, are two examples of such use cases. Subsequently, the VNFO should decide on the placement and scheduling of the chain of VNFs to minimize total network energy consumption while resulting AoI values must be less than a predefined threshold. Therefore, to mathematically model such a constrained decision-making problem, the

extended version of the Markov Decision Problem (MDP), i.e., constrained MDP (CMDP) [23], is utilized. More specifically, we consider a distributed multi-agent CMDP (Dec-CMMDP) where each UAV is responsible for placement and scheduling its corresponding VNFs belonging to its support services. Nevertheless, unless for a particular case that the agents are transition and reward-independent, the optimal policy for this problem is NEXP-complete with no standard solution in polynomial time [24]. Recently, machine learning algorithms and artificial intelligence (AI) based solutions appear viable ways to solve such complex problems in polynomial time [25], [26], [27]. Since its inception in 2017 [28], Federated Learning (FL) has reshaped many emerging intelligent IoT systems toward advanced FL architecture. The distributed nature of FL, where some clients cooperatively train a global ML model without directly sharing the local data, makes FL an attractive alternative to traditional centralized ML schemes. Particularly, FL enhances the privacy and scalability of IoT applications and networks by pushing intelligent ML functions to the network edge [27].

In the context of delay-sensitive smart agriculture applications, we address the problem of robust and flexible management of *virtualized computing functions* by distributing them into processing nodes: UAVs and the local MEC server. The purpose is to perform this function chaining in such a way that the total energy consumption of the UAVs and IoT nodes is minimized. At the same time, a strict condition on the instantaneous values of AoI is satisfied. We formulate the problem above as a distributed multi-agent CMDP model and propose a novel energy-efficient FL-based solution to solve it. 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 *constrained* dynamic orchestration of NFV-enabled SFCs in a UAV-aided MEC network is considered under *instantaneous* strict conditions.
- We formulate this joint optimization problem as a Dec-CMMDP, where a strict condition on the instantaneous value of AoI must be satisfied.
- We developed an extended version of the Augmented Lagrangian dual optimization method in conjunction with Federated Learning to obtain the optimal policy for the formulated Dec-CMMDP problem.
- As the formulated problem is NEXP-complete, we adopted the inherent symmetry in the structure of the problem and proposed a novel Iterative Federated-based algorithm in which a set of distributed parties learns in parallel and aggregates their own experience through a coordinator.

The rest of the paper is organized as follows. Section II introduces the related works. Section III describes the system model. Section IV presents the problem definition and formulation. Section V explains how the problem can be modeled as a Dec-CMMDP. The proposed Iterative federated-based solution and the analytical results that support our proposed algorithm are presented in Section VI. The effectiveness and performance of the proposed scheme are demonstrated in Section VII. Finally, Section VIII concludes the paper.

II. RELATED WORKS

To enable different operations, such as environmental monitoring and automation, numerous IoT devices are used in IoT-based smart agriculture [1], [12], [15], [29]. For a comprehensive review of emerging technologies for IoT-based smart agriculture, refer to [1]. For a UAV-aided farm monitoring IoT scheme, Nguyen et al. [12] considered the problem of processing deadline-critical tasks which are fed by IoT devices deployed on the field. Assuming that a Multi-access MEC infrastructure is available, the energy-efficient monitoring problem is modeled as a multi-objective maximization problem, and a Q-Learning-based solution is proposed which aims to process the tasks before their deadline. The same authors in [29] have extended the proposed scheme in [12] to a multi-actor-based risk-sensitive RL approach.

In the context of UAV-aided IoT networks and to quantify the freshness of information, AoI has been widely used in some recent works [15], [16]. In [15], Han et al. considered a UAV-aided IoT system in which the performance of data gathering is analyzed in terms of packet loss rate and data quantity using a Markov chain. To define the freshness of data packets, they analyzed the AoI of devices as a first-come-first-served (FCFS) model and M/M/1 queuing. In [16], the authors considered a UAV-aided wireless powered IoT system, where a UAV takes off from a data center, flies toward sensor nodes to transfer energy, collects their information, and then returns to the data center. To minimize the average AoI of the data gathered from all ground sensors in such a system, an optimization problem is defined. Then, a suboptimal method is proposed to decompose the problem into two subproblems. The solution to the first subproblem is the input for the second subproblem. Zhu et al. in [30] investigated the age-sensitive MEC systems which benefit from UAVs. They proposed a multi-agent RL scheme for intelligent control of UAV's trajectory planning, data scheduling, and bandwidth allocation. The problem is modeled as an average Age-of-Information (AoI) minimization. Then, an actor-critic-based multi-agent RL framework is proposed, where edge devices and a center controller cooperatively learn the interactive strategies through their observations. To enhance system performance in terms of convergence, an FL mode is introduced into multi-agent collaboration.

Considering NFV-enabled UAV-aided MEC IoT networks, each IoT service can be expressed as a service function chain (SFC) defined as several strictly ordered VNFs. The VNFs can be geographically placed into the local MEC server close to IoT terminals or UAVs. However, optimally and efficiently placing VNFs and routing service paths through the VNF instances are challenging problems. This problem is also known as SFC dynamic orchestration (SFC-DOP) [20]. In [20], Liu et al. presented a DRL-based framework for dynamic SFC orchestration in IoT networks. Pei et al. in [22]

TABLE 1. Summary of related works.

	Application	MDP type		MEC architecture		Objective Function (Minimization of)			ML solution
Ref.	Smart agriculture	Constrained MDP	Multi-agent MDP	UAV-aided MEC	NFV-enabled MEC	Energy efficient	AoI awareness	SFC	Fedetared RL (Communication efficient)
[12]	✓	×	×	✓	×	✓	×	×	×
[7]	×	×	×	✓	×	✓	×	×	×
[9]	✓	×	×	✓	×	✓	×	×	×
[10]	✓	×	×	✓	×	✓	×	×	×
[15]	✓	×	×	✓	×	✓	✓	×	×
[16]	×	×	×	✓	×	✓	✓	×	×
[20]	×	×	×	×	✓	×	×	✓	×
[22]	×	×	×	×	✓	×	×	✓	×
[25]	×	×	✓	✓	×	✓	✓	×	×
[26]	×	×	×	✓	×	✓	×	×	×
[29]	✓	×	✓	✓	×	✓	×	×	×
[30]	×	×	✓	✓	×	×	✓	×	✓
Our paper	✓	✓	✓	✓	✓	✓	✓	✓	✓

have dealt with the SFC embedding and dynamic VNF placement in a geo-distributed cloud system. The problem is formulated as a Binary Integer Programming (BIP) to embed the SFC requests at the lowest possible cost.

A summary of related works along with the main topics they have mainly focused on is provided in Table 1. All problems mentioned above are modeled as *conventional MDP*, as this work did not consider the case that there is a strict condition on the AoI values. Liu et al. in [31] discuss this issue that, in practice, RL techniques often cannot be directly applied to physical systems, especially in cases where there are some constraints to satisfy, e.g., limit resource consumption. This paper has surveyed the existing approaches addressing CMDP using RL. Two main types of constraints, i.e., *cumulative* and *instantaneous* are considered, and their pros and cons are discussed. Among all approaches dealing with CMDP, there are two popular methods of *Lagrangian relaxation* based algorithms [31], [32], [33] and *Lyapunov function* based safe policy determination algorithms [34], [35], [36]. None of the above studies consider CMDPs with instantaneous constraints. Li et al. in [33] dealt with this problem as a reward-maximizing policy determination while satisfying certain constraints at each time step. They first treated the conditions in which the strong duality of CMDP is in place and then, inspired by the *Augmented Lagrangian* Method [32], have proposed a policy-gradient-based RL algorithm for instantaneously-constrained RL problems.

We summarized the main references in the literature review in Table 1. As evident from this table, there are a few works that consider AoI in smart agriculture networks, however, none of these studies consider CMDPs with instantaneous constraint. In addition, none of them consider the problem of SFC in such networks while benefiting from NFV technology. Therefore, the techniques that are proposed in these works are not able to be deployed as a solution for the proposed smart agriculture framework formulated in this paper. Our paper focuses on SFC in NFV-enabled MEC where the problem is formulated as a Dec-CMMDP. As opposed to other SFC studies we consider an instantaneously-constrained Dec-CMMDP, where a strict condition on the instantaneous value of AoI must be satisfied. At the next step in section VI, inspired by [33], we will propose our novel solution which is an extended version of the Augmented Lagrangian dual optimization method in conjunction with Federated Learning.

III. SYSTEM MODEL

The notations used in this section and the rest of the paper are as follows. Matrices and sets are denoted by Bold uppercase characters, and vectors are denoted by bold lower-case characters. $\mathbb{Z}^+$ and $\mathbb{Z}_0^+$ denote positive integers and positive integers plus zero, respectively. $|\mathcal{A}|$ is the cardinality of a set $\mathcal{A}$. $\boldsymbol{E}[X]$ indicates expected value of X. $\mathbf{1}_{\mathcal{A}}(a)$ is indicator function, $\mathbf{1}_{\mathcal{A}}(a) = 1$ if the element a belongs to $\mathcal{A}$, and $\mathbf{1}_{\mathcal{A}}(a) = 0$ if the element a does not belong to $\mathcal{A}$. $X^+ = Max\{x, 0\}, \forall x \in \mathbb{R}$.

In the context of smart agriculture, a real-time IoT network provides different types of real-time services to the network operator. These services can cover various applications, from environmental monitoring to automation. To be more specific, as it is depicted in Fig. 1, there is a set $\mathcal{N}$ of N IoT nodes that collect delay-sensitive real-time information; Then send them to a local server through an *Aerial Network* in the form of packets. The Aerial Network consists of U UAVs denoted by the set $\mathcal{U}$. Each IoT node is connected to a UAV in its range. There is a local server located near the network and is equipped with a MEC server[1] indexed by M.

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

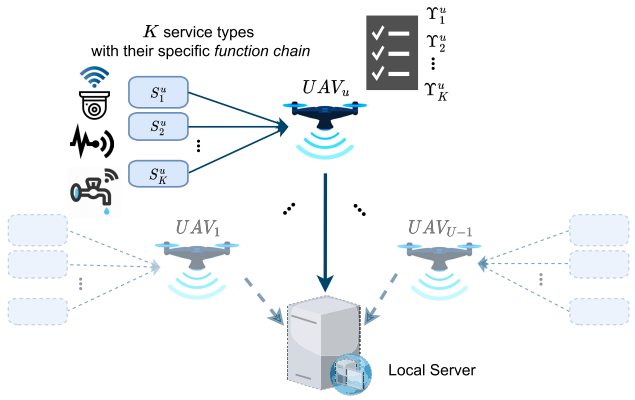

FIGURE 1. System model.

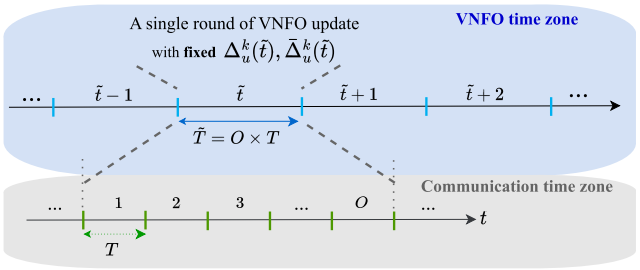

FIGURE 2. Illustration of timing model.

The IoT network provides a set $\mathcal{S} = \{S^k\}_{k=1}^{K}$ of $|\mathcal{S}| = K$ different services, each of which consists of a set $\mathcal{F}^k = \{F_f^k\}_{f=1}^{F^k}$ of F^k different processing functions should be performed on the packets of that service by following a logical order. Hence, it is assumed that the MEC server M and each UAV u, as processing nodes, can run $F = \sum_{k\in\mathcal{S}} F^k$ different VNF types on their physical computing machine. The total available resources at the physical machine (processing node) $p \in \mathcal{U} \cup M$ is indicated by C_p (in Hz) and B_p (in Byte), where C_p and B_p denote the computing and memory capacity at node p, respectively.

Each IoT node only supports one of the K different services provided by the network. If $s_n^k \in \{0, 1\}$ denotes the service type $k \in \mathcal{S}$ in IoT node $n \in \mathcal{N}$ is active, $s_n^k = 1$, or not, $s_n^k = 0$, then we have $\sum_{k\in\mathcal{S}} s_n^k = 1, \forall n \in \mathcal{N}$. However, there is a strict condition on the AoI of the service provided by each group of IoT devices. We will get back to this point later.

We consider a discrete-time system with two hierarchical timing levels as illustrated in Fig. 2. According to the first and smaller one, the time is divided into equal time slots indexed by $t = 1, 2, \ldots$, each with duration T. All communications in uplink and downlink directions are according to this time schedule. On top of that, we have the VNF-scheduling time slots $\tilde{t}$ with duration $\widetilde{T}$ that is a single round of VNF placement and scheduling updates. $\widetilde{T}$ is multiples of T, $\widetilde{T} = O \times T, O \in \mathbb{Z}^+$. At the beginning of each time slot $\tilde{t}$, the VNFO updates the VNF placements. Each UAV u combines all the data packets of the IoT nodes with the same service type, say k, and forwards the combined packet-flow $\Upsilon_u^k(t)$ throughout the aerial network toward 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.

At the beginning of each VNF-scheduling time slot $\tilde{t}$, for each service packet-flow $\Upsilon_u^k(t)$, the set of processing functions $\Delta_u^k(\tilde{t}) \subseteq \mathcal{F}^k$ which will be performed by the UAV u is determined. The remaining processing functions for that service, $\bar{\Delta}_u^k(\tilde{t}) = \mathcal{F}^k \setminus \Delta_u^k(\tilde{t})$, will be performed by the MEC server. $\Delta_u^k(\tilde{t}) = \emptyset$ means all the VNFs of that service will be performed at the MEC server. During the VNF-scheduling $\widetilde{T}$, the network follows a fixed placement rule determined by the VNFO until the next placement at the next VNF-scheduling time slot. It is assumed that the VNF orchestrator (VNFO) located in the local server decides the placement of the processing functions (VNF chains) of each service packet flow. Although the VNFO module is located on the local server, the orchestration process is performed by coordinating the UAVs as distributed agents. We will explain this process in the following sections in detail.

Let $v_p^{fk}(\tilde{t}) \in \mathbb{Z}_0^+$ denote the number of VNF instances from $\mathcal{V}^{fk}$ that is selected to be run on processing node p at VNF-scheduling time slot $\tilde{t}$:

$$v_p^{fk}(\tilde{t}) = \begin{cases} \mathbf{1}_{\Delta_p^k(\tilde{t})}(f) & \text{if } p \in \mathcal{U}, \\ \sum_{u\in\mathcal{U}} \mathbf{1}_{\bar{\Delta}_u^k(\tilde{t})}(f) & \text{if } p = M. \end{cases} \tag{1}$$

The set of all assigned VNFs to the processing node $p \in \mathcal{U} \cup M$ is indicated by $\mathcal{P}_p^{to-do}(\tilde{t}) = \{|v_p^{fk}(\tilde{t})| \text{ instances of } \mathcal{V}^{fk}\}$.

There are two communication links among the network nodes: the wireless links between IoT nodes and an aerial network consisting of UAVs and the wireless links between the UAVs and the local terrestrial server. Let $l_{nu}(t)$ 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 uplink direction will be $R_{nu}(t) = w_n \log_2(1 + \frac{p_{nu}}{l_{nu}(t)\sigma^2}), \forall n \in \mathcal{N}, u \in \mathcal{U}$, 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 [37]. According to this model, the path loss, l_{nu} can be calculated as [30],

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

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 on Line-of-Sight (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 of existing on the LoS path and can be closely approximated as [30],

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

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 the MEC server M will be $R_{uM}(t) = w_{uM} \log_2(1 + \frac{p_{uM}}{l_{uM}(t)\sigma^2}), \forall u \in \mathcal{U}$, where w_{uM} denotes the channel bandwidth, σ^2 is the noise variance, p_{uM} is the transmission power level, and $l_{uM}(t)$ denotes the channel Loss at time t. 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

As explained in the previous section, at the beginning of each VNF-Scheduling time slot $\acute{t}$, the VNFO decides on the placement of VNF chains.

Definition 1 (Placement Index): For the chain of VNFs corresponding to service packet-flow $\Upsilon_u^k(t), \forall k \in \mathcal{S}, u \in \mathcal{U}$, placement index $\delta_u^k \in [0, |\mathcal{F}^k|]$ is defined as the point where the chain is broken into two parts. The first part will be placed in UAV u and the second part in MEC server M. $\delta_u^k = 0$ (or $\delta_u^k = |\mathcal{F}^k|$) means all VNFs are performed in the MEC server M (or UAV u).

Definition 1 implies that the packets travel a loop-free route. Each packet belonging to $\Upsilon_u^k(t)$ needs a specific computational capacity c^{fk} in terms of CPU cycles. Assuming that all the processing capacity of processing node p in a single time slot with duration $\widetilde{T}$ is $C_p\widetilde{T}$, the following condition at each VNF-scheduling time slot $\tilde{t}$ should be satisfied:

$$\sum_{k\in\mathcal{S}}\sum_{f\in\mathcal{F}_k} \mathbf{1}_{\mathcal{P}_p^{to-do}(\tilde{t})}(\mathcal{V}^{fk})|v_p^{fk}(\tilde{t})|c^{fk} \le C_p\widetilde{T}, \qquad \forall p \in \mathcal{U}\cup M. \tag{5}$$

The above condition ensures that the computing capacity of the selected processing node is enough to serve the assigned VNFs. The same condition also needs to be fulfilled regarding the storage capacity requirement b^{fk} (in Bytes):

$$\sum_{k\in\mathcal{S}}\sum_{f\in\mathcal{F}_k} \mathbf{1}_{\mathcal{P}_p^{to-do}(\tilde{t})}(\mathcal{V}^{fk})|v_p^{fk}(\tilde{t})|b^{fk} \le B_p, \qquad \forall p \in \mathcal{U}\cup M. \tag{6}$$

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

A. AGE OF INFORMATION

The AoI metric is adopted to quantify the freshness of the received packet at the destination. As mentioned, AoI is defined as the time elapsed from the generation of the last received packet from that service. This subsection will show how to formulate and quantify this metric regarding the network parameters.

For each service packet-flow $\Upsilon_u^k(t)$, let $\tau_u^k(t)$ denote the expectation of IoT access network delay with respect to transmission rate $R_{nu}(t)$,

$$\begin{aligned}\tau_u^k(t) &= E_{R_{nu}(t)}[\tau_{nu}^k(t)],\\ \tau_{nu}^k(t) &= D_{nu}/C + \Lambda_{nu}^k/R_{nu}(t)\end{aligned} \tag{7}$$

where D_{nu} is distance between IoT node $n \in \mathcal{N}$ and UAV $u \in \mathcal{U}$ and Λ^k is the packet length of service type $k \in \mathcal{S}$.

If t_u^k represents the time elapsed from the beginning of the time slot t in which a packet from service packet-flow $\Upsilon_u^k(t)$ has arrived, the AoI of service packet-flows at the UAV nodes can be calculated as,

$$\Theta_u^k(t) = \begin{cases}\tau_u^k + T - t_u^k, & \text{if } \alpha_u^k(t) = 1,\\ \Theta_u^k(t-1) + T, & \text{if } \alpha_u^k(t) = 0,\end{cases} \qquad \forall k \in \mathcal{S}, u \in \mathcal{U}. \tag{8}$$

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, $\alpha_u^k(t) = 0$. As mentioned above, t_u^k represents the time elapsed from the beginning of the time slot t in which a packet from packet-flow $\Upsilon_u^k(t)$ has arrived, so referring to the first equation of (8), $T - t_u^k$ is the time passed from receiving that last packet of packet-flow $\Upsilon_u^k(t)$ in the case that in time slot t a new packet from this packet-flow is received; As a result, $\tau_u^k + T - t_u^k$ would the time elapsed since the generation of the last received packet from packet-flow $\Upsilon_u^k(t)$ at time slot t, i.e., the AoI of this packet-flow at time slot t. 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 which leads to the second equation of (8). On each packet of service packet-flow $\Upsilon_u^k(t)$, the set $\mathcal{F}^k$ of VNFs (processing functions) should be performed, the subset $\Delta_u^k(\tilde{t})$ in UAV and the remaining VNFs $\bar{\Delta}_u^k(\tilde{t})$ at the MEC server. The processing time of every packet of this flow will be,

$$\begin{aligned}\Phi_u^k(t) &= \left(\left\lceil \frac{\sum_{\mathcal{V}\in\Delta_u^k(\tilde{t})}\theta_u^{\mathcal{V}}}{T}\right\rceil T + \tau_{uM}^k(t) + \left\lceil \frac{\sum_{\mathcal{V}\in\bar{\Delta}_u^k(\tilde{t})}\theta_M^{\mathcal{V}}}{T}\right\rceil T\right),\\ \tau_{uM}^k(t) &= D_{uM}/C + \Lambda^k(\delta_u^k)/R_{uM}(t),\\ &\qquad \forall k \in \mathcal{S}, u \in \mathcal{U}.\end{aligned} \tag{9}$$

where $\theta_u^{\mathcal{V}}$ and $\theta_M^{\mathcal{V}}$ is the run time of VNF $\mathcal{V}$ when it is performed on UAV u and the MEC server M, respectively; $\Lambda^k(\delta_u^k)$ is the packet length of service type k after performing the δ_u^kth VNF of the chain, and, $\tau_{uM}^k(t)$ is total transmission delay between UAV u and the MEC server M for transmission of a single packet consists of propagation delay and transmission delay.

Let binary variable $\beta_u^k(\tilde{t}) \in \{0, 1\}$ indicate whether the VNF scheduling for service packet-flow $\Upsilon_u^k(t)$ in a single round of VNF scheduling $\tilde{t}$ was successful, then, the AoI at the Local server will be,

$$\Pi_u^k(t) = \begin{cases}\Theta_u^k(t) + \Phi_u^k(t), & \text{if } \beta_u^k(\tilde{t}) = 1,\\ \Pi_u^k(t-1) + \widetilde{T}, & \text{if } \beta_u^k(\tilde{t}) = 0,\end{cases} \qquad \forall k \in \mathcal{S}, u \in \mathcal{U}. \tag{10}$$

For every unsuccessful VNF scheduling, the AoI of that service packet flow increases by $\widetilde{T}$.

To support our time-sensitive application, we need to guarantee that the AoI for each service packet flow is less than a predefined threshold all the time. This point is reflected in the following remark.

Remark 1: For each service packet-flow $\Upsilon_u^k(t)$, we have this ***strict condition*** *that the age of information $\Pi_u^k(t)$ in relation (10) be less than a predefined threshold ξ,*

$$\Pi_u^k(t) < \xi, \qquad \forall k \in \mathcal{S}, u \in \mathcal{U}. \tag{11}$$

B. ENERGY CONSUMPTION

In this subsection, the total energy consumption for processing and delivering a single packet of each service packet flow to the local server is formulated. For the uplink direction, the energy consumption can be calculated as

$$\begin{aligned} \mathcal{E}^{UL}(t) &= \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{S}}\sum_{u\in\mathcal{U}} s_n^k p_{nu} \tau_{nu}^k(t), \\ \tau_{nu}^k(t) &= \Lambda^k(0)/R_{nu}(t), \end{aligned} \tag{12}$$

where $\tau_{nu}^k(t)$ denotes the transmission time between IoT node n, with service type k, and UAV u at TS t, and $\Lambda^k(0)$ is the packet length of service type k before performing any processing functions. Similarly, for the energy consumption in the downlink direction, we have,

$$\begin{aligned} \mathcal{E}^{DL}(t) &= \sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{S}} p_{uM} \tau_{uM}^k(t), \\ \tau_{uM}^k(t) &= \Lambda^k(\delta_u^k)/R_{uM}(t), \end{aligned} \tag{13}$$

where $\tau_{uM}^k(t)$ denotes the transmission time of packets belong to service type k from UAV u to the MEC server M, and, $\Lambda^k(\delta_u^k)$ is the packet length of service type k after performing the δ_u^kth VNF of the chain.

Finally, if $\psi_u^{\mathcal{V}}$ and $\psi_M^{\mathcal{V}}$ denote the power that UAV u and the MEC server M which hosts the VNF $\mathcal{V}$ consumes to run this VNF on each packet of this service packet-flow, respectively, then, the total energy required for performing the VNFs on a single packet of each service type can be calculated as follows:

$$\mathcal{E}^{\mathcal{V}} = \sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{S}}\Big(\sum_{\mathcal{V}\in\Delta_u^k(\tilde{t})} \psi_u^{\mathcal{V}}\theta_u^{\mathcal{V}} + \sum_{\mathcal{V}\in\bar{\Delta}_u^k(\tilde{t})} \psi_M^{\mathcal{V}}\theta_M^{\mathcal{V}}\Big). \tag{14}$$

Th exact value of $\psi_u^{\mathcal{V}}$ and $\psi_M^{\mathcal{V}}$ parameters depend on the hosting processing power including CPU power, efficiency, and so on. Here, for the sake of brevity, in the math formulation we have considered these affecting parameters in a single parameter. In the deployment phase, they should be determined and taken into account depending on the exact specifications of the hosting machine.

Using (13)-(14), the total energy consumption of the network to process a single packet of all service types for all UAVs will be,

$$\mathcal{E}^{total}(t) = \mathcal{E}^{UL}(t) + \mathcal{E}^{DL}(t) + \mathcal{E}^{\mathcal{V}}(t). \tag{15}$$

IoT nodes' are energy-limited and UAVs also are battery operated; hence, their available energy to compute and communicate is limited. It is important to note that, in certain circumstances, the VNFO may place some of the VNFs to be processed locally by IoT nodes. Thus, the total long-term cumulative value of their energy consumption should be minimized as well. On the other hand, as declared in *Remark 1*, there is a strict condition (*constraint*) on the AoI value of the service packet-flows. Therefore, the following constrained optimization problem should be solved.

Problem 1 (Constrained Energy-Efficient VNFO): Considering the service packet-flow requirements, UAVs/MEC-server available resources, and the condition of the access network, an energy-efficient VNFO solution is needed that guarantees the required instantaneous AoI value at the local server:

$$\begin{aligned} &\underset{\delta_u^k,\forall u\in\mathcal{U},k\in\mathcal{S}}{Minimize} \sum_{t=0}^{\infty}\mathcal{E}^{total}(t), \\ &s.t. \quad \Pi_u^k(t) < \xi, \qquad \forall k \in \mathcal{S}, u \in \mathcal{U}, \\ &\qquad (5)-(6). \end{aligned} \tag{16}$$

It is worth mentioning that $\alpha_u^k(t)$ in (8) and $\beta_u^k(\tilde{t})$ in (10) are the results of the policy δ_u^k which is the expected output of a solution for solving *Problem 1*. These two variables $\alpha_u^k(t)$ and $\beta_u^k(\tilde{t})$ are defined to formulate the AoI mathematically but they are not independent variables actually. δ_u^k will be the policy of VNF placement that will be deployed by NFVO and will determine the actual value of $\alpha_u^k(t)$ and $\beta_u^k(\tilde{t})$ in the deployment phase.

In each VNFO-level time slot $\widetilde{T}$, the orchestrator sequentially decides on the chain of NFVs of each service flow of UAVs. Markov Decision Process (MDP) is a powerful framework for mathematically formulating and studying this type of problem for a class of sequential decision-making problems. 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 [38]. However, according to *Remark 1*, we have a strict condition on the AoI at the local server that should be guaranteed. To incorporate this condition, in the next section, we reformulate *Problem 1* under the framework of the constrained Markov decision process (CMDP). CMDP is an extension of the standard MDP where the purpose is optimizing an objective while explicitly satisfying some constraints in terms of auxiliary costs [23].

V. DEC-CMMDP FORMULATION

A CMDP is modeled as a tuple $\langle S, s_0, A, T, r, c, c_0\rangle$, where S is the state space, $s_0 \in S$ is the initial state, and $\forall(\acute{s}, s) \in S, a \in A$, $A(s, a)$ is action space, $T(\acute{s}|s, a), \forall(\acute{s}, s) \in S, a \in A$ is the transition probability function, $r(s, a)$ is the immediate reward function, $c(s, a)$ is the immediate cost and c_0 is an upper bound on the expected cumulative cost. In a CMDP, the agent based on the current state s uses a policy π to perform an action $a(s)$ that

determines the next state $\acute{s}$ according to the transition probability of $T(\acute{s}|s, a)$. The best policy π among feasible policies is the one that maximizes the expected discounted cumulative reward $\mathbb{E}\{\sum_{\tilde{t}=0}^{\infty} \gamma^{\tilde{t}} r(s, a)|\pi\}$ while keeps the discounted cumulative cost $\mathbb{E}\{\sum_{\tilde{t}=0}^{\infty} \gamma^{\tilde{t}} c(s, a)|\pi\}$ less than the predefined threshold c_0, in a sequence of decision-making instances $\tilde{t} = 0, 1, 2, \ldots$, where, γ is the discounting factor.

Indeed, in our scheme, the VNFO is the decision maker. Although a centralized VNFO may be optimal, it requires a large volume of communication transactions to share the local information with the centralized VNFO to make it aware of the exact current state of the network. Another drawback of a fully centralized VNFO is that it will not be scalable and would be a single point of failure from the processing and communication viewpoint. Therefore, in this paper, we consider the constrained multi-agent MDP (CMMDP) scheme [39] in which multiple agents or decision-makers exist. In the literature, this scheme is also referred to as a stochastic game [40]. In the CMMDP case, VNFO is implemented as a multi-agent VNFO, where there is coordination among the agents (or actors) that are the UAVs in our system model. In the proposed multi-agent VNFO, a CMMDP is formally defined as a tuple with the following definition.

Definition 2 (CMMDP Model): A CMMDP $\mathbb{G}$ with a set $\mathcal{U}$ of U agents is defined as a tuple $\mathbb{G} = \langle \mathcal{U}, S, \mathbb{A}, T, \mathbb{R}, \mathbb{C}, c_0 \rangle$, where

- *S is the finite set of entire environment states,*
- *$\mathbb{A} = \prod_{u\in\mathcal{U}} A_u$ is the joint action space, where A_u is the set of actions 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 $T(\acute{s}|s, \boldsymbol{a}(s))$ is defined as the transition probability from state s to $\acute{s}$ by doing joint action $\boldsymbol{a}(s)$,*
- *$R : \bigcup_{s\in\mathbb{S}} (s \times \mathbb{A}(s)) \rightarrow \mathcal{R}$ is reward functions, where $R(s, \boldsymbol{a}(s))$ is defined as the total reward received by all agents when the joint action $\boldsymbol{a}(s)$ is executed at the entire environment state s,*
- *$C : \bigcup_{s\in\mathbb{S}} (s \times \mathbb{A}(s)) \rightarrow \mathcal{R}$ is cost functions, where $C(s, \boldsymbol{a}(s))$ is defined as the total cost incurred by all agents when the joint action $\boldsymbol{a}(s)$ is executed at the entire environment state s,*
- *c_o is a predefined upper-bound on the expected cumulative cost: $\mathbf{E}\{\sum_{t=0}^{\infty} \gamma^t C(s, \boldsymbol{a}(s))|\pi\} < c_o$.*

By definition, in a CMMDP, the agents based on entire network state s and joint policy $\boldsymbol{\pi}$ perform independent actions $a_u^{\boldsymbol{\pi}} \in A_u$. The joint action $\boldsymbol{a}^{\boldsymbol{\pi}(s)} = \prod_{u\in\mathcal{U}} a_u^{\boldsymbol{\pi}(s)}$ determines the next entire environment state $\acute{s}$ according to the transition probability $T(\acute{s}|s, \boldsymbol{a}^{\boldsymbol{\pi}(s)})$. The total reward $R(s, \boldsymbol{a}^{\boldsymbol{\pi}(s)})$ and total cost $C(s, \boldsymbol{a}^{\boldsymbol{\pi}(s)})$ received by all agents is also based on the current entire-state s and joint action $\boldsymbol{a}^{\boldsymbol{\pi}(s)}$ performed by all U agents.

In a large-scale CMMDP, the requirement that all agents can observe everything is too restrictive [41]. As an alternative, the case where the state space is factored into per-agent sets, $S = \prod_{u\in\mathcal{U}} S_u$ is introduced. In factored-state CMMDP, which is called a decentralized CMMDP (Dec-CMMDP) [39], each agent $u \in \mathcal{U}$ conditions its own policy π_u only on locally observed state $s_u \in S_u$, receive reward $r_u(s, \boldsymbol{a}(s))$ and incur cost $c_u(s, \boldsymbol{a}(s))$ with required upper-bound of c_u^o. It is worth noting that in a Dec-CMMDP, the reward r_u and cost c_u each agent experiences depend on the entire network state s and joint action $\boldsymbol{a}(s)$. Whereas the local factored state s_u is the basis for choosing the action by the agent.

As declared in *Problem1*, the purpose of our problem is to find the best policy for VNF placement by determining the best *Placement Indices* δ_u^k for all service packet-flows at each state by the UAVs as distributed agents.

Let define the extended state $s_{u,\tilde{t}}^e = (\boldsymbol{s}_{-u,\tilde{t}}, \boldsymbol{a}_{-u,\tilde{t}}, s_{u,\tilde{t}})$, $\forall u \in \mathcal{U}$ as a combination of factored state of UAV itself, $s_{u,\tilde{t}}$, joint factored state of the other UAVs, $\boldsymbol{s}_{-u,\tilde{t}}$, and joint action of the other UAVs, $\boldsymbol{a}_{-u,\tilde{t}}$. In the following, where it does not cause ambiguity, we ignore the time index $\tilde{t}$. According to Bellman expectation equation [42], the action-value function $Q_{u,\tilde{t}}^{\boldsymbol{\pi}}(s_u^e, a_u)$, $\forall u \in \mathcal{U}$ is defined as the expected return starting from state s_u^e taking action a_u according to policy π_u, while the other agents are taken the joint action $\boldsymbol{a}_{-u}$ according to joint policy $\boldsymbol{\pi}_{-u}$,

$$Q_{u,\tilde{t}}^{\boldsymbol{\pi}}(s_u^e, a_u) = \mathbf{E}\Big\{r_u(s_u^e, a_u) + \gamma_u Q_{u,\tilde{t}}^{\boldsymbol{\pi}}(\acute{s}_u^e, \acute{a}_u)\Big\}, \tag{17}$$

where the action-value function decomposed into immediate reward $r_u(s_u^e, a_u)$ plus discounted action-value of the successor state $\acute{s}_u^e$, while the agent u will execute action $\acute{a}_u$; and γ is discount factor. With some mathematical manipulation, (17) can be written as,

$$\begin{aligned} Q_{u,\tilde{t}}^{\boldsymbol{\pi}}(s_u^e, a_u) &= r_u(s_u^e, a_u) + \gamma_u \sum_{\acute{s}_u^e} T(\acute{s}_u^e|s_u^e, a_u) V_{u,\tilde{t}}^{\boldsymbol{\pi}}(\acute{s}_u^e), \\ V_{u,\tilde{t}}^{\boldsymbol{\pi}}(\acute{s}_u^e) &= \sum_{\acute{a}_u} \pi_u(\acute{a}_u|\acute{s}_u^e) Q_{u,\tilde{t}}^{\boldsymbol{\pi}}(\acute{s}_u^e, \acute{a}_u) \end{aligned} \tag{18}$$

where, $V_{u,\tilde{t}}^{\boldsymbol{\pi}}(s_u^e)$, $\forall u \in \mathcal{U}$ is the value function at the extended state s_u^e and is defined as the expected discounted cumulative reward starting from state s_u^e following the joint policy $\boldsymbol{\pi}$. For an enough large value of $\tilde{t}$ ($\tilde{t} \rightarrow \infty$), 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 while the constraint in (16) is satisfied,

$$\begin{aligned} Q_{u,\tilde{t}}^{\boldsymbol{\pi}*}(s_u^e, a_u) &= \arg\max_{\pi_u} Q_{u,\tilde{t}}^{\boldsymbol{\pi}}(s_u^e, a_u), \\ &\text{s.t.} \quad \mathbf{E}\{\sum_{\tilde{t}=0}^{\infty} \gamma_u^{\tilde{t}} c_u(s_u^e, a_u)|\pi_u\} < c_u^o. \end{aligned} \tag{19}$$

Using (17) and (18), and some math operations (19) can be written as,

$$\begin{aligned} &Q_{u,\tilde{t}}^{\boldsymbol{\pi}*}(s_u^e, a_u) \\ &= r_u(s_u^e, a_u) + \gamma_u \sum_{\acute{s}_u^e} T(\acute{s}_u^e|s_u^e, a_u) \max_{\acute{a}_u} Q_{u,\tilde{t}}^{\boldsymbol{\pi}*}(\acute{s}_u^e, \acute{a}_u), \end{aligned}$$

$$\text{s.t.} \quad \mathbf{E}\{\sum_{\tilde{t}=0}^{\infty} \gamma_u^{\tilde{t}} c_u(s_u^e, a_u)|\pi_u\} < c_u^o. \tag{20}$$

Determining an optimal policy for a Dec-CMMDP is NEXP-complete [24] with no straightforward solution. A few efforts, [43], [44], [45], 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 (20). One of those that are proposed for solving partially observable stochastic games (POSG) is [44], where a class of POSGs is characterized by symmetry across players (agents) in terms of cost and state dynamics. Inspired by this work, we claim the following lemma which helps to develop our proposed algorithm to solve *Problem 1*.

Lemma 1: Problem (16) is a symmetric Dec-CMMDP. With being a symmetric CMMDP, 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 $\pi_{\acute{u}} = \pi_u$, $\forall \acute{u} \neq u$.

In section VII we will prove this lemma.

Lemma 1 implies $\{\pi_u^*\}_{u=1}^{U} = \pi^*$. Although this result is promising, as we will utilize this result in developing our proposed method in Section VI, from the implementation viewpoint finding the best policy in a distributed form is still challenging. On the other hand, the recursive form of (20) provides the possibility of exploiting iterative solutions like dynamic programming algorithm; however, for our *multi-agent* CMDP case, they are inefficient [39] and require the full information of the model which makes it impractical. Therefore, we formulate the problem of finding optimal policy (20) for *Problem 1* as a *model-free* RL problem.

Considering the limited available resources of processing nodes in (5)-(6), the objective of *Problem 1* is minimizing the total energy consumption $\mathcal{E}^{total}$ defined in (15), therefore the reward function $r_u(s_{u,t}^e, a_{u,t})$ defines as,

$$r_{u,\tilde{t}}(s_u^e, a_u) = \delta^{\mathcal{V}} \times \zeta_u(\tilde{t}) - \delta^{\mathcal{E}} \times \mathcal{E}_u^{total}(\tilde{t}), \quad \forall \tilde{t}, u \in \mathcal{U}, \tag{21}$$

where, $\zeta_u(\tilde{t}) \in [0, K]$ is the number of packet-flows for which the VNF scheduling result has satisfied (5)-(6) conditions; $\delta^{\mathcal{V}}$ and $\delta^{\mathcal{E}}$ are the normalization coefficients for VNF scheduling result and energy consumption terms, respectively.

On the other hand, the policy should guarantee the required instantaneous AoI value at the local server; hence, from (16) the constraint $c_{u,\tilde{t}}(s_u^e, a_u)$ defines as,

$$c_{u,\tilde{t}}(s_u^e, a_u) = \underset{k\in\mathcal{S}}{Max}\{\Pi_u^k(\tilde{t})\} - \xi, \qquad \forall \tilde{t}, u \in \mathcal{U}, \tag{22}$$

According to the above discussion, *Problem 1* can be reformulated as an RL problem as follows.

Problem 2 (Cumulatively Constrained RL Problem): Consider a Dec-CMMDP problem with U agents, unknown transition function, reward function $\{r_{u,\tilde{t}}(s_u^e, a_u)\}_{u=1}^{U}$ defined in (21) and cost function $\{c_{u,\tilde{t}}(s_u^e, a_u)\}_{u=1}^{U}$ defined in (22), the objective is to find the optimal policies $\{\pi_u^*\}_{u=1}^{U}$ for the following infinite horizon constraint optimization problem:

$$\underset{\pi_u}{Maximize} \quad \mathbf{E}\Big\{\sum_{\tilde{t}=0}^{\infty} \gamma^{\tilde{t}} r_{u,\tilde{t}}(s_u^e, a_u)\Big|\pi_u\Big\}, \quad \forall \tilde{t}, u \in \mathcal{U},$$

$$s.t. \quad \mathbf{E}\Big\{\sum_{\tilde{t}=0}^{\infty} \gamma^{\tilde{t}} c_{u,\tilde{t}}(s_u^e, a_u)\Big|\pi_u\Big\} < 0. \tag{23}$$

However, there is a point in fully modeling our problem as a Dec-CMMDP. The AoI constrained in (16) is instantaneous and requires that the instantaneous value of the AoI (represented by constraint function in (22)) at all time instances satisfies this constraint. We will return to this point in the next section, where we introduce our proposed framework for solving *Problem 2*.

VI. ITERATIVE CONSTRAINED FEDERATED-DQN FRAMEWORK

In this section, we introduce our proposed Iterative Constrained Federated DQN (IC-FDQN) algorithm as depicted in Fig. 3.

A. AUGMENTED LAGRANGIAN BASED SURROGATE OBJECTIVE FUNCTION

Even though Dec-CMMDP, and its model-free form declared in *Problem 2*, are an appropriate choice for modeling *Problem 1*, a cumulatively-constrained MDP does not guarantee the associated instantaneous constraint. Using CMMDP terminology, according to (22), we require to have

$$\mathbf{E}\Big\{c_{u,\tilde{t}}(s_u^e, a_u)\Big|\pi_u\Big\} = \mathbf{E}\Big\{\underset{k\in\mathcal{S}}{Max}\{\Pi_u^k(\tilde{t})\} - \xi\Big|\pi_u\Big\} < 0, \qquad \forall \tilde{t}, u \in \mathcal{U}. \tag{24}$$

whereas the constraint in the CMMDP model defined in *Definition 2* is cumulative. Hence, there is a gap here.

We base our proposed method on classical Lagrangian dual optimization. Even though it is shown that for infinite-horizon RL problems with cumulative constraints, under certain regularity conditions, there is no duality gap despite their non-convex nature [46], the same statement is no longer true for the MDPs with instantaneous constraints [33]. Therefore, we will adapt a newly-proposed extended version of the Augmented Lagrangian (AugLag) method [33] to develop a scheme for obtaining the optimal policy for *Problem 2* while satisfying the instantaneous constraint of (24) in a decentralized manner.

Li et al. [33] consider condition under which strong duality holds for *Problem 2 with instantaneous constraint, and then design a new rewarding mechanism for* a new unconstrained surrogate objective function. It is proven analytically and demonstrated empirically [33] that this method results in a high-quality policy with smaller constraint violations than the primal-dual method. We will return to this subject in Section VII where we discuss the convergence of the proposed algorithm. According to this method, for *Problem 2* subject to instantaneous constraint (24) the surrogate reward function $\tilde{r}_{u,\tilde{t}}(s_u^e, a_u)$ for *unconstrained* surrogate MMDP

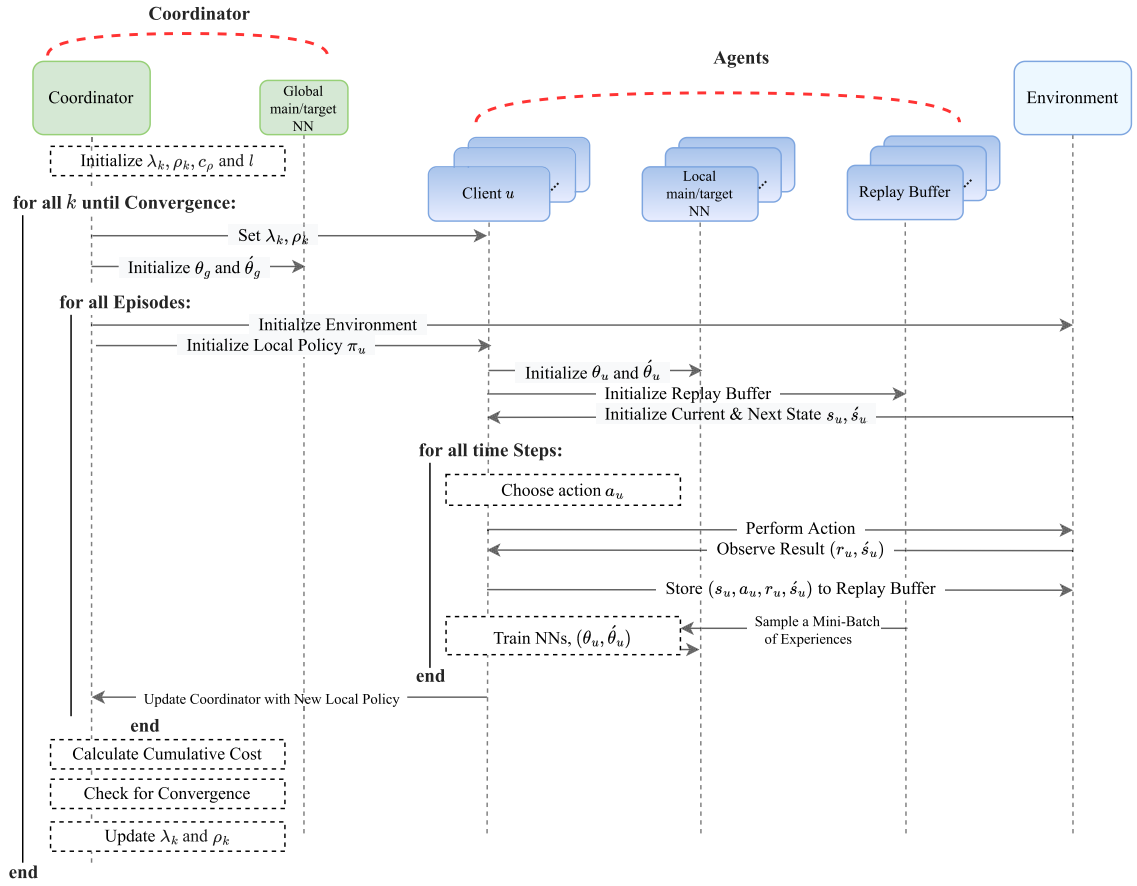

FIGURE 3. **IC-FDQN Algorithm steps and parties.**

problem would be,

$$\tilde{r}_{u,\tilde{t}}\left(s_u^e, a_u\right) = r_{u,\tilde{t}}\left(s_u^e, a_u\right) + \lambda X^+\left(c_{u,\tilde{t}}\left(s_u^e, a_u\right)\right) - \frac{\rho}{2} X^+\left(c_{u,\tilde{t}}\left(s_u^e, a_u\right)\right)^2. \tag{25}$$

From (25), the reward function $r_{u,\tilde{t}}\left(s_u^e, a_u\right)$ is augmented with two additional terms: The first one $\lambda X^+\left(c_{u,\tilde{t}}\left(s_u^e, a_u\right)\right)$ is inspired by traditional Lagrangian dual optimization, where λ is the Lagrangian multiplier. The second term $\frac{\rho}{2} X^+\left(c_{u,\tilde{t}}\left(s_u^e, a_u\right)\right)^2$ is inspired by quadratic penalty function with penalty parameter $\frac{\rho}{2}$ [32].

Problem 3 (Surrogate Unconstrained RL Problem): Consider a Dec-MMDP problem with U agents, unknown transition function and reward function $\left\{\tilde{r}_{u,\tilde{t}}\left(s_u^e, a_u\right)\right\}_{u=1}^U$, *the objective is to find the optimal policies* $\left\{\pi_u^*\right\}_{u=1}^U$ *for the following infinite horizon unconstrained optimization problem*:

$$\underset{\pi_u}{Maximize} \quad \mathbf{E}\left\{\sum_{\tilde{t}=0}^{\infty} \gamma^{\tilde{t}}\left[r_{u,\tilde{t}}\left(s_u^e, a_u\right) + \tilde{r}_{u,\tilde{t}}\left(s_u^e, a_u\right)\right] \middle| \pi_u\right\},$$

$$\tilde{r}_{u,\tilde{t}}\left(s_u^e, a_u\right) = \lambda X^+\left(c_{u,\tilde{t}}\left(s_u^e, a_u\right)\right) - \frac{\rho}{2} X^+\left(c_{u,\tilde{t}}\left(s_u^e, a_u\right)\right)^2,$$

$$r_{u,\tilde{t}}\left(s_u^e, a_u\right) = \delta^{\mathcal{V}} \times \zeta_u(\tilde{t}) - \delta^{\mathcal{E}} \times \mathcal{E}_u^{total}(\tilde{t}),$$

$$c_{u,\tilde{t}}\left(s_u^e, a_u\right) = \underset{k \in \mathcal{S}}{Max}\left\{\Pi_u^k(\tilde{t})\right\} - \xi. \tag{26}$$

The non-negative function X^+ in (25) is crucial; otherwise, it is not generally true that an optimal policy for (26) is also optimal for (23) while satisfying instantaneous constraint (24) [33]. In the following subsection, we propose our Federated-based iterative learning algorithm for solving (26).

B. PROPOSED IC-FDQN METHOD

In *Problem 3*, besides optimal policies, $\left\{\pi_u^*\right\}_{u=1}^U$, λ and ρ are also two design parameters that should be optimally determined. For a fixed value of (λ, ρ), *Problem 3* would be a multi-agent MDP. Hence, the optimal solution can be determined through two *nested* iterative loops, where the *inner-loop* determines the optimal policies $\left\{\pi_{u,i}^*\right\}_{u=1}^U$ for a fixed value of $\left(\lambda^{(i)}, \rho^{(i)}\right)$, then, the *outer-loop* over $\left(\lambda^{(i)}, \rho^{(i)}\right)$ aims to determine the optimal values of λ and ρ from an initialized point of $\lambda^{(0)}$ and $\rho^{(0)}$, respectively, [33],

$$\lambda^{(i+1)} = \lambda^{(i)} - l\left(\boldsymbol{E}\left[\sum_{\tilde{t}=0}^{\infty} \gamma^{\tilde{t}} X^+\left(c_{u,\tilde{t}}\left(s_u^e, a_u\right)\right) \middle| \pi_{u,i}\right]\right), \tag{27}$$

$$\rho^{(i+1)} = c_\rho \rho^{(i)}. \tag{28}$$

where, $l \in [1, \infty)$ and $c_\rho \in \mathbb{R}^+$ are the increasing rate of the quadratic penalty coefficient and dual ascent stepsize, respectively.

According to *Lemma 1*, the optimal policy among the agents is the same, an excellent promotion to adapt Deep FL in our proposed scheme. Among decentralized methods, the multi-agent solution needs a large volume of communication overhead between the agents to share their local observations. It does not fully utilize the potential of *Lemma 1* well. 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.

As illustrated in Fig. 3, to estimate Q-value functions (17), deep reinforcement learning (DRL) is deployed, where deep neural networks (DNNs) are used as the function approximators to predict the Q-values. The Q-function estimated by the neural network (NN) in each agent u is represented by $\left\{Q^{\boldsymbol{\pi}}_{u,\tilde{t}}\left(s^e_{u,\tilde{t}}, a_{u,\tilde{t}}; \boldsymbol{\theta}_{u,\tilde{t}}\right)\right\}_{u=1}^{U}$, where the parameter $\boldsymbol{\theta}_{u,\tilde{t}}$ represents the weights of the NN. The updated value of $\boldsymbol{\theta}_{u,\tilde{t}}$ is used to train the NN and approximate the actual values of $Q^{\boldsymbol{\pi}}_{u,\tilde{t}}$ [47], [48]. Let's define the loss function $L\left(\boldsymbol{\theta}_{u,\tilde{t}}\right)$ as the expectation value of the mean squared error of the estimated Q-value $Q^{\boldsymbol{\pi}}_{u,\tilde{t}}\left(s^e_{u,\tilde{t}}, a_{u,\tilde{t}}; \boldsymbol{\theta}_{u,\tilde{t}}\right)$ from the target value $y_{u,\tilde{t}}$ [47],

$$L\left(\boldsymbol{\theta}_{u,\tilde{t}}\right) = \boldsymbol{E}\Big[\left(y_{u,\tilde{t}} - Q^{\boldsymbol{\pi}}_{u,\tilde{t}}\left(s^e_{u,\tilde{t}}, a_{u,\tilde{t}}; \boldsymbol{\theta}_{u,\tilde{t}}\right)\right)^2\Big], \tag{29}$$

where, $y_{u,\tilde{t}} = r_{u,\tilde{t}} + \gamma \times \underset{a_{u,\tilde{t}+1}}{argmax}\, Q^{\boldsymbol{\pi}}_{u,\tilde{t}+1}\left(s^e_{u,\tilde{t}+1}, a_{u,\tilde{t}+1}; \boldsymbol{\theta}_{u,\tilde{t}}\right)$ and $a_{u,\tilde{t}+1}$ indicates the agent's action generated by the DNN at $\tilde{t}+1$, given the state $s^e_{u,\tilde{t}+1}$.

At each iteration, the deep Q-function approximator is trained to learn the best estimate of the Q-function by minimizing the loss function $L\left(\boldsymbol{\theta}_{u,\tilde{t}}\right)$. To improve the stability of the algorithm and cope with sample correlation, as depicted in Fig. 3, two novel techniques, namely Fixed Target Network [49], and Experience Replay Buffer [50] are deployed, respectively. Utilizing these two techniques, the loss function $L\left(\boldsymbol{\theta}_{u,\tilde{t}}\right)$ can be written as

$$L\left(\boldsymbol{\theta}_{u,\tilde{t}}\right) = \boldsymbol{E}_D\Big[\Big(r_{u,\tilde{t}} + \gamma \times \underset{a_{u,\tilde{t}+1}}{argmax}\, Q^{\boldsymbol{\pi}}_{u,\tilde{t}+1}\left(s^e_{u,\tilde{t}+1}, a_{u,\tilde{t}+1}; \acute{\boldsymbol{\theta}}_{u,\tilde{t}}\right) - Q^{\boldsymbol{\pi}}_{u,\tilde{t}}\left(s^e_{u,\tilde{t}}, a_{u,\tilde{t}}; \boldsymbol{\theta}_{u,\tilde{t}}\right)\Big)^2\Big], \tag{30}$$

where $\acute{\boldsymbol{\theta}}_{u,\tilde{t}}$ denotes the target network parameters, and the expectation E_D is taken over the randomly selected mini-batches of samples from the replay buffer D. As it is illustrated in Fig. 3, we have two main entities, the set $\mathcal{U} = \{1, \ldots, U\}$ of UAVs that are our distributed agents or, in FL terminology, the clients, and the coordinator that in our model is a local server (MEC-server). FL allows the UAVs (clients) to train a shared global model parameterized by $\boldsymbol{\theta}_g$ that is an exact copy of the clients' local model $\left\{\boldsymbol{\theta}_u\right\}_{u=1}^{u=U}$ using their own local observations $\left\{D_u\right\}_{u=1}^{u=U}$, while the original data have remained at UAVs. After local training, clients 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 clients, the local server can enhance the training performance without significant communication overhead as it just needs an update of the local model parameters, not the clients' 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 transmits it to the clients. Upon receiving $\boldsymbol{\theta}_{g,0}$, during VNF-scheduling time slots $\tilde{t}$ the clients interact with environment and train their local model $\left\{\boldsymbol{\theta}_{u,\tilde{t}}\right\}_{u=1}^{u=U}$ using their own local observations $\left\{D_{u,\tilde{t}}\right\}_{u=1}^{u=U}$ by minimizing a loss function $\left\{L_u\left(\boldsymbol{\theta}_{u,\tilde{t}}\right)\right\}_{u=1}^{u=U}$,

$$\boldsymbol{\theta}^*_{u,\tilde{t}} = \arg\min_{\pi_u} L_u\left(\boldsymbol{\theta}_{u,\tilde{t}}\right),\ \forall u \in \mathcal{U} \tag{31}$$

Then, the clients upload their local update on $\left\{\boldsymbol{\theta}_{u,\tilde{t}}\right\}_{u=1}^{u=U}$ to the coordinator for aggregation.

2) MODEL AGGREGATION

After collecting the clients' local model updates, $\left\{\boldsymbol{\theta}_{u,\tilde{t}}\right\}_{u=1}^{u=U}$, the next step is aggregating them into a new version of the global model, $\boldsymbol{\theta}_{g,\tilde{t}+1}$, which is performed by coordinator through averaging among the agent's contributions,

$$\boldsymbol{\theta}_{g,\tilde{t}+1} = \sum_{u=1}^{U} \omega_u \boldsymbol{\theta}_{u,\tilde{t}}. \tag{32}$$

where ω_u represents the relative contribution of each agent on the global model.

According to *Lemma 1* and symmetry among the UAVs, the best choice for ω_u is $\omega_u = \frac{1}{U}$. After deriving a new update $\boldsymbol{\theta}_{g,\tilde{t}+1}$, the coordinator broadcasts it to all clients. Upon receiving the update from the coordinator, the clients upgrade their local model accordingly. Finally, for the locally-running DQN model at the clients, the local state space S_u, the action space A_u, and the reward function R_u are defined as follows:

- ***State:*** We define the client's state-space as a vector including 1) Computational, $\left\{c^{fk}\right\}_{f=1,k=1}^{F^k,K}$, and storage, $\left\{b^{fk}\right\}_{f=1,k=1}^{F^k,K}$, capacity requirement of packet flows belong to different services, 2) available CPU $\left\{C_p(t)|t = \tilde{t}\right\}_{p=1}^{U\cup M}$ and storage $\left\{B_p(t)|t = \tilde{t}\right\}_{p=1}^{U\cup M}$ of the processing nodes (UAVs and the MEC server), 3) Transmission rate in uplink $\left\{R_{nu}(t)|t = \tilde{t}\right\}_{n=1}^{N}$, and 4) Transmission rate in downlink $R_{uM}(t)|t = \tilde{t}$ at VNF-scheduling time $\tilde{t}$. Therefore, the state space S_u

will be,

$$S_u = \left\{ \{c^{fk}\}_{f,k}, \{b^{fk}\}_{f,k}, \{C_p\}_p, \{B_p\}_p, \{R_{nu}\}_n, R_{uM} \right\}. \quad (33)$$

- ***Action:*** The action space is possible choices of the placement index $\{\delta_u^k \in [0, |\mathcal{F}^k|]\}_{k=1}^K$ for all packet-flows $\{\Upsilon_u^k\}_{k=1}^K$:

$$A_u = \{\delta_u^k\}_k. \quad (34)$$

- ***Reward:*** According to (26), the reward function at VNF-scheduling time $\tilde{t}$ is given by:

$$\tilde{r}_{u,\tilde{t}} = \lambda X^+(c_{u,\tilde{t}}) - \frac{\rho}{2} X^+(c_{u,\tilde{t}})^2. \quad (35)$$

where $r_{u,\tilde{t}} = \delta^{\mathcal{V}} \zeta_u(\tilde{t}) - \delta^{\mathcal{E}} \mathcal{E}_u^{total}(\tilde{t})$, and $c_{u,\tilde{t}} = \underset{k \in \mathcal{S}}{Max} \{\Pi_u^k(\tilde{t})\} - \xi$.

VII. CONVERGENCE AND COMPUTATION COMPLEXITY ANALYSIS

The Convergence and computational complexity of the proposed algorithm are discussed here. The convergence analysis consists of two main parts, i.e., the convergence of the *outer-loop* to determine the optimal values (λ^*, ρ^*), and the *inner-loop* to determine the optimal policy $\{\pi_{u,i}^*\}_{u=1}^U$ for fixed intermediate values of $(\lambda^{(i)}, \rho^{(i)})$. The outer-loop is the extended version of the augmented Lagrangian method and its convergence is discussed in [32] and [33]. Therefore, in the following, we will present the convergence of the inner-loop and then we will determine the computational complexity of the whole algorithm.

A. INNER-LOOP CONVERGENCE

We prove the inner-loop convergence in two steps as follows. In step one, we justify our method in *subsection VI-B* on how to aggregate the local models reported by the UAVs as local agents, which is averaging. Then, we discuss the convergence of the aggregation method itself. For step one, first, we need to prove the *Lemma 1* introduced in Section V and repeated below for ease of reference.

Lemma 1: Problem (16) is a symmetric Dec-CMMDP. With being a symmetric CMMDP, 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 $\pi_{\acute{u}} = \pi_u$, $\forall \acute{u} \neq u$.

Proof: Inspired by [44], let's define a symmetric Dec-CMMDP model as follows.

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

(i) $\forall u, \acute{u} \in \mathcal{U}$, $A_u = A_{\acute{u}}$, $c_u^o = c_{\acute{u}}^o$ *and* $\gamma_u = \gamma_{\acute{u}}$.

(ii) $\forall\, u \in \mathcal{U}$, $a \in \mathbb{A}$, *and an arbitrary permutation function* $\sigma(.)$ *over all single actions* $\{a_u\}_{u=1}^{u=\mathcal{U}}$ *chosen by the agents*:

$$(a)\ r_u(s_u^e, \sigma(a)) = r_{\sigma(u)}(s_u^e, a),$$
$$(b)\ c_u(s_u^e, \sigma(a)) = c_{\sigma(u)}(s_u^e, a),$$
$$(c)\ T(\cdot|s_u^e, \sigma(a)) = T(\cdot|s_u^e, a).$$

Condition *(ii)* of *Definition 3* means the agents have the same statistical and decision model and are independent of each other. In our VNF-enabled SFC problem, as it is depicted in Fig. 1, each distributed agent $u \in \mathcal{U}$, has K packet flows $\{\Upsilon_u^k(t)\}_{k=1}^K$ belonging to different services each of which requires running a different VNF chain on its packets. The statistical model of each service type packet flow is the same for the agents. They independently decide how to break each VNF chain between the local server and itself while they follow the same objective function declared in *Problem 1*. Therefore, it can be inferred that the agents conceptually have the same decision-process model. Accordingly, without loss of generality, we can assume that the discount factor and the predefined upper-bound on the expected cumulative cost of the agents are the same: $\{\gamma_u\}_{u \in \mathcal{U}} = \gamma$, $\{c_u^o\}_{u \in \mathcal{U}} = c^o$. As a result, it can be inferred that *Problem 1* has the conditions of *Definition 3*, so it is a symmetric Dec-CMMDP. For such a class of symmetric multi-agent MDP, [44] demonstrated that for any $u, \acute{u} \in \mathcal{U}$, if $\pi_u = \pi_{\acute{u}}$, then, π_u is ϵ-best-response to $\boldsymbol{\pi}_{-u}$ if and only if $\pi_{\acute{u}}$ is ϵ-best-response to $\boldsymbol{\pi}_{-\acute{u}}$, 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. With being a symmetric CMMDP, 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 $\pi_{\acute{u}} = \pi_u$, $\forall \acute{u} \neq u$. This proves *Lemma 1*. ■

Lemma 1 implies $\{\pi_u^*\}_{u=1}^U = \pi^*$ which means despite what the optimal policy is, all the agents are the same. *This justifies our model aggregation described in subsection VI-B*, i.e. averaging over locally computed models by the agents. This method is perhaps the most popular method of doing aggregation on local models in the literature, called *FedAvg* [28], and its convergence has been extensively discussed in several papers [51], [52], [53]. Hence, in the following step two, we only discuss whether our problem and the proposed solution satisfy the required conditions for convergence.

To have *FedAvg* convergence guaranteed, basically, two particular conditions are assumed [51], [52], [53]: 1) data among the agents are i.i.d. and, 2) all the agents are participating in global model training. Although the latter condition in our case is true, however, the UAV's observations are not independent. This case is also dealt with in some recent works. In particular, Wan et al. [54], have analyzed the convergence rate of FL training in a more general setting with non-i.i.d. local observations, where they considered the joint impact of communication and training limitations while the condition of the agents to be i.i.d is released.

B. COMPLEXITY ANALYSIS

In this section, the computational complexity of the proposed algorithm in Fig. 3 is calculated. We consider the complexity

of the proposed algorithm in two phases, i.e. *Model Training* and *Action Selection* when the trained model will be deployed.

For each iteration of the outer-loop, we have the training of the local models by the agents (UAVs), and then, the aggregation of the local model by the MEC server to achieve the global model. The complexity of *local* model training by the agents is given by the summation of the complexity of action selection and the complexity of the back-propagation algorithm for each sample of the replay buffer multiplied by the mini-batch size. For the last multiplication by mini-batch size, note that in each iteration of the training process, the local agent randomly takes a mini-batch of the samples recorded by the agent in its local replay buffer.

For a fully connected NN with a fixed number of hidden layers and neurons in each hidden layer, the complexity of action selection is proportional to the summation of input and output size of the exploited NN [55], [56]. The input size of the NN equals the state space size which from (33) is given by $2FK + 2(U + 1) + N + 1$; and, the output size of the NN equals the action space size which from (34) is given by $K(F + 1)$. Therefore the complexity of action selection is $\mathcal{O}(FK)$. On the other hand, from [55] and [56], for a given sample of the replay buffer the computational complexity of the back-propagation is proportional to the product of the input and output size of the NN. Therefore, for our case of interest from the above calculations, it would be $\mathcal{O}(OF^2K^2)$, where, O is the batch size. As a result, assuming that the outer loop of the algorithm converges after a fixed number of iterations, $\mathcal{O}(1)$, and the aggregation process also increases in proportion to the number of the UAVs, U, the overall computational complexity of the training phase of the whole network is $\mathcal{O}(UOF^2K^2 + U)$, or indeed $\mathcal{O}(UOF^2K^2)$.

According to the above discussion, the complexity of the action selection of all U agents in the deployment phase is given by $\mathcal{O}(UFK)$.

VIII. PERFORMANCE EVALUATION

In this section, the performance of the proposed algorithm is evaluated. The performance results are compared for four different methods: 1) The proposed IC-FDQN method, 2) The centralized version of the proposed method; Iterative Constrained Centralized DQN (IC2DQN), 3) The multi-agent version of the proposed method; Iterative Constrained Multi-agent DQN (IC-MDQN), and 4) The heuristic method of *Minimum Delay*. In the IC2DQN method, it is assumed that the VNFO is the local server while it is aware of the entire network state s. In the IC-MDQN case, there is no coordination between the UAVs as distributed agents. Finally, the minimum delay method at each VNF scheduling time $\tilde{t}$ aims to select the action that leads to minimum end-to-end delay between the IoT nodes and the local server.

A. SIMULATION SETUP

The simulation environment is implemented in Python using OpenAI gym [57], a widely used tool for developing RL algorithms, and conducted in a computer with Intel(R) Core(TM) i7-10700 CPU 2.90 GHz and 64 GB RAM. Through simulations, the impact of the UAV's load variation on the performance in terms of Coefficient of Variation (CV) is evaluated. CV or Relative Standard Deviation is formally defined as $C_v = \frac{\sigma}{\mu} \times 100$, where σ and μ are the standard deviation and mean value of a statistical distribution [58]. A higher value for CV means a higher degree of variation to its mean. CV is a standardized *measure of dispersion* of a probability distribution or frequency distribution and widely used in engineering as a *quality and reliability assurance measure* [58], [59]. For a system under test, $20 < CV < 30$ is normal so for a reliable system it is expected that it will tolerate this variation in terms of the output performance. Whereas, $CV > 30$ is a strict condition [59], [60]. Hence, in the simulations, the performance of the proposed algorithm has been investigated for $CV \in \{10, 20, 30, 40\}$ and is compared with the baseline methods. In addition, the impact of increasing the number of IoT nodes on the performance of the proposed scheme in terms of average AoI, AoI-condition violation, and network energy consumption is evaluated. Simulation parameter settings are summarized in Table 2.

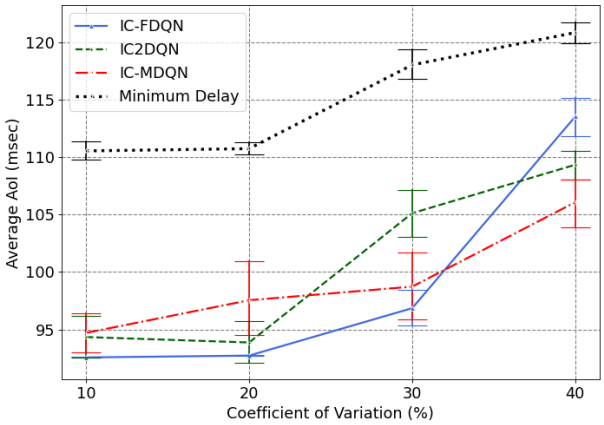

FIGURE 4. Average AoI versus different values of coefficient of variation, $CV \in \{10, 20, 30, 40\}$.

B. SIMULATION RESULTS

The effect of load variation on AoI and energy efficiency of the proposed method and baselines are presented in Fig. 4 to Fig. 6. In the implemented model, it is implicitly assumed that all the assigned VNFs to a processing node should be finished in a single round of VNF scheduling. In Fig. 4, the average AoI versus different values of CV is shown. As it is evidenced, by increasing the CV the average AoI increases too. This means an increase in load variation leads to an increase in AoI. This happens mainly because for bigger values of CV the agents have to find the best policy for a state of bigger dimension. However, compared to baseline methods, for different values of CV (except for $CV = 40$) average AoI of the proposed IC-FDQN method is the smallest, and the minimum delay method has the worst performance.

TABLE 2. Model parameters.

Parameter	Description	Value
U	Number of UAVs	3
N	Number of IoT devices	20, 25, 30, 35, 40, 45, 50, 55, 60
S	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	1.5 (for UAVs), 4.0 (for local server) (normalized)
b_fk	Storage capacity, required by service flows	Randomly between 0.8 to 1.2 (normalized)
B_p	Storage capacity in processing nodes	1.5 (for UAVs), 4.0 (for local server)
$\tilde{T}$	VNF-scheduling time period	200 msec
Λ^k	Service packet length	$k = 1$: 16KB; $k = 2$: 8KB
$\Lambda^k(\delta_u^k)$	Packet length after performing δ_u^kth VNF	$k = 1 \rightarrow f_1$: 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
$\delta^{\mathcal{V}}$	Normalization factor for VNF scheduling result	0.5
$\delta^{\mathcal{E}}$	Normalization factor for energy consumption	0.5
γ	DQN discount factor	0.9
ϵ	Exploration rate for DQN	0.001
N/A	Batch-size	16
N/A	Activation function	ReLU
N/A	IC-FDQN method DNN hidden layers	16, 32, 32, 16
N/A	IC2DQN method DNN hidden layers	32, 64, 64, 32
N/A	IC-MDQN method DNN hidden layers	16, 32, 32, 16
N/A	Number of episodes	1000
N/A	Target network update frequency	10
N/A	Experience replay buffer size	2000
N/A	DNN optimizer	Adam (LR=0.001)
N/A	DNN kernel initializer	He-Uniform

Generally speaking, similar results can be deduced from Fig. 5 and Fig. 6. Nevertheless, each method with respect to different values of *CV* behaves differently which needs to be discussed. For $CV = 10$ and $CV = 20$, i.e. when the variance of the UAV's load is small, the IC2DQN that has a centralized view over the network status reaches smaller AoI in comparison with the IC-MDQN in which the agents have just local observations. However, for the proposed IC-FDQN method, although the agents have their local observations (like IC-MDQN) but cooperatively contributed to the same model that results in a single global model which is able to reach the minimal AoI among the other baseline methods. This result is consistent with *Lemma 1* and confirms the fact that for asymmetric Dec-CMMDP, which is the case in this paper, the best policy for the agents is the same.

By increasing *CV* to 30 and more, IC-MDQN starts to surpass IC2DQN, as in IC2DQN the local server as a central agent tries to concurrently maximize the performance of all UAVs; for a fixed size of DNN, with the increase in *CV*, reaching out to the best policy for all the agents becomes more difficult until it would be infeasible after a certain amount of *CV* and performance degrades significantly. However, for $CV = 30$ both the proposed IC-FDQN method and IC-MDQN are able to reach a lower AoI in comparison with IC2DQN. The reason can be explained as follows. In IC-FDQN and IC-MDQN methods the agents behave independently, so they have more degrees of freedom in determining the local optimal action. That is why IC-MDQN provides better performance than IC2DQN as the *CV* increases. However, comparing IC-FDQN and IC-MDQN, the proposed IC-FDQN method which is FL-based still is able to reach the minimum AoI because the agents while acting independently, have the same model (a copy of the global model) that is cooperatively built upon the aggregation of local agents' observations. Nevertheless, for $CV = 40$, the story is different. As explained before, $CV = 40$ is not a normal case for an under-test system and is a strict condition. In this case, the problem from the global viewpoint of IC-FDQN and IC2DQN methods is not feasible, however, IC-MDQN could achieve better performance because the agents follow different policies and are greedy. As a result, some of the agents are able to minimize the local AoI which eventually leads to minimization of the average AoI. It should be mentioned that in calculating the AoI value of each agent, an upper limit of $\widetilde{T}$ is considered. Finally, the heuristic *Minimum Delay* method which tries to minimize the point-to-point delay comes with the largest value of AoI as it does not consider the end-to-end performance and has a local short-term target. In summary, for normal values of load variations, the proposed IC-FDQN method is able to achieve the minimum value for average AoI in comparison to the baselines for two reasons. First, the agents interactively

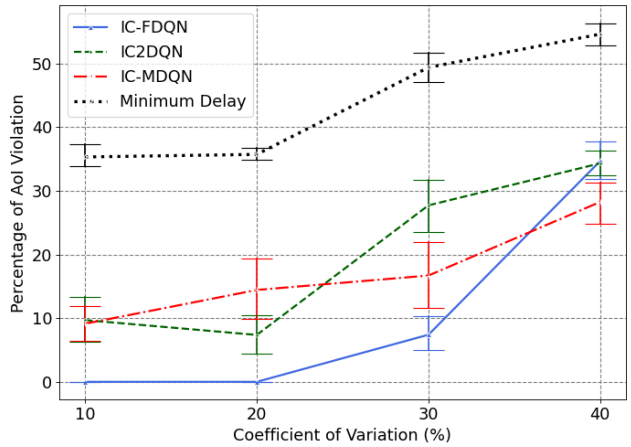

FIGURE 5. **Percentage of AoI violation versus different values of coefficient of variation, $CV \in \{10, 20, 30, 40\}$.**

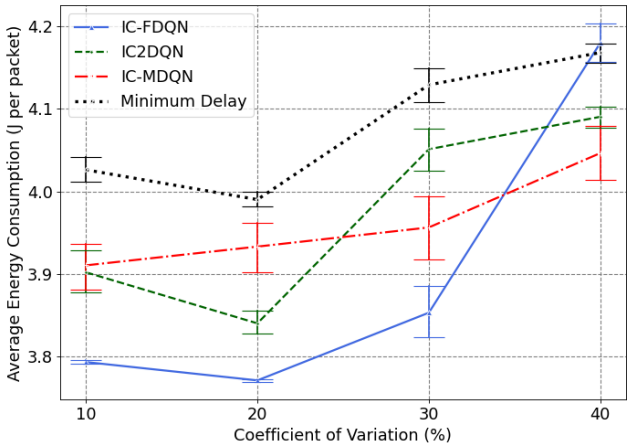

FIGURE 6. **Average energy consumption per packet versus different values of coefficient of variation, $CV \in \{10, 20, 30, 40\}$.**

cooperate to build the optimal global model based on their local observations. Second, although the model is the same, the agents act independently.

A similar discussion can be made for the results presented in Fig. 5 for the percentage of AoI violation and Fig. 6 for average energy consumption. The only point that is worth mentioning is the temporary reduction of the average energy consumption for IC-FDQN and IC2DQN when $CV = 20$. The point is that for a normal coefficient variation and for those methods that have a global viewpoint (all methods except IC-MDQN), independent load variation of the UAVs provides the central agent with the opportunity to manage the resources network-wide to minimize the energy consumption at some point that the load is low in some UAVs. However, the IC-MDQN is not able to use this opportunity because the agents behave greedily, independently, and based on local observations. Therefore, this reduction in the level of energy consumption for the case that the load of the UAVs is low leads to an overall minimization of energy consumption.

The outer-loop convergence is depicted in Fig. 7, where the optimal value for λ and ρ are determined through an iteration

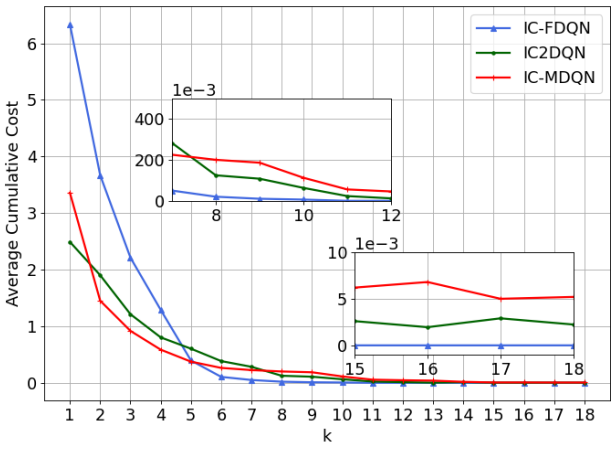

FIGURE 7. **Outer-loop convergence: Average cumulative cost vs iteration over k.**

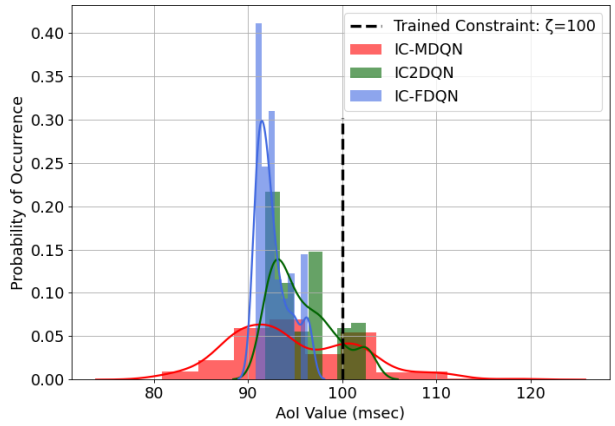

FIGURE 8. **Probability distribution function of AoI values.**

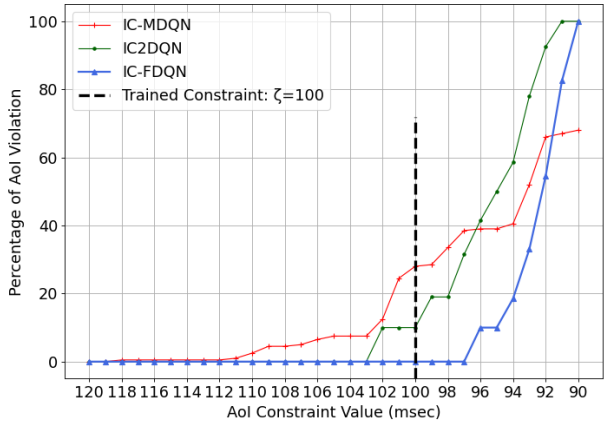

FIGURE 9. **The effect of changing the constraint value ξ after training.**

over k using (27) and (28). From (27), we define the *average cumulative cost* given by,

$$E\left[\sum_{\tilde{t}=0}^{\infty} \gamma^{\tilde{t}} X^{+}\left(c_{u,\tilde{t}}\left(s_u^e, a_u\right)\right) \middle| \pi_{u,i}\right]$$

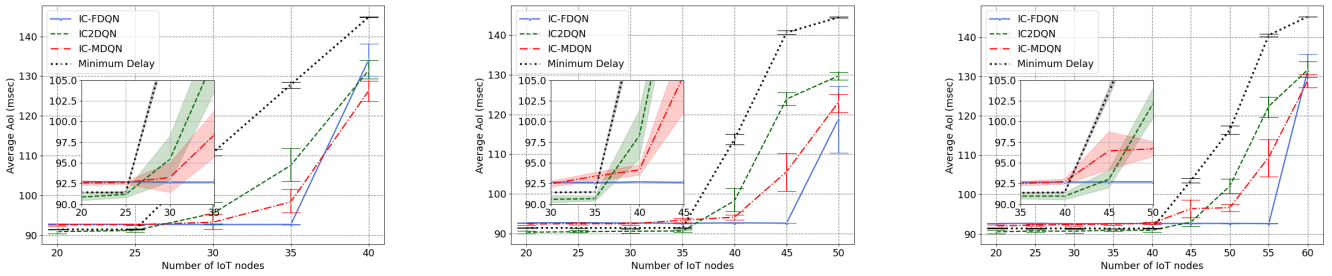

FIGURE 10. **The effect of increasing the number of IoT nodes on *AoI value* in different parameter settings for the available computing and storage capacity of processing nodes, while total available computing resources to processing nodes, C_p and B_p, (a) are like *Table 2*. (b) has been increased by a scale factor of 1.25 (c) has been increased by a scale factor of 1.5.**

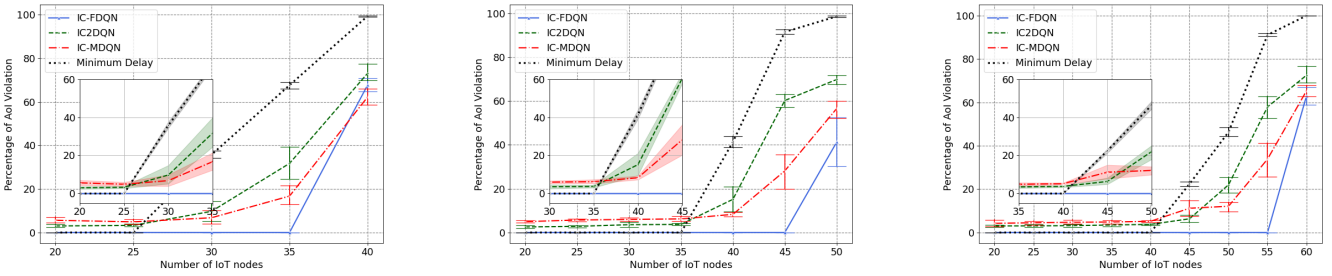

FIGURE 11. **The effect of increasing the number of IoT nodes on *percentage of AoI violation* in different parameter settings for the computing and storage capacity of processing nodes, while total available computing resources to processing nodes, C_p and B_p, (a) are like *Table 2*. (b) has been increased by a scale factor of 1.25 (c) has been increased by a scale factor of 1.5.**

Fig. 7 illustrates that for an optimal pair of (λ, ρ) in which the average cumulative cost converges, both IC2DQN and IC-MDQN have a residual cost while for the proposed. Whereas the proposed IC-FDQN method is able to reach the residual cost of zero for $k > 11$. In other words, the IC-FDQN method is able to successfully satisfy the constraint in all system states, as it is desired in Problem 1.

To further clarify, the histogram (the probability of occurrence) of the AoI values for the proposed method and baselines is determined by simulation and presented in Fig. 8. As it is evident, for the proposed IC-FDQN method the probability of occurrence for AoI values bigger than the constraint, i.e. $\xi = 100 msec$, is zero. Whereas, for IC2DQN and IC-MDQN the tail of the graph extends after the constraint that is not acceptable in CMDP context. Comparing IC-MDQN and IC2DQN, both methods violate the required instantaneous constraint on AoI values, but for the IC-MDQN method with a centralized view over the network, values are distributed over a wider range.

In Fig. 9, the effect of considering different values for AoI constraint value ξ while the training is performed for $\xi = 100 msec$ is investigated. This experiment illustrates how much each algorithm is robust against demanding smaller constraint values of ξ, i.e. more stringent conditions, while the NNs are trained for a bigger value of ξ. Note that in this experiment the NNs are *not* retrained for the new conditions. From this figure, the proposed IC-FDQN method is able to successfully satisfy the demands for lower values up to $\xi = 97 msec$. In other words, even though the algorithm is trained to provide instantaneous AoI values less than $\xi = 100 msec$, the algorithm is able to satisfy the more stringent conditions when ξ is a number belongs to range [97, 100]. However, IC-MDQN and IC2DQN methods are not able to even satisfy the trained constraint value of $\xi = 100 msec$.

In Fig. 10(a) to Fig. 10(c) the effect of increasing the number of IoT nodes is illustrated. For a fixed value of processing resources, as the number of IoT nodes increases the achievable AoI increases as well until for a certain number of IoT nodes in which the problem would be infeasible and AoI values increase significantly. In *scenario 2* and *scenario 3*, the processing node's total resources, C_p and B_p, are increased with the scale factors of 1.25 and 1.5, respectively. A worth mentioning result inferred from these three figures is that for smaller values of IoT nodes, 20-25 in Scenario 1, 20-35 in Scenario 2, and 20-40 in Scenario 3, the performance of IC2DQN is the best, and IC-MDQN is better than IC-FDQN. However, for larger values of IoT node numbers, the IC-FDQN outperforms the baselines. Indeed, in IC2DQN the

local server as a central agent tries to concurrently maximize the performance of all UAVs. With a fixed size of DNN, for a large number of IoT nodes, it is not feasible and performance degrades. However, in IC-MDQN and IC-FDQN, the agents behave independently and have more degrees of freedom in determining the local optimal policy. The same discussion for the percentage of AoI violations is true.

The effect of an increase in the number of IoT nodes on AoI violation is illustrated in Fig. 11(a) to Fig. 11(c). In this investigation, the available processing resources are assumed to be fixed when the number of IoT nodes increases. The same behavior as Fig. 11(a) to Fig. 11(c) but this time for the percentage of AoI violation can be seen also here. In *scenario 2* and *scenario 3*, the processing node's total resources, C_p and B_p, are increased with the scale factors of 1.25 and 1.5, respectively. As it is evident, just the proposed IC-FDQN method is able to achieve *zero* percentage of AoI violation. A result that was expected with respect to the results of Fig. 7, where it is shown that the only method that is able to converge to zero residual cost is IC-FDQN. For a significant range of IoT numbers, this superiority remains. For example in Fig. 11(a), where the IoT nodes number is 35, the percentage of AoI violation for the baseline methods is 20% and more whereas this value for the proposed IC-FDQN method is zero.

IX. CONCLUSION

We considered the problem of dynamic placement and scheduling of NFV-enabled SFCs in a smart agriculture application with the aim of minimizing total energy consumption throughout the network while there is a strict condition on the Age of Information (AoI). The problem is formulated as a Dec-CMDP. Then, adopting the symmetric structure of the network, a novel federated learning-based iterative method is proposed to solve the problem efficiently. The proposed method is distributed and energy efficient since the local agents only need to share the parameters of their locally trained model with each other. The privacy-supporting feature of FL significantly decreased the communication overhead which in our problem led to a significant reduction in the total energy consumption of the network. Regarding satisfaction of the constraint and instantaneous AoI values, the proposed method is able to meet the required constraint for a reasonable range of parameter settings and has the best performance in comparison to the baseline methods. In terms of freshness of information and for a realistic parameter setting, the AoI is minimized jointly. Simulation results demonstrated that the achieved value for the AoI is appropriate for most near real-time applications.

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. Autom. Sinica*, vol. 8, no. 4, pp. 718–752, Apr. 2021.

[2] P. K. R. 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.

[3] K. Gupta and N. Rakesh, "IoT-based solution for food adulteration," in *Proc. 1st Int. Conf. Smart Syst., Innov. Comput.* Singapore: Springer, Jan. 2018, pp. 9–18.

[4] S. Nirenjena, D. L. BalaSubramanian, and M. Monisha, "Advancement in monitoring the food supply chain management using IoT," *Int. J. Pure Appl. Math.*, vol. 119, no. 14, pp. 1193–1197, Jan. 2018.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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)*, Oct. 2022, pp. 270–275.

[10] 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.*, early access, Feb. 24, 2023, doi: 10.1109/TII.2023.3248616.

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

[12] 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)*, May 2022, pp. 189–194.

[13] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[14] Y. Sun, I. Kadota, R. Talak, and E. Modiano, "Age of information: A new metric for information freshness," *Synth. Lectures Commun. Netw.*, vol. 12, no. 2, pp. 1–224, Dec. 2019.

[15] 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.

[16] 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.

[17] 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," *Biosystems Eng.*, vol. 177, pp. 4–17, Jan. 2019.

[18] 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.

[19] 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.

[20] 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.

[21] *NFV Release 2015 Definition*, European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, 2015.

[22] 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, Oct. 2019.

[23] E. Altman, *Constrained Markov Decision Processes*. Boca Raton, FL, USA: Chapman & Hall/CRC, 1999.

[24] 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, Nov. 2002.

[25] 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.

[26] 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.

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

[28] H. 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 Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[29] 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.

[30] 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.

[31] Y. Liu, A. Halev, and X. Liu, "Policy learning with constraints in model-free reinforcement learning: A survey," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 4508–4515.

[32] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[33] J. Li, D. Fridovich-Keil, S. Sojoudi, and C. J. Tomlin, "Augmented Lagrangian method for instantaneously constrained reinforcement learning problems," in *Proc. 60th IEEE Conf. Decis. Control (CDC)*, Dec. 2021, pp. 2982–2989.

[34] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8103–8112.

[35] T. Zhang, J. Lei, Y. Liu, C. Feng, and A. Nallanathan, "Trajectory optimization for UAV emergency communication with limited user equipment energy: A safe-DQN approach," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1236–1247, Sep. 2021.

[36] Z. Zhang et al., "Energy-efficient secure video streaming in UAV-enabled wireless networks: A safe-DQN approach," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 4, pp. 1892–1905, Dec. 2021.

[37] 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.

[38] R. Bellman, "A Markovian decision process," *Indiana Univ. Math. J.*, vol. 6, no. 4, pp. 679–684, 1957.

[39] F. De Nijs, E. Walraven, M. De Weerdt, and M. Spaan, "Constrained multiagent Markov decision processes: A taxonomy of problems and algorithms," *J. Artif. Intell. Res.*, vol. 70, pp. 955–1001, Mar. 2021.

[40] Y. Yang and J. Wang, "An overview of multi-agent reinforcement learning from game theoretical perspective," 2020, *arXiv:2011.00583*.

[41] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman, "Solving transition independent decentralized Markov decision processes," *J. Artif. Intell. Res.*, vol. 22, pp. 423–455, Dec. 2004.

[42] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction* (Adaptive Computation and Machine Learning Series), 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[43] 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.

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

[45] M. Zinkevich and T. Balch, "Symmetry in Markov decision processes and its implications for single agent and multiagent learning," in *Proc. 18th Int. Conf. Mach. Learn.*, San Francisco, CA, USA, 2001, p. 632.

[46] S. Paternain, L. F. O. Chamon, M. Calvo-Fullana, and A. Ribeiro, "Constrained reinforcement learning has zero duality gap," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 1–11.

[47] 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.

[48] 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.

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

[50] 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, Oct. 2017, pp. 316–321.

[51] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[52] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-iid data," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–26.

[53] Y. Jee Cho, P. Sharma, G. Joshi, Z. Xu, S. Kale, and T. Zhang, "On the convergence of federated averaging with cyclic client participation," 2023, *arXiv:2302.03109*.

[54] S. Wan, J. Lu, P. Fan, Y. Shao, C. Peng, and K. B. Letaief, "Convergence analysis and system design for federated learning over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3622–3639, Dec. 2021.

[55] M. Sipper, "A serial complexity measure of neural networks," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 2, Mar./Apr. 1993, pp. 962–966.

[56] 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.

[57] *Getting Started With Gym*. Accessed: Sep. 13, 2023. [Online]. Available: https://www.gymlibrary.dev/

[58] K. K. Sharma and H. Krishna, "Asymptotic sampling distribution of inverse coefficient-of-variation and its applications," *IEEE Trans. Rel.*, vol. 43, no. 4, pp. 630–633, Dec. 1994.

[59] S. Aja-Fernandez and C. Alberola-Lopez, "On the estimation of the coefficient of variation for anisotropic diffusion speckle filtering," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2694–2701, Sep. 2006.

[60] K. A. Gomez and A. A. Gomez, *Statistical Procedures for Agricultural Research*, 2nd ed. Hoboken, NJ, USA: Wiley, 1984.

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 (IUST), Tehran, Iran, in 2010 and 2016, respectively. From 2010 to 2017, he was a Senior System Designer with the Afratab Research and Development Group, Tehran. In 2018, he joined the Department of Communication Technology, ICT Research Institute (ITRC), Tehran, as a Research Assistant Professor. Since September 2021, he has been a Post-Doctoral Fellow with the University of Ottawa, Ottawa, ON, Canada. His current research interests include 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 currently an Augmented Dynamic Networks Researcher with the Modeling and Optimization Group, Nokia Bell Labs. Her current research interests include automating networks 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 received the joint Ph.D. degree in mathematics and computer science from McGill University, Montreal, Canada. In 2011, he joined Bell Labs as a Post-Doctoral Researcher before becoming a member of Technical Staff with the Mathematics of Network Systems Department. He currently leads the Nokia Bell Labs' Artificial Intelligence Research Laboratory 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 toward systems that mimic the human capacity for analytical thinking.

MELIKE EROL-KANTARCI (Senior Member, IEEE) is currently 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 also the Founding Director of the Networked Systems and Communications Research (NETCORE) Laboratory. She is the co-editor of three books on smart grids, smart cities, and intelligent transportation. She has more than 200 peer-reviewed publications. She has delivered more than 70 keynotes, plenary talks, and tutorials around the globe. Her main research interests include AI-enabled wireless networks, 5G and 6G wireless communications, smart grid, and the Internet of Things. She is an ACM Senior Member and IEEE ComSoc Distinguished Lecturer. She has received numerous awards and recognitions, including the 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. She is an IEEE ComSoc Distinguished Lecturer.