Computation and Privacy Protection for Satellite-Ground Digital Twin Networks

Yongkang Gong, *Member, IEEE,* Haipeng Yao, *Senior Member, IEEE,* Xiaonan Liu, *Student Member, IEEE,* Mehdi Bennis, *Fellow, IEEE,* Arumugam Nallanathan, *Fellow, IEEE,* and Zhu Han, *Fellow, IEEE*

Abstract—**Satellite-ground integrated heterogeneous networks can relieve network congestion, release network resources and provide ubiquitous intelligence services for terrestrial users. Furthermore, digital twin technology can enable nearly-instant data mapping from the physical world to digital systems. The integration between satellite-ground integrated heterogeneous networks and digital twin alleviates the gap between data analyses and physical unities. However, the current challenges, such as the pricing policy, the stochastic task arrivals, the time-varying satellite locations, mutual channel interference, and resource scheduling mechanisms between the users and cloud servers, severely affect the improvement of quality of service. Hence, we establish a blockchain-aided Stackelberg game model for maximizing the pricing profits and network throughput in terms of minimizing privacy overhead, which is able to perform computation offloading, decrease channel interference, and improve privacy protection. Due to the long-term task queue in Stackelberg model, we propose a Lyapunov stability theory-based model-agnostic meta-learning aided multi-agent deep federated reinforcement learning framework to transfer the long-term task queue into the single time slot, and then optimize the central processing unit frequency, channel selection, task-offloading decision, block size, and cloud server price, which facilitate the integration of communication, computation, and block resources. Subsequently, several performance analyses show that the proposed learning framework can strengthen the privacy protection, approach the optimal time average function, and fulfill the long-term average queue size via lower computational complexity. Finally, our simulation results indicate that the proposed learning framework is superior to the existing baseline methods in terms of network throughput, channel interference, cloud server profits, and privacy overhead.**

Index Terms—**Satellite-ground integrated digital twin networks, model-agnostic meta-learning multi-agent deep federated reinforcement learning, blockchain-aided transaction verification, resource management.**

I. Introduction

Although terrestrial networks [1] can support high data rate and large-scale terminal devices access, it is hard to provide seamless, global, and uniform coverage for low user-density regions and to fulfill the rapid proliferation of mobile applications in the sixth-generation wireless communication systems [2]. Fortunately, satellite-ground integrated networks [3] can fill the coverage-holes to support global coverage, especially in remote suburbs, oceans, and deserts. Specifically, satellite networks (e.g., Starlink) consist of multiple low earth orbits (LEO), medium earth orbits, and geostationary earth orbits [4], which provide global coverage, moderate relay transmission, and task processing functionalities for terrestrial users.

There have been some related contributions about edge task offloading. Specifically, Luo *et al.* [5] proposed edge server network design algorithms to balance the construction cost and the network density for edge networks. Next, Alnoman *et al.* [6] explored a sharing and disjoint cloud-edge system to minimize the response time via dynamic programming and exhaustive searching methods. Fan *et al.* [7] established a game-theory based multi-type computation offloading mechanism to balance the task computing delay among multiple base stations.

However, when terrestrial users offload tasks to macro base station (MBS) servers, it may cause severe channel interference among different edge networks, bring huge transmission energy consumption and reduce the ability of processing number of tasks. Moreover, the above works do not consider the corresponding cloud servers' pricing mechanisms and the processed number of task bits. Fortunately, digital twin (DT) [8] is regarded as a novel technique, which can enable instant wireless connectivity as well as data mapping services, and shorten the gap among physical utilities and digital systems [9]. Furthermore, Lu *et al.* [10] formulated the edge association problem including DT placement and DT migration, and then employed the deep reinforcement learning (DRL) and transfer learning mechanisms to improve the convergent rate. Huynh *et al.* [11] established a multi-access edge computing-based ultra-reliable and low latency communications architecture to optimize the offloading portions, bandwidth and server computation capability, which can improve latency and reliability in metaverse applications.

However, the privacy protection for task offloading and lack of mutual trust among terrestrial users impede resource sharing and cooperation. To solve these challenges, blockchain technologies can be widely deployed to achieve transaction verification and privacy protection functionalities among substantial users. Qiu *et al.* [12] improved the proof of work via proposing a blockchain-assisted collective Q-learning method.

Y. Gong is with the School of Computer Science and Technology, SDUT, QingDao, 266237, China (e-mail: gokawa@sdu.edu.cn).
H. Yao is with the School of Information and Communication Engineering, BUPT, Beijing, 100876, China (e-mail: yaohaipeng@bupt.edu.cn).
X. Liu is with the University of Edinburgh, EDIN, Edinburgh, EH8 9YL, the United Kingdom (e-mail: liuxiaonan19931107@gmail.com).
A. Nallanathan is with the Communication Systems Research group in Queen Mary University of London, QMUL, London, E1 4NS, the United Kingdom (e-mail: a.nallanathan@qmul.ac.uk).
M. Bennis is with the Center for Wireless Communications, University of Oulu, 90014 Oulu, Finland (e-mail: mehdi.bennis@oulu.fi).
Z. Han is with the School of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA (e-mail: zhan2@uh.edu).

Furthermore, Cao *et al.* [13] conceived a blockchain-aided software-defined energy network and designed a distributed energy smart contract to guarantee transactions reliably and accurately.

Despite the advantage of blockchain technique [14] for improving privacy protection, the learning efficiency needs to be further explored for dynamic network environment. Nguyen *et al.* [15] utilized DRL to minimize the latency and mining cost of machine learning model owner. Furthermore, Du *et al.* [16] utilized an asynchronous advantage actor-critic algorithm to obtain the optimal resource pricing and allocation, and used the prospect theory to balance risks and rewards. Ma *et al.* [17] established an autonomous control platform to optimize network resources, adjust power services, and maximize the profits for consumers and operators. However, these mentioned methods cause large transmission overhead as well as low learning efficacy and may leak user privacy. Hence, a model-agnostic meta-learning (MAML) [18] framework is introduced to quickly adapt to new tasks from small samples, which can greatly improve the learning efficiency and accelerate the convergent rate. Furthermore, multi-agent deep federated reinforcement learning (MADFRL) is deployed to execute the task scheduling and strengthen the privacy protection for dynamic network environment.

Inspired by above challenges, we conceive satellite-ground integrated digital twin networks (SGIN-DT) scenario for computation offloading, alleviating channel interference, and improving privacy protection among users under dynamic network environment, and then maximize the cloud servers' profits for time-varying DT computation capability. The main contributions are presented as follows.

- We envision a blockchain-aided two-stage Stackelberg game model to maximize the processed number of task bits and cloud servers profits. Integrated with in-orbit intelligent computation, it helps the network adapt to the stochastic task arrivals, the time-varying LEO locations, the cloud server price, and the DT computation frequency. Furthermore, it decouples the variable coupling for the long-term task queue and the short-term task offloading.
- We propose a Lyapunov stability theory-based MAML-MADFRL framework to optimize DT computation frequency, allocate wireless channel, execute task offloading, choose block size, and obtain the optimal price for cloud servers. Specifically, the Lyapunov-based policy is convoked to decouple the long-term task queues. Next, the proposed MAML-MADFRL framework is utilized to process the computation offloading, channel interference, and privacy protection. Moreover, the MAML-MADFRL framework can obtain the optimal price for cloud servers in the second-stage Stackelberg game.
- Several theoretical analyses show that the proposed Lyapunov-based MAML-MADFRL framework can validate the transaction process, approach the optimal performance, and satisfy the long-term task queue constraint via lower computation complexity. Furthermore, extensive simulation results indicate that the proposed learning framework is superior to the traditional baselines, such as multi-agent random task offloading (MARTO), multi-agent mean central processing unit (CPU) cycle (MAMCC), multi-agent greedy channel selection (MAGCS) and multi-agent proximal policy optimization (MAPPO).

The structure of the paper is organized as follows. We list some related works in Section II. Moreover, Section III establishes the blockchain-aided system model. Next, we introduce the corresponding Lyapunov stability theory-based learning framework in Section IV. Furthermore, the transaction verification, the computational complexity of the algorithm, the convergence rate, and the task queue constraints are demonstrated in Section V. Subsequently, massive simulation results are presented in Section VI. Finally, we conclude this paper in Section VII.

II. Related Works

Blockchain-aided SGIN-DT is considered as the prospective network architecture to achieve flexible deployment, global coverage, and cognitive capability. Specifically, Cao *et al.* [19] conceived a transmission control policy for ground-air-space and ground-to-space links and maximized the overall network throughput. Guo *et al.* [20] provided a detailed survey about network security on space-air-ground-sea network. Fan *et al.* [21] processed the network selection via evolutionary game and utilized the deep deterministic policy gradient to handle high-dimensional action spaces. However, the aforementioned works do not consider the real-time task processing between physical unities and digital systems.

Recently, DT can be utilized to accelerate the wireless network evolution and map the task data to digital systems. Lu *et al.* [22] introduced DT to wireless networks and proposed a learning framework to balance the learning efficiency and time cost. Bellavista *et al.* [23] processed the application-enabled DT equipment and applied software-defined networking to explore communication mechanisms. Lei *et al.* [24] established a DT-based thermal power plant and explored the web-based architecture and control algorithm.

However, the lack of mutual trust as well as privacy protection, and low training efficiency reduce the quality of service. Hence, some research works focused on blockchain and federated learning (FL) [25]. Specifically, Qu *et al.* [26] developed a decentralized cognitive computing paradigm and utilized the blockchain-aided federated learning technique to solve data island and incentive mechanism. Cui *et al.* [27] proposed a blockchain-aided compressed federated learning framework to maximize the final model accuracy and minimize the training loss. Wang *et al.* [28] designed an "In-Edge AI" framework to reduce system communication overhead and integrated DRL with federated learning to optimize communication, computation as well as caching resources. Nevertheless, aforementioned works cannot guarantee fast model adaptability from small batches of samples.

In contrast with aforementioned works, we propose a Lyapunov stability theory-based MAML-MADFRL learning framework to decouple the long-term task queues and accelerate the learning rate for multiple task scenarios in SGIN-

DT, which can adapt to dynamic network environment and maximize the network throughput as well as cloud server profits in two-stage Stackelberg game process.

III. System Model

A. SGIN-DT Blockchain Scenario

As shown in Fig. 1, we introduce the SGIN-DT blockchain scenario, which is deployed to achieve computation offloading and privacy protection for multiple ground devices. The network scenario consists of two main layers, i.e., satellite networks and terrestrial networks. Specifically, the terrestrial network is composed of multiple MBSs, the sets of which are represented as $\mathbb{N} = \{1, 2, ..., n, ..., N\}$. Moreover, each n overlays $\mathbb{M} = \{1, 2, ..., m, ..., M\}$ ground devices and all M ground devices execute instant wireless access and reliable data mapping from physical entities to digital space via the DT technology.

Subsequently, the satellite networks consist of massive LEO satellites, whose sets are denoted as $\mathbb{O} = \{1, 2, ..., o, ..., O\}$. Additionally, all LEO satellites can provide global communication coverage and seamless connectivity for ground devices. Meanwhile, tasks are offloaded to LEO to relieve computation pressure while protecting data privacy by federated aggregation and issuing mechanisms. In this case, we consider that each ground device corresponds to one DT. Specifically, as multiple DTs lack mutual trust and cannot share local data, we propose a blockchain-aided federated aggregation policy to execute model parameters aggregation and issue network parameters, which can not only achieve computation offloading and improve training efficiency, but also further protect data privacy.

Next, when multiple DTs offload their tasks to LEO satellites, cloud providers can set own price to earn more economic profits, and then massive followers can determine own service demand after obtaining the price strategy of cloud providers. Hence, the resource orchestration and pricing strategy between cloud providers and multiple DTs can be regarded as a Stackelberg game process, and we need to find the optimal policy to fulfill the service requirements for both cloud providers and DTs.

Meanwhile, blockchain is a novel distributed ledger technology, which can be utilized to strengthen mutual trust among massive ground devices. Specifically, when tasks are offloaded to LEO servers, the corresponding LEO server can help verify model parameters and issue them to corresponding ground DTs, which guarantee secure network transaction.

B. DT Computation Model

Furthermore, we can divide the Stackelberg game process into two stages, i.e., the follower stage and the provider stage. First, massive DTs tend to maximize the number of processed bits in terms of minimum blockchain verification overhead, and detailed computation offloading and blockchain verification processes are shown as follows.

For each time slot t, the DT m in the n^{th} MBS receives the application task of size $A^t_{n,m}$ (measured in bits) and assuming

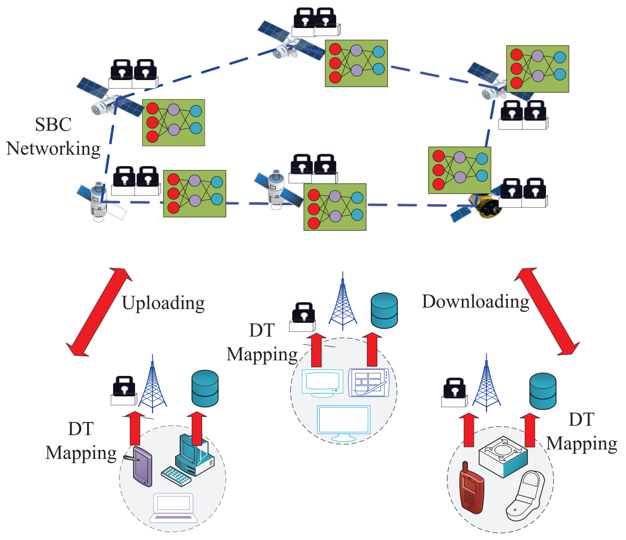

Fig. 1: The proposed SGIN-DT blockchain scenario.

that the second moment is limited, i.e., $\mathbb{E}\left([A^t_{n,m}]^2\right) = \eta_{n,m} < \infty$. Moreover, $\eta_{n,m}$ can be obtained via collecting previous network statistics. Next, when each DT processes the task locally, the processed number of task bits is calculated as

$$D^{t1}_{n,m} = \frac{f^t_{n,m}}{w}\tau, \tag{1}$$

where $f^t_{n,m}$ is the CPU cycle frequency of each DT, w is the allocated number of CPU cycles while processing one bit task, and τ is time slot duration.

C. Computation Offloading Model

When DTs offload their tasks to LEO satellites, stochastic task arrivals, dynamic LEO locations, and mutual channel interference among different ground edge networks cause severe impact on computation offloading and privacy protection. Moreover, the loss of the communication link between DT m and LEO o for the n^{th} MBS is denoted as

$$L^{t,o}_{n,m} = 20\log_2\left(\frac{4\pi f_c\sqrt{x^2_{n,m,o} + y^2_{n,m,o}}}{c}\right) + p^{LoS}_{n,m,o}\varepsilon^{LoS}_{n,m,o} + \left(1 - p^{LoS}_{n,m,o}\right)\varepsilon^{NLoS}_{n,m,o}, \tag{2}$$

where f_c is the carrier frequency and c is the speed of light, $x_{n,m,o}$ is the horizontal distance between DT m and LEO o, $y_{n,m,o}$ is the flight altitude for each LEO satellite o. Moreover, the unit of $L^{t,o}_{n,m}$ is dB. $\varepsilon^{LoS}_{n,m,o}$ and $\varepsilon^{NLoS}_{n,m,o}$ are related additional path loss imposed on free space propagation from line-of-sight and non-line-of-sight, respectively. Furthermore, the corresponding line-of-sight propagation probability is denoted as

$$p^{LoS}_{n,m,o} = \frac{1}{1 + b1\exp\left\{-b2\left[\arctan\left({}^{y_{n,m,o}}/_{x_{n,m,o}}\right)\right] - b1\right\}}, \tag{3}$$

where $b1$ and $b2$ are corresponding constants [3], which are obtained via interacting with dynamic environment. Subsequently, when multiple DTs offload their tasks to LEO satellites, it can cause massive mutual interference among different edge networks. We assume that multiple DTs transmit their tasks via orthogonal frequency division multiple access, which causes interference among multiple edge networks. Hence, for the m^{th} DT in the n^{th} MBS, the channel interference is represented as

$$I^{o}_{n,m,r} = \sum_{q=1,q\neq n}^{N} \sum_{m'=1}^{M} \beta_{q,m',r} P_{q,m',r} \left| 10^{\frac{-L^{t,o}_{q,m'}}{10}} \right|^2, \quad (4)$$

where $\beta_{q,m',r} = 1$ represents that the channel r is assigned to DT m' in the q^{th} MBS. Otherwise, $\beta_{q,m',r} = 0$. Moreover, $P_{q,m',r}$ is the transmission power. Hence, the data transfer rate in the computation offloading mode is denoted as

$$R^{t,o}_{n,m,r} = B_{n,m,r} \log_2 \left(1 + \frac{a^{o}_{n,m,r} P_{n,m,r} |10^{\frac{-L^{t,o}_{n,m}}{10}}|^2}{\sigma^2 + I^{o}_{n,m,r}} \right), \quad (5)$$

where $B_{n,m,r}$ is the allocated channel bandwidth for DT m. $a^{o}_{n,m,r} = 1$ indicates that DT m offloads the task to the LEO o via the channel r and σ^2 is channel noise power. Consequently, the processed number of bits is represented as

$$D^{t2,o}_{n,m} = R^{t,o}_{n,m,r} \tau. \quad (6)$$

D. Blockchain Verification Model

Since multiple DTs cannot share task data with each other, we utilize the blockchain technology [12] to strengthen data privacy and prevent data tampering. Hence, each block records related model parameters and these information is verified via corresponding DTs. Subsequently, the privacy protection overhead is divided into parameters aggregation, transmission, and verification parts. Specifically, the parameters aggregation overhead is denoted as

$$C1 = \frac{|W_m|}{f_{MBS}}, \quad (7)$$

where $|W_m|$ and f_{MBS} are corresponding training model parameters and CPU cycle frequency from the MBS. Next, the parameters transmission overhead is denoted as

$$C2 = (\delta \log_2 N) \frac{|W_m|}{r_{up}}, \quad (8)$$

where δ is the model transmission factor and r_{up} is uplink transmission rate from the MBS to the LEO. Finally, the parameters verification overhead is represented as

$$C3 = (\delta \log_2 MN) \frac{S_B}{r_{down}} + \max_{\{m\}} \left\{ \frac{S_B}{f^{t}_{n,m}} \right\}, \quad (9)$$

where S_B is the size of blockchain and r_{down} is data downloading rate from the LEO to the MBS. Finally, the total blockchain privacy protection overhead is calculated as

$$C_{SBC} = C1 + C2 + C3. \quad (10)$$

Remark 1: Based on the reference [12], we establish the blockchain verification model and the privacy protection overhead can be divided into three parts, i.e., parameters aggregation, parameters transmission and parameters verification parts. Specifically, $|W_m|$ is the corresponding training model parameters for each DT m. Hence, the parameters aggregation time can be calculated as equation (7). Next, the parameters transmission overhead is related with the number of MBS and transmission efficiency. After model parameters aggregation, it can be broadcasted to other MBSs. Since the parameters need to be transmitted to remote server in the broadcast process, the transmission overhead is also related with $\log_2 N$, i.e., other MBSs also need to transmit the identification information. Meanwhile, we introduce the model transmission factor δ to denote the transmission efficiency. Then, each MBS can collect the transaction information and assemble them to the block S_B. Furthermore, it can be broadcasted to other DTs and the time overhead is related with $\log_2 MN$. Finally, it needs to be validated via the DT m.

E. Problem Formulation

In this section, we formulate the two-stage Stackelberg game model between the cloud servers and the terrestrial users. As terrestrial users can be regarded as DTs projected to the MBS, the cloud providers need to set the price to earn more profits after collecting the CPU cycle frequency $f^{t}_{n,m}$. Hence, the corresponding cloud servers profits are represented as

$$P1: \max_{\lambda_{n,m}} \sum_{n=1}^{N} \sum_{m=1}^{M} (\lambda_{n,m} f^{t}_{n,m} - c f^{t}_{n,m}) \quad (11)$$

$$s.t. \quad \lambda_{n,m} \geq 0, \quad (12)$$

where $\lambda_{n,m}$ indicates the cloud servers' price and c is one unit electronic consumption. Next, for massive DTs m, the network throughput is calculated after knowing the pricing policy from cloud servers, which is represented as

$$F1 = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\alpha^{t}_{n,m} D^{t1}_{n,m} + (1 - \alpha^{t}_{n,m}) D^{t2,o}_{n,m}], \quad (13)$$

where $\alpha^{t}_{n,m}$ denotes the offloading decision for each DT m in the n^{th} MBS and the offloading is performed to any LEO in general. As $\alpha^{t}_{n,m} \in \{0, 1\}$, the offloading process is a full offloading for any DT and LEO, i.e., each task cannot be partitioned into multiple LEOs for execution. Meanwhile, $D^{t2,o}_{n,m}$ indicates that each DT m in the n^{th} MBS can choose any LEO for achieving computation offloading model instead of depending on a specific LEO, which corresponds to the offloading decision $\alpha^{t}_{n,m}$. In other words, each LEO can cover multiple DTs, i.e., each DT m can offload the complete task to any corresponding LEO and the set of LEO is $\mathbb{O} = \{1, 2, ..., o, ..., O\}$. Next, the privacy overhead and price loss are denoted as

$$F2 = \mathbb{E}[C_{SBC} + \lambda_{n,m} f^{t}_{n,m}]. \quad (14)$$

Furthermore, the total service demands for users are represented as

$$P2: \max_{\{f^{t}_{n,m}, \alpha^{t}_{n,m}, a^{o}_{n,m,r}, S_B\}} \{F1 - F2\} \quad (15)$$

$$\text{s.t.} \quad \alpha_{n,m}^t \frac{f_{n,m}^t}{w}\tau + (1-\alpha_{n,m}^t)(R_{n,m,r}^{t,o}\tau) \le Q_{n,m}^t, \tag{16}$$

$$I_{n,m,r}^o \le I_{\max}, \tag{17}$$

$$\alpha_{n,m}^t \in \{0,1\}, \tag{18}$$

$$\lim_{T\to\propto} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[Q_{n,m}^t] < \propto, \tag{19}$$

$$S_{\min} \le S_B \le S_{\max}, \tag{20}$$

where (16) represents that the processed number of tasks cannot exceed the task queue $Q_{\mathrm{n},m}^t$, (17) means that the channel interference should be less than maximum $I_{\max}$. More importantly, for the variable $\beta_{q,m',r}$, it can be obtained from other terrestrial users $m^{'}$ and input to P2. Thus, we consider it as the hyper-parameter in P2. $\alpha_{\mathrm{n,m}}^t$ is corresponding offloading decision, and (19) indicates that the long-term task queue for each DT is limited. Finally, (20) denotes the block size. Meanwhile, we list the TABLE I for massive acronyms to improve the readability.

TABLE I: Summary of Acronyms

Description	Acronyms
satellite-ground integrated heterogeneous networks	SGIN
digital twin	DT
model agnostic meta learning	MAML
multi-agent deep federated reinforcement learning	MADFRL
macro base station	MBS
deep reinforcement learning	DRL
low earth orbit	LEO
multi-agent random task offloading	MARTO
multi-agent mean CPU cycle	MAMCC
multi-agent greedy channel selection	MAGCS
federated learning	FL

IV. Algorithm Design in SAG-DT Integrated Blockchain Network

A. Lyapunov-based Problem Transformation

In terms of P1 and P2, when the network provider sets the service price $\lambda_{n,m}$, multiple DTs make optimal task decision-makings in terms of cloud server pricing, which can be regarded as a two-stage Stackelberg game process. Hence, we need to explore the optimal DTs service demands according to servers pricing. However, P2 indicates the variables coupling between long-term task queue and short-term computation offloading as well as privacy protection, which make it hard to decouple corresponding optimization variables. Subsequently, we introduce Lyapunov stability theory to further transform multiple time slots problem into a single time slot subproblem. Meanwhile, we denote the virtual task queue $\{\vec{Q}(t)\}_{n,m}^t$ to fulfill the long-term task queue constraint. Noting that the problem P2 is solved individually via each user in the MBS, which is different from traditional Lyapunov function and drift function. Hence, the corresponding Lyapunov function and drift function are denoted as

$$L(\vec{Q}(t)) = 0.5(Q_{n,m}^t)^2, \tag{21}$$

$$\Delta L(\vec{Q}(t)) = \mathbb{E}\left\{L(\vec{Q}(t+1)) - L(\vec{Q}(t))|\vec{Q}(t)\right\}. \tag{22}$$

In terms of (21) and (22), we can decompose the long-term constraint (19) and further derive the minimum values of drift-plus-penalty algorithm. Moreover, we denote the virtual queue backup $Q_{n,m}^t$ as

$$Q_{n,m}^{t+1} = \max\left\{Q_{n,m}^t - D_{n,m}^{t1/t2} + A_{n,m}^t, 0\right\}, \tag{23}$$

where $D_{n,m}^{t1/t2} = \alpha_{n,m}^t D_{n,m}^{t1} + (1-\alpha_{n,m}^t) D_{\mathrm{n,m}}^{\mathrm{t2,o}}$.

We further simplify (23) as

$$\begin{aligned}(Q_{n,m}^{t+1})^2 \le (Q_{n,m}^t)^2 + 2Q_{n,m}^t(A_{n,m}^t - D_{n,m}^{t1/t2}) \\ + (A_{n,m}^t - D_{n,m}^{t1/t2})^2.\end{aligned} \tag{24}$$

Moreover, as the problem P2 is individually solved via each user in the MBS, based on the Lyapunov function (21) and drift function (22), we transform the formula (24) via dividing 0.5 on either side of the inequality, which is denoted as

$$\begin{aligned}0.5\left(Q_{n,m}^{t+1}\right)^2 - 0.5\left(Q_{n,m}^t\right)^2 \\ \le Q_{n,m}^t(A_{n,m}^t - D_{n,m}^{t1/t2}) + 0.5(A_{n,m}^t - D_{n,m}^{t1/t2})^2.\end{aligned} \tag{25}$$

Next, we derive the Lyapunov drift as

$$\begin{aligned}&\triangle\ L(\vec{Q}(t)) = \mathbb{E}\left\{L(\vec{Q}(t+1)) - L(\vec{Q}(t))|\vec{Q}(t)\right\} \\ &\le \mathbb{E}\left\{0.5\left(A_{n,m}^t - D_{n,m}^{t1/t2}\right)^2 + Q_{n,m}^t\left(A_{n,m}^t - D_{n,m}^{t1/t2}\right)\right\} \\ &\le \mathbb{E}\left\{X + Q_{n,m}^t\left(A_{n,m}^t - D_{n,m}^{t1/t2}\right)\right\},\end{aligned} \tag{26}$$

where

$$\mathbb{E}\left\{0.5\left(A_{n,m}^t - D_{n,m}^{t1/t2}\right)^2\right\} \le \mathbb{E}\left\{0.5\left(A_{n,m}^t\right)^2 + 0.5\left(D_{n,m}^{t1/t2}\right)^2\right\} \tag{27}$$

$$\le 0.5\eta_{n,m} + \left(D_{n,m}^{t1/t2/\max}\right)^2 = X,$$

where $D_{n,m}^{t1/t2/\max}$ is the maximum of $D_{n,m}^{t1/t2}$. Moreover, it denotes the maximum local DT execution and offloading execution.

Furthermore, the drift-plus-penalty equation is calculated as

$$\Theta\left(\vec{Q}(t)\right) = \Delta L\left(\vec{Q}(t)\right) - V\mathbb{E}\{F\}, \tag{28}$$

where V is the corresponding Lyapunov control parameter and

$$\begin{aligned}F = \alpha_{n,m}^t D_{n,m}^{t1} + (1-\alpha_{n,m}^t) D_{n,m}^{t2,o} \\ - C_{SBC} - \lambda_{n,m} f_{n,m}^t.\end{aligned} \tag{29}$$

Hence, the original optimization P2 for all DTs is transformed

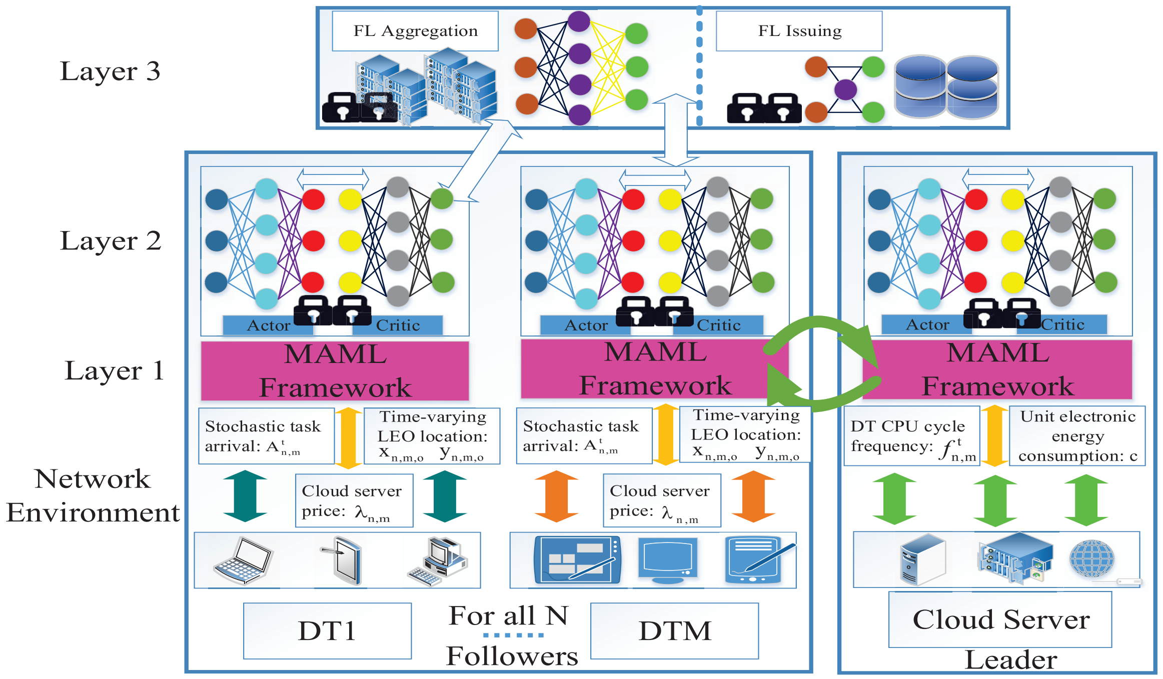
Layer 3
Layer 2
Layer 1
Network Environment
FL Aggregation
FL Issuing
Actor
Critic
MAML Framework
Stochastic task arrival: $A^t_{n,m}$
Time-varying LEO location: $x_{n,m,o}$ $y_{n,m,o}$
Cloud server price: $\lambda_{n,m}$
DT CPU cycle frequency: $f^t_{n,m}$
Unit electronic energy consumption: c
DT1
For all N
DTM
Followers
Cloud Server
Leader

Fig. 2: The proposed three-layer MAML-MADFRL framework.

into

$$P2^{'}: \max_{\{f^t_{n,m}, \alpha^t_{n,m}, a^o_{n,m,r}, S_B\}} Q^t_{n,m} D^{t1/t2}_{n,m} + VF \quad (30)$$
$$s.t. \quad (16), (17), (18), (20)$$

Specifically, we transform the long-term multi-slot optimization problem into a single-slot optimization problem, which utilizes the proposed Lyapunov drift-plus-penalty algorithm to maximize the number of processed bits in terms of minimum privacy overhead and pricing strategy from cloud servers. Subsequently, the specific Lyapunov-based problem transformation is shown in Algorithm 1.

Remark 2: For the Lyapunov-based problem transformation, we further decompose the long-term multi-slot problem into a single-slot optimization problem for all followers. After the Lyapunov-based problem transformation, it is not necessary to know future system state information and probability distribution of random events, which is regarded as a model-free learning. Moreover, the proposed Lyapunov-based transformation mechanism and MAML-MADFRL algorithm can satisfy the constraint in (19), which are verified in Section V.

B. MAML-MADFRL Algorithm Design

As shown in Fig. 2, we envision a three-layer MAML-MADFRL framework, which consists of a MAML fast adaptation module and a MADFRL optimization module. Moreover, the framework includes multiple followers (i.e., DTs) and one cloud server (leader). Specifically, multiple DTs can adapt to stochastic task arrival ($A^t_{n,m}$), time-varying LEO location ($x_{n,m,o}$, $y_{n,m,o}$), and cloud server price ($\lambda_{n,m}$), which can further obtain the optimal CPU cycle frequency $f^t_{n,m}$, task

```
Algorithm 1 Lyapunov drift-plus-penalty algorithm
Input:
    The original multi-slot optimization problem P2 for all followers.
Output:
    The single-slot optimization problem $P2^{'}$.
1: Define the virtual task queue $Q^t_{n,m}$;
2: Compute the Lyapunov function $L\left(\vec{Q}(t)\right)$ and Lyapunov drift function $\Delta L\left(\vec{Q}(t)\right)$.
3: Derive the Lyapunov drift from (23) to (27);
4: Obtain the drift-plus-penalty equation in (28);
5: Output the single-slot optimization problem $P2^{'}$;
```

offloading decision-making $\alpha^t_{n,m}$, channel selection $a^o_{n,m,r}$, and block size S_B. Meanwhile, the leader can set the cloud server price ($\lambda_{n,m}$) in terms of $f^t_{n,m}$ and unit electronic energy consumption c, which in turn helps each DT make corresponding execution actions.

Subsequently, we utilize the proposed FL aggregation and FL issuing policies to further accelerate model convergent rate and protect user privacy. More importantly, the FL aggregation mechanism adjusts corresponding DT actor network parameters in terms of task weight and time-varying LEO locations. After finishing FL aggregation, the corresponding model parameters are issued to each DT, which can further achieve privacy protection and security verification. Next, we introduce related MAML fast adaptation algorithms and MADFRL optimization mechanisms.

1) *MAML Fast Adaptation Algorithm*

The goal of MAML is to rapidly learn new tasks from small batches of samples, which are more efficient than learning from scratch. Moreover, MAML can collect historical experience from past tasks to rapidly adapt to new tasks. Assuming that these old tasks for meta-training and new tasks for meta-testing are subject to the same basic distribution $p(\Gamma)$, there are some common characteristics among different tasks. For conventional DRL scenarios, the aim is to minimize the loss function L_Γ for specific tasks Γ. In our SAG-DT scenario, the specific task Γ refers to the number of size $A_{n,m}^t$. Furthermore, the proposed MAML is a fast adaptation algorithm and it can accelerate the convergence rate of MADFRL. Hence, the form of the specific loss function is described as $L_\Gamma = \sum_\Gamma -Q\left(S_{n,m}, A_{n,m}\right)$, where $Q\left(S_{n,m}, A_{n,m}\right)$ is the Q-value function from the critic neural network. The form of function $Q\left(S_{n,m}, A_{n,m}\right)$ is represented as $Q\left(S_{n,m}, A_{n,m} \left| S_t = S_{n,m}, A_t = A_{n,m}\right.\right) = \mathbb{E}\left(R_{n,m}^{t+1} + \gamma R_{n,m}^{t+2} + \gamma^2 R_{n,m}^{t+3} + ... \left| S_t = S_{n,m}, A_t = A_{n,m}\right.\right)$. Specifically, when we input the state $S_{n,m}$ and action $A_{n,m}$ to the critic neural network, the neural network generates the Q-value function [29]. As a larger Q-value means more reward and less loss, we take a negative sign for the Q-value function to update the actor neural network. MAML can learn corresponding network weight parameters $w^{'} = u_\varphi\left(D_\Gamma^{tr}, w\right)$, which can utilize small batch of samples to rapidly adapt to new tasks Γ. Hence, MAML-RL problem is represented as

$$\min_{w} \quad \mathbb{E}_{\Gamma \sim p(\Gamma)}\left[L\left(D_\Gamma^{test}, w^{'}\right)\right] \tag{31}$$

$$s.t. \quad w^{'} = \mu_\varphi\left(D_\Gamma^{tr}, w\right), \tag{32}$$

where D_Γ^{tr} and D_Γ^{test} represent training task samples and testing task samples from $p\left(\Gamma\right)$ and $L\left(D_\Gamma^{test}, w^{'}\right)$ means the testing loss function from new network weight parameters $w^{'}$ in terms of testing task samples. Next, MAML process is divided into two parts, i.e., the inner loop and outer loop. For the inner loop, we sample new training tasks to update network weight parameters $w^{'}$, which are utilized to test model in the outer loop. Subsequently, we regard testing task samples error as loss function to retrain network model in the outer loop. We note that the model is an initial model for the inner loop and it only generates testing error for training tasks in the inner loop. Finally, MAML is responsible for updating initial model. The detailed MAML algorithm is shown in Algorithm 2.

Remark 3: MAML is based on gradient descent, and $w^{'} = u_\varphi\left(D_\Gamma^{tr}, w\right)$ is updated via several gradient descent steps to obtain better network performance gains for new testing tasks. The goal of MAML is to train and obtain model initial parameters, which maximizes the new tasks performance via several gradient update steps from small batches of samples. Although these update principles are fixed, a set of well-optimized neural network parameters followed with several gradient update steps are utilized to generalize new testing tasks.

Algorithm 2 MAML for MADFRL

```
Input:
    Followers: stochastic tasks arrivals $p(\Gamma)$ for followers, time-varying LEO location $x_{n,m,o}$ and $y_{n,m,o}$, cloud server price $\lambda_{n,m}$;
    Leader: DT CPU cycle frequency $f_{n,m}^t$ and unit electronic energy consumption $c$;
    Learning rate: $\alpha$ and $\beta$;
Output:
    Neural network parameters $w^{'}$;
1: Randomly initialize neural network parameters $w$;
2: while Not finished do
3:   Extract small batches of states $\Gamma$ from all followers and the leader;
4:   for all $\Gamma$ do
5:     Extract $K$ MDP trajectories $\Omega = \left\{S_{n,m}^{'} | S_{n,m}, A_{n,m}, R_{n,m}\right\}$ utilizing $w$ in $\Gamma$;
6:     Compute $\nabla_w L_\Gamma(w)$ utilizing $\Omega$ and $L_\Gamma(w)$;
7:     Update neural network parameters via gradient descent: $w^{'} = w - \alpha \nabla_w L_\Gamma\left(w\right)$;
8:     Continue to extract MDP trajectories $\Omega^{'} = \left\{S_{n,m}^{'} | S_{n,m}, A_{n,m}, R_{n,m}\right\}$ using $w^{'}$;
9:   end for
10:  Update $w = w - \beta \nabla_w \sum_\Gamma L_\Gamma\left(w^{'}\right)$ utilizing $\Omega^{'}$;
11: end while
```

2) *MADFRL Optimization Mechanism*

For the MAML algorithm, it helps network model obtain the optimal policy of new tasks from small samples via several gradient descent steps. Subsequently, we explore the MADFRL algorithm principle in Layer 2 and Layer 3. In Layer 2, we employ actor-critic network model to execute task offloading as well as resource allocation and adapt to dynamic network environment. Specifically, for multiple followers from 1 to M, we utilize multi-agent actor-critic network structure to process corresponding stochastic task arrival $A_{n,m}^t$, time-varying LEO location $x_{n,m,o}$ as well as $y_{n,m,o}$, and cloud server price $\lambda_{n,m}$, which cannot cause any information exchange among multiple followers to protect followers' privacy. Moreover, the proposed multi-agent actor-critic framework consists of a actor neural network and a critic neural network. The specific state, action and reward function for followers are defined as follows.

State Space: At the beginning of time slot t, each follower m in the n_{th} MBS receives the local state $S_{n,m} = \left\{A_{n,m}^t, x_{n,m,o}, y_{n,m,o}, \lambda_{n,m}\right\}$, and $S_{n,m}$ is not allowed to interact among multiple followers because of privacy protection. Meanwhile, the network state is constant in a single time slot t but varies across different time slots.

Action Space: For the multi-agent actor-critic framework, each follower makes following four actions via actor neural network, i.e., CPU cycle frequency $f_{n,m}^t$, task-offloading decision $\alpha_{n,m}^t$, channel selection $a_{n,m,r}^o$, and block size S_B. Hence, the action space for each follower is denoted as $A_{n,m} = \left(f_{n,m}^t, \alpha_{n,m}^t, a_{n,m,r}^o, S_B\right)$, which is executed via actor neural network.

Reward Function: As we intend to maximize the processed task numbers while minimizing the privacy protection overhead and frequency consumption, each follower needs to maximize its own reward function. Subsequently, the instant reward is represented as

$$R_{n,m} = Q_{n,m}^{t} D_{n,m}^{t1/t2} + VF. \tag{33}$$

Hence, in the training and testing phases, each follower can maximize the reward function from small batches of samples.

MDP Transition Process: For the blockchain-aided SGIN-DT model, it is hard to find a fixed transition policy to cover network states. Consequently, we can use $\Omega = \left\{ S_{n,m}^{'} | S_{n,m}, A_{n,m}, R_{n,m} \right\}$ to further represent the state transition between followers and network environment.

Similar to followers, the state space, action space and reward function for the leader are be represented as follows.

The leader needs to adjust its own price policy $\lambda_{n,m}$ according to DT CPU cycle frequency $f_{n,m}^{t}$ and unit electronic energy consumption c. Hence, the corresponding state space for the leader is $S_L = \{f_{n,m}^{t}, c\}$. Next, after obtaining the state space, each actor neural network generates network serving price $A_L = \{\lambda_{n,m}\}$. Moreover, the related reward function is denoted as $R_L = \sum_{n=1}^{N} \sum_{m=1}^{M} \lambda_{n,m} f_{n,m}^{t} - c f_{n,m}^{t}$. After multiple iterations and training, the leader will obtain the maximum profits. Subsequently, the MDP transition policy for the leader is denoted as $\Psi = \{S_L^{'} | S_L, A_L, R_L\}$.

The specific MADFRL algorithm for multiple followers is illustrated in Algorithm 3.

Algorithm 3 The specific MADFRL algorithm

```
Input:
    Followers: $A_{n,m}^{t}$; $x_{n,m,o}$; $y_{n,m,o}$ and $\lambda_{n,\mathrm{m}}$. Leader: $f_{n,m}^{t}$ and $c$.
Output:
    Actor network weight parameters.
 1: for each MBS $n \in \{1, 2, .., N\}$ do
 2:     for each follower $m \in \{1, 2, .., M\}$ do
 3:         Initialize actor network parameters $w$; critic network parameters $Q(S_{n,m}, A_{n,m})$;
 4:     end for
 5: end for
 6: for each $t \in \{1, 2, .., T\}$ do
 7:     for each $n \in \{1, 2, .., N\}$ do
 8:         for each $m \in \{1, 2, .., M\}$ do
 9:             Execute the current task decision-making $A_{n,m}$;
10:             Obtain the reward function $R_{n,m}$;
11:             Transfer to next states $S_{n,m}^{'}$;
12:             Input $A_{n,m}$ and $S_{n,m}$ to critic networks and obtain $Q\left(S_{n,\mathrm{m}}, A_{n,m}\right)$;
13:             Update critic network parameters via $\sum \left(R_{n,m} + \gamma Q\left(S_{n,m}^{'}, A_{n,m}^{'}\right)\right)^{2}$;
14:             Update actor network via $-Q\left(S_{n,m}, A_{n,m}\right)$
15:             Record the state transitions $\Omega = \left\{ S_{n,m}^{'} | S_{n,m}, A_{n,m}, R_{n,m} \right\}$;
16:         end for
17:         Compute the corresponding reward function $R_{n,m}$;
18:         Output actor network parameters $Z_{n,m}$ and transmit them to Layer 3 in order to execute FL aggregation and issuing;
19:     end for
20: end for
```

C. Federated Aggregation and Parameters Issuing Mechanism

After each actor network outputs the offloading policy, we need to appraise these actor network actions to further adjust neural network parameters. However, the information exchange among multiple followers cannot prevent user privacy disclosure. Hence, we propose the FL aggregation and issuing mechanism to centrally update actor network parameters and distributively issue them to each follower. Specifically, the FL aggregation and issuing module belong to Layer 3 for the proposed MAML-MADFRL framework and Layer 3 can collect all actor network parameters from followers in order to average followers' network parameters via different tasks size and time-varying location weights, which can then issue these network weight parameters to each follower. As actor network has lightweight network parameters [30], it is beneficial to decrease parameters transmission overhead and improve communication efficacy in terms of large data volume. Furthermore, all followers receive the global update model $Z(t)$ from the leader according to computation offloading policy. Meanwhile, each follower can obtain actor neural network model $Z_{n,m}(t)$ in terms of corresponding state $S_{n,m}$. Hence, each follower uploads the new model to the leader, which is represented as

$$I_{n,m}(t) = Z(t) - Z_{n,m}(t). \tag{34}$$

Next, the leader receives the uploaded model and further optimizes them via the proposed federated aggregation policy, which are generally defined as

$$Z(t+1) = Z(t) + uI(t), \tag{35}$$

where u is the FL aggregation learning rate and $I(t)$ is represented as

$$I(t) = \frac{|D_{n,\mathrm{m}}| + |L_{\mathrm{n},m}|}{|D_{total}| + |L_{total}|} I_{n,m}(t), \tag{36}$$

where $D_{n,m}$ and D_{total} are task size of each follower and all task sets from followers, respectively. Meanwhile, $L_{n,m}$ is the relative position distance from LEO to ground follower and L_{total} is the sum of total distance for all followers. As illustrated above, larger task bits and relative location distance mean more model update ratio. The specific federated aggregation and issuing mechanism are shown in Algorithm 4.

Remark 4: For multiple followers, we execute FL aggregation and issuing mechanism to weigh actor network parameters, which aims to protect data privacy. Meanwhile, each actor network parameter is issued to corresponding followers in terms of different tasks and location ratios. Hence, larger task

Algorithm 4 The FL aggregation mechanism

```
Input:
    Actor network parameters $Z_{n,m}(t)$;
Output:
    Issue actor network weight for each follower;
1: for each $n \in \{1, 2, .., N\}$ do
2:     for each $m \in \{1, 2, .., M\}$ do
3:         Upload respective actor network parameters $Z_{n,m}(t)$;
4:         Execute the FL aggregation and issuing mechanism
           from (34) to (36);
5:     end for
6: end for
7: Obtain the optimal actor network parameters.
```

and location ratio mean more parameters weight issuing for each follower, which help maximize the processed number of task bits in terms of minimizing privacy overhead and cloud server cost.

V. Performance Analysis

For this section, we explore the corresponding privacy protection-based blockchain verification mechanism for followers and leader, algorithm complexity analysis, long-term task queue constraints and MAML-MADFRL algorithm convergent rate.

A. Transaction Verification

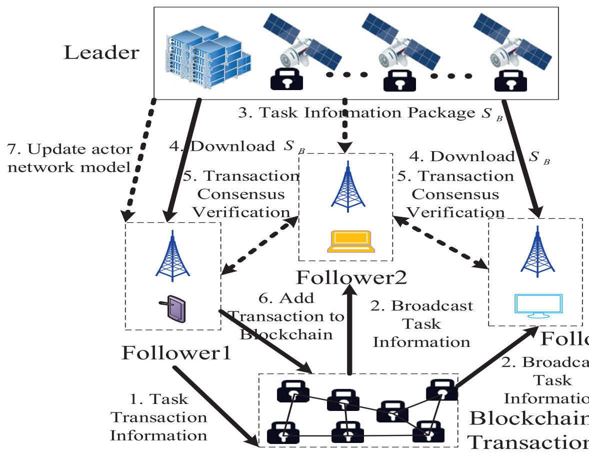
Leader
3. Task Information Package S_B
7. Update actor network model
4. Download S_B
5. Transaction Consensus Verification
4. Download S_B
5. Transaction Consensus Verification
Follower2
6. Add Transaction to Blockchain
2. Broadcast Task Information
Follower1
Follower3
2. Broadcast Task Information
1. Task Transaction Information
Blockchain Transaction

Fig. 3: Privacy protection-based blockchain verification mechanism for followers and the leader.

The complete privacy protection based-blockchain verification mechanism is shown in Fig. 3, which can be divided into several procedures. First, follower 1 generates the task transaction information stored in the blockchain. Next, this task information is broadcast to other followers and transmitted to the leader. Subsequently, the leader collects this task information and packages it into S_B. Moreover, each follower in the MBS starts to download block S_B and the transaction consensus verification is achieved via delegated stakes proof protocols [31]. After the task transaction process is passed via other followers, the coin rewards are returned back to initiators. If it suffers from fraud or tampering, the task transaction is closed. Finally, the verified transaction is added to blockchain and the follower updates actor network parameters.

B. Algorithm Complexity Analysis

In this subsection, we derive computational complexity of the proposed four algorithms, i.e., the Lyapunov drift-plus-penalty algorithm, MAML for MADFRL algorithm, MADFRL algorithm, and FL aggregation mechanism. Specifically, as the Lyapunov drift-plus-penalty algorithm transforms the original problem P2 into $P2^{'}$, the computational complexity of the Lyapunov drift-plus-penalty algorithm is $O(1)$. Next, the computation complexity of MAML for MADFRL can be calculated as $O\left(G + G\Gamma K\right)$, where G is the number of task execution. Subsequently, when executing the task offloading and resource allocation, the MADFRL algorithm complexity is represented as $O\left[MN\left(1 + T\Gamma K\right)\right]$. Finally, after each follower transfers these network parameters to the leader, the algorithm complexity for FL aggregation mechanism is represented as $O\left(MN\right)$.

C. Convergence and Constraints Demonstration

In this subsection, we discuss the asymptotic convergent performance of the proposed MAML-MADFRL algorithm and validate the long-term task queue constraints. Subsequently, we introduce related preliminary knowledge about Lyapunov optimization for verifying the convergent performance and task queue constraints, which is regarded as **W-only** policy independent of task queue backlogs.

W-only policy: The policy only relies on observed network states to choose optimal control actions, i.e., for any observed network states $S_{n,m}$, W-only policy can choose actions according to certain probability distribution, which is represented as

$$\alpha^{*}(t) = \arg\max \Pr\left(\alpha(t) | S_{n,m}(t)\right). \tag{37}$$

Subsequently, the W-only policy does not rely on any queue states. In terms of Lyapunov stability theory [32], when the original problem P2 is feasible, for any $\varpi > 0$, there must exist a W-only policy α, where specific inequalities are formulated as

$$\mathbb{E}[F\left(\alpha\right)] \geq S^{*} - \varpi, \tag{38}$$

$$\mathbb{E}\left[A_{n,m}^{t}\right] \leq \mathbb{E}\left[D_{n,m}^{t1/t2}\left(\alpha\right)\right] + \varpi, \tag{39}$$

where S^{*} is the optimal solution for the original problem via all chosen policies (not limited in W-only policies).

Proof: The detailed proof process can be found in appendix 4.A of [32].

1) Time average function analysis

Based on the introduced W-only policy α, We further derive the asymptotic optimal performance of the proposed MAML-MADFRL algorithm. Because the minimum of drift-plus-penalty algorithm is obtained via all chosen control variables, the derivation is presented as

$$\Theta\left(\vec{Q}(t)\right) = \Delta L\left(\vec{Q}(t)\right) - V\mathbb{E}\{F\} \tag{40}$$

$$\leq X + \mathbb{E}\left[Q_{n,m}^t\left(A_{n,m}^t - D_{n,m}^{t1/t2}(\alpha)\right)\right] - V\mathbb{E}\{F(\alpha)\}$$
$$\leq X + Q_{n,m}^t A_{n,m}^t + \varpi\mathbb{E}\left[Q_{n,m}^t\right] - V\left(S^* - \varpi\right). \tag{41}$$

More importantly, we can let the constant ϖ tend to be 0. Hence, (40) is further simplified as

$$\Delta L\left(\vec{Q}(t)\right) - V\mathbb{E}\{F\} \leq \hat{X} - VS^*, \tag{42}$$

where $\hat{X} = X + Q_{n,m}^t A_{n,m}^t$. Next, we take the summation for both sides of (42) from 1 to T, which is derived as

$$\left(\hat{X} - VS^*\right)T \geq \sum_{t=1}^{T}\mathbb{E}\left[\Delta L\left(\vec{Q}(t) - V\mathbb{E}\{F\}\right)|\vec{Q}(t)\right] \tag{43}$$

$$\geq \mathbb{E}\left[L\left(\vec{Q}(T)\right)\right] - V\sum_{t=1}^{T}\mathbb{E}\left\{F|\vec{Q}(t)\right\}$$

$$\geq -V\sum_{t=1}^{T}\mathbb{E}\{F|\vec{Q}(t)\}.$$

Finally, the time average function is represented as

$$\lim_{T\to\infty}\sum_{t=1}^{T}\frac{1}{T}\mathbb{E}\left\{F|\vec{Q}(t)\right\} \geq S^* - \frac{\hat{X}}{V}. \tag{44}$$

2) Time average queue size

First, we assume the **W-only policy** η (not the optimal policy), which is represented as

$$D_{n,m}^{t1/t2}(\eta) \geq \varsigma, \tag{45}$$

where $\varsigma > 0$. Meanwhile, in actual problems, the cloud server price $\lambda_{n,m}$, the CPU cycle frequency $f_{n,m}^t$ and the block size S_B are bounded because of limited communication, computation and block resources. Hence, we assume that F is a bounded function, which can be represented as

$$S_{\min} \leq F \leq S_{\max}. \tag{46}$$

Hence, the time average queue size is derived as

$$\Delta L\left(\vec{Q}(t)\right) - V\mathbb{E}\{F\} \leq X + Q_{n,m}^t A_{n,m}^t \tag{47}$$
$$- Q_{n,m}^t D_{n,m}^{t1/t2}(\eta) - V\mathbb{E}\{F\}$$
$$\Delta L\left(\vec{Q}(t)\right) - VS_{\max} \leq \hat{X} - VS_{\min} - Q_{n,m}^t\varsigma. \tag{48}$$

Furthermore, when $t = 0$, $\mathbb{E}\left[L\left(\vec{Q}(0)\right)\right]$=0. As $Q_{n,m}^t$ represents the task queue and $A_{n,m}^t$ denotes the received task bit, they can be considered as the constant and we can obtain the value in each round iteration. Next, we directly take the summation and limit for both sides of inequality, which is represented as

$$\mathbb{E}\left[L\left(\vec{Q}(T)\right)\right] \leq T * \hat{X} + \left(S_{\max} - S_{\min}\right)VT - \sum_{t=1}^{T}Q_{n,m}^t\varsigma \tag{49}$$

$$\lim_{T\to\infty}\frac{\sum_{t=1}^{T}\mathbb{E}\left[Q_{n,m}^t\right]}{T} \leq \frac{\hat{X} + \left(S_{\max} - S_{\min}\right)V}{\varsigma}. \tag{50}$$

Because of $\mathbb{E}\left[L\left(\vec{Q}(t)\right)\right] \geq 0$, the time average queue size is proved.

Remark 5: It means that we can adjust the V to asymptotically approach the maximum S^* and satisfy the time average queue size $Q_{n,m}^t$. Specifically, (44) shows that when V is larger, the time average function for the proposed MAML-MADFRL algorithm is closer to the optimal solution S^*. Moreover, the convergent rate for the time average function is $O\left(\frac{1}{V}\right)$. However, (50) indicates that if we only increase the Lyapunov control parameter V, it can cost longer to fulfill the long-term queue constraints, because the convergent rate for the average queue size is represented as $O(V)$. Finally, we show detailed performance results for the Lyapunov control parameter V in the simulation results section.

VI. Simulation Results and Discussion

In this section, we have executed massive simulation experiments to validate the proposed algorithm performance. First, we demonstrate the impact of the Lyapunov control parameter V on time average function and time average queue. Next, we compare the proposed MAML-MADFRL algorithm with other four advanced schemes, which demonstrate that it has better performance gains in terms of price profits, network throughput, and reducing channel interference. Finally, the proposed blockchain-aided verification mechanism can not only protect users' privacy, but also reduce privacy verification overhead. The specific simulation details are as follows.

A. Parameters Settings

In the blockchain-aided SGIN-DT scenario, we explore the proposed MAML-MADFRL algorithm performance via a massive simulation experiments. Specifically, we set that $N = 4$ MBS, $M = 12$ users and $O = 4$ LEOs. Next, the stochastic task arrivals $A_{n,m}^t$ are uniformly distributed at (10, 30) MB and the required CPU number of cycles w is uniformly distributed at (2,000, 4,000) cycle/bit. Moreover, the horizontal distance $x_{n,m,o}$ and vertical distance $y_{n,m,o}$ for LEOs are uniformly distributed at (1,000, 2,000) KM and (500, 2,000) KM. Furthermore, the carrier frequency f_c and light speed c are set as $0.1 * 10^9$ HZ and $3 * 10^8$ m/s. Meanwhile, the path loss $\varepsilon_{n,m,o}^{LoS}$ and $\varepsilon_{n,m,o}^{NLoS}$ are uniformly distributed at (0,1) and (10, 30), respectively. The noise power σ is 10^{-13} W. For MBS, the CPU cycle frequency f_{MBS} is $6 * 10^9$ cycle/s. Moreover, the uplink rate and downlink rate are $0.5*10^{10}$ bit/s and $1 * 10^{10}$ bit/s [12] and [33], respectively.

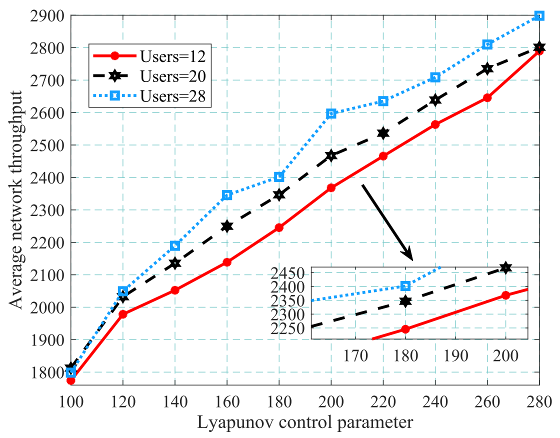

Fig. 4: The average network throughput versus Lyapunov control parameter.

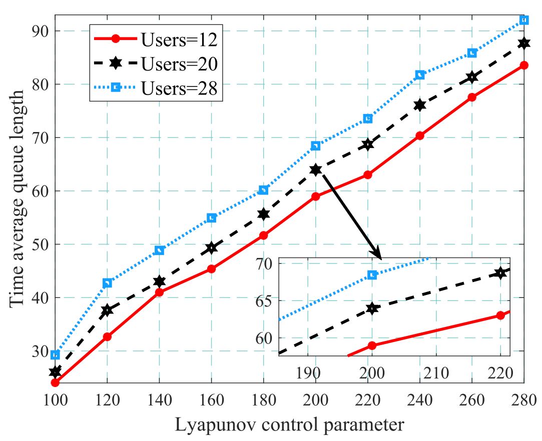

Fig. 5: The time average task queue versus Lyapunov control parameter.

B. Comparison Schemes

We compare the proposed learning framework with other four baseline methods, such as MAPPO [3], MARTO, MAGCS and MAMCC. Moreover, we demonstrate the performance gains for the proposed learning framework in terms of task queue, Lyapunov control parameter, unit energy consumption, network throughput, channel interference, cloud server profits and privacy overhead. Detailed baseline methods are shown as follows.

- MAPPO: When solving the primal problem P1 and P2, the MAPPO employs two neural networks for each agents, including actor network and critic network. Meanwhile, the MAPPO framework utilizes the centralized processing and distribute execution schemes to optimize corresponding DT variables, i.e., $f_{n,m}^t, \alpha_{n,m}^t, a_{n,m,r}^o, S_B$.
- MARTO: For each stochastic task, the agent randomly chooses proper task processing units, such as local processing or LEO computation. However, it is hard to find the optimal task offloading decisions.
- MAGCS: In order to reduce the channel interference from other areas, each agent greedily selects the proper channel according to path poss, i.e., each agent tends to choose the wireless channel with minimized path loss.
- MAMCC: This method allocates the computational resources equally for each agent. While processing the stochastic tasks, each agent obtains the equal CPU cycle frequency to maximize the network throughput and cloud server profits.

C. Performance Evaluation

Specifically, for the average network throughput and queue length given different Lyapunov control parameters V, the time horizon T is a hyper-parameter, which can be preset according to the empirical evidence, i.e., when the number of users is divided into 12, 20 and 28, we can set the different time horizon T to execute the simulation results. As the time horizon T proceeds, the network throughput or queue lengths remain unchanged. Once it is not changed, we can determine the specific time horizon T for different number of users in terms of certain Lyapunov control parameter V, whose specific time horizon T value can be used to calculate the average network throughput. Although the time horizon T for different number of users is different, we only demonstrate the effect of Lyapunov control parameter V on average network throughput, which will further verify the time average function (44). Meanwhile, when we execute the simulation experiments, the size of application task $A_{n,m}^t$ is uniformly distributed at (10, 30) MB for time round t of each user. Next, we introduce the specific performance gains.

1) MAML-MADFRL Performance Gains

Next, we show the performance gains of the proposed algorithm in terms of different Lyapunov control parameters. As shown in Fig. 4, we explore the impact of the Lyapunov control parameter on average network throughput according to different number of users. As the Lyapunov control parameter increases, it brings larger network throughput, which further demonstrates the time average function (44). Evidently, because more terrestrial users bring larger task bits, the proposed learning framework has faster network throughput to adapt to the increasing number of users. Moreover, when there are more users, the proposed method still stabilizes the network queue because of the Lyapunov drift framework, which means that each user can process more task bits under dynamic network environments.

Next, as shown in Fig. 5, we further explore the impact of the Lyapunov control parameter on the time average queue length in terms of different number of ground users. To be simplicity, we set the task size to 15 MB. Specifically, as the number of users increases, the time average queue length is larger since the proposed algorithm needs to adapt to more complex network environments and issues these parameters to more users. Meanwhile, we have more users who independently try to reach the optimal solution, thus the network needs to spend more time to converge, which causes that the queue sizes are expected to increase with the number of users. Moreover, when the Lyapunov control parameter increases, each user needs more time to converge to the optimal queue length, which further demonstrates the time average queue size

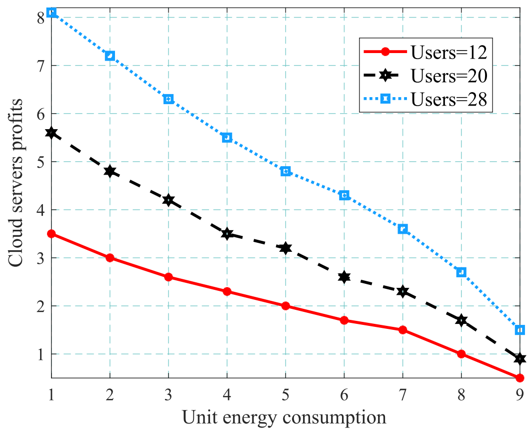

Fig. 6: The cloud servers' profits versus unit energy consumption.

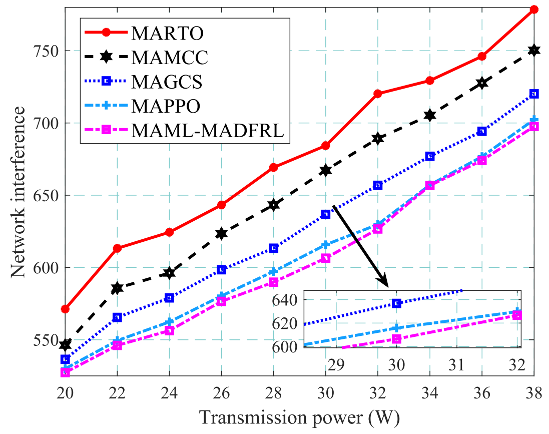

Fig. 7: The mutual channel interference versus transmission power.

(50).

Moreover, we explore the impact of unit energy consumption on cloud servers profits in terms of different number of users as shown in Fig. 6. Because the proposed learning framework can adapt to the dynamic network environment and make the optimal offloading decision for the satellite cloud servers, it can bring more cloud server profits as the number of users increases. Moreover, when the unit energy consumption of cloud servers increases, this brings lower servers profits because it can cause larger energy consumption while serving terrestrial users according to the formula (11). However, the unit energy consumption and the number of terrestrial users have inverse impact on the cloud servers profits. Hence, we need to adjust the unit energy consumption to maximize the cloud server profits in terms of different number of users.

2) *Algorithm Comparisons*

In this sequel, we compare our proposed algorithm with four baseline methods, i.e., MARTO, MAMCC, MAGCS and MAPPO. As shown in Fig. 7, the unit of transmission power is Watt(W). Moreover, the proposed algorithm has lower channel interference since it can better adapt to dynamic network environments, such as stochastic task arrivals and time-varying LEO locations, which can help each terrestrial user make the optimal task offloading decision, CPU cycle frequency and block size. Furthermore, the proposed learning framework can access the optimal wireless channel, which is beneficial to reducing the mutual channel interference among multiple proximal areas and improving the processed number of task bits. As the MARTO method only randomly processes the task either users or remote LEO units, it is hard to find the optimal task scheduling scheme. Meanwhile, MAMCC only equally allocates the computational resources for each agent and MAGCS greedily chooses the wireless channel with minimum path loss for each agent, which all cause higher mutual channel interference. Subsequently, when the transmission power is 32 W, the proposed learning framework has approximately 1% performance gain compared with MAPPO, which demonstrates the superiority of MAML-MADFRL.

As shown in Fig. 8, we explore the impact of transmission bandwidth on network throughput. The unit of transmission bandwidth is megahertz (MHz). As transmission bandwidth increases, the average network throughput for all users gradually increases. This is because larger transmission bandwidth can bring a higher network throughput. Moreover, as the proposed algorithm can help execute the optimal scheduling, choose the optimal wireless channel and adjust the cloud server price dynamically, it has higher network throughput compared with the other four baseline methods. Specifically, MAPPO employs the centralized processing and distributed execution method to optimize the CPU cycle frequency, task scheduling, channel selection and block size, and causes large interaction and computational pressure with an increasing number of states and actions, which is hard to obtain the optimal decisions. Meanwhile, the channel bandwidth is limited in practical scenarios. Hence, we deploy the proposed learning framework in practical systems to save bandwidth while guaranteeing the quality of service. This is because meta learning can accelerate the training process from small batches of samples and the federated aggregation mechanism can help each user adjust the actor network parameters. Compared with MAPPO, the proposed MAML-MADFRL has approximately 1.3% performance gains when the transmission bandwidth is 30 MHz.

As shown in Fig. 9, we explore the impact of cloud server prices on cloud server profits. When the cloud servers price is from 2, 4, 6, 8 to 10, the proposed learning framework has higher cloud servers profits as it can not only adapt to dynamic network environments, but also adjust the satellite cloud price under different CPU cycle frequencies, which can better coordinate the server profits and user's quality of service. Furthermore, MARTO, MAMCC and MAGCS only randomly make the task offloading decisions, equally allocate the computational resources and greedily select the wireless channel, which cannot guarantee the optimal Stackelberg game process. Meanwhile, MAPPO only centrally optimizes these variables, as the interaction process executes, the satellite cloud server is hard to obtain the optimal server profits. Hence,

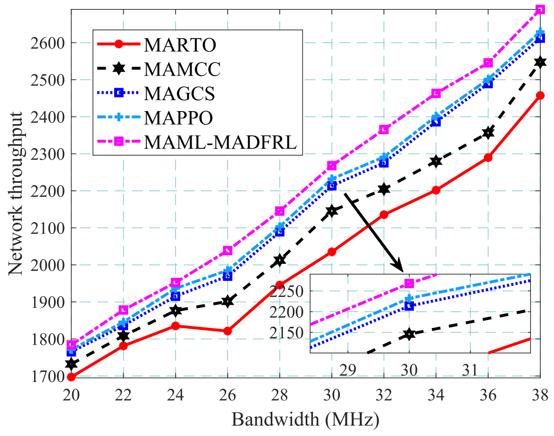

Fig. 8: The network throughput versus transmission bandwidth.

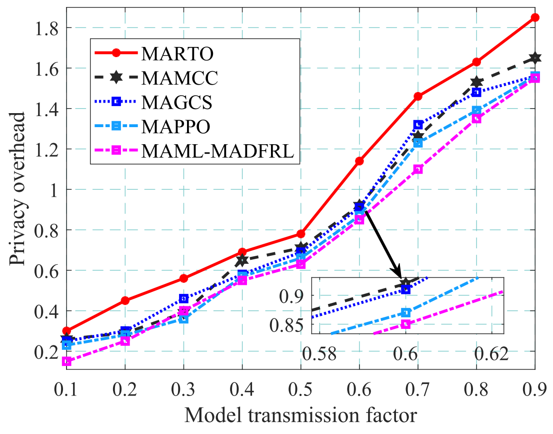

Fig. 10: The privacy overhead versus model transmission factor.

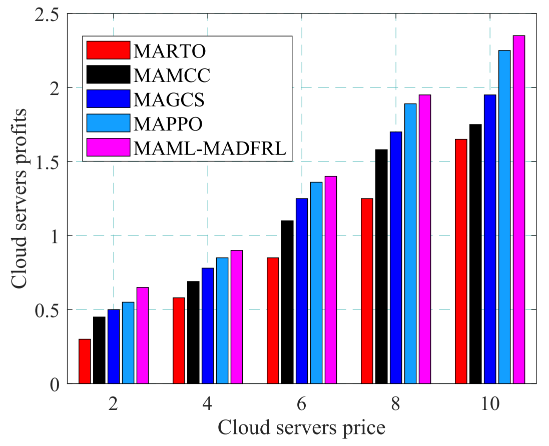

Fig. 9: The cloud servers' profits versus cloud servers' price.

compared with MARTO, MAMCC, MAGCS and MAPPO, the proposed learning framework has approximately 55%, 45%, 15%, and 6% performance gains when the cloud server price is 6.

3) Privacy Protection Overhead

In this sequel, we explore the impact of model transmission factor on privacy protection overhead. As shown in Fig. 10, as the model transmission factor increases, the privacy overhead gradually increases since a larger model transmission factor brings more privacy overhead according to the formula (10). However, the proposed algorithm has lower privacy overhead since the blockchain-aided federated aggregation and parameters issuing mechanism can help each user identify the network attack and choose the optimal block. Furthermore, the proposed privacy protection-based blockchain verification protocol can better verify the transaction process and prevent tampering and data leakage. Moreover, when the model transmission factor is greater than 0.5, the MARTO algorithm is hard to stabilize the privacy overhead because of random task scheduling. Compared with the MARTO algorithm, the MAML-MADFRL learning framework can achieve approximately 20.7% performance gains when the model transmission factor is 0.8, which demonstrates that the proposed learning framework can reduce the privacy protection overhead and strengthen the transaction verification.

Next, we explore the impact of MBS computational capability on the privacy overhead. As shown in Fig. 11, when the computational capability of MBS gradually increases, the privacy overhead decreases. This is because the MBS with high computational capability accelerates network parameters aggregation. Moreover, the proposed algorithm has lower privacy overhead compared to other baseline methods, since MAML-MADFRL can not only speed up the parameters aggregation, but also allocate the optimal block for each user, which further reduces the privacy protection overhead. Meanwhile, the corresponding MARTO, MAMCC, MAGCS, and MAPPO methods do not consider the privacy protection-based blockchain verification protocols, which cannot choose the optimal block size for each user and efficiently prevent data leakage. Compared with the other four baseline methods, i.e., MARTO, MAMCC, MAGCS and MAPPO, the MAML-MADFRL learning framework approximately has 67%, 55%, 41% and 20% performance gains when the MBS computational capability is $5 * 10^9$ cycle/s.

VII. Conclusions

In this paper, we consider a blockchain-aided two-stage Stackelberg game model in the SGIN-DT scenario to maximize the network throughput and cloud servers' profits in terms of minimum privacy protection overhead, stochastic task arrival, time-varying LEO locations and variables coupling for long-term task queue constraints and short-term computation offloading. Next, we propose a Lyapunov stability theory-based MAML-MADFRL learning framework to process the task scheduling, reduce the channel interference, optimize the CPU cycle frequency, and allocate the block size, which further achieve the optimum cloud servers profits via optimizing the cloud servers' prices. Moreover, we analyse the corresponding blockchain verification mechanism, the computational complexity of the proposed algorithm, and algorithm performance

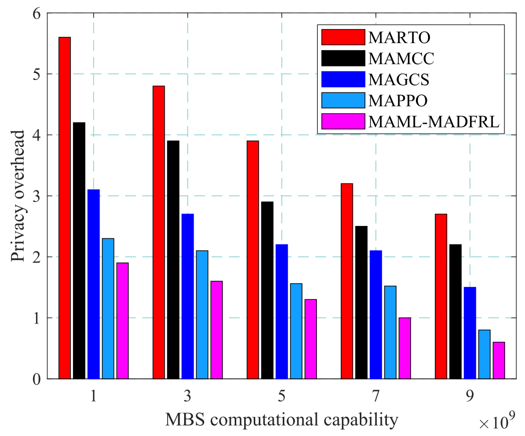

Fig. 11: The privacy overhead versus MBS computational capability.

bounds as well as task queue convergence. Finally, massive simulation results show that the proposed learning framework achieves higher network throughput, cloud servers' profits, and lower privacy verification overhead compared with MARTO, MAMCC, MAGCS and MAPPO.

References

[1] B. Feng, H. Zhou, H. Zhang, G. Li, H. Li, S. Yu, and H.-C. Chao, "Hetnet: A flexible architecture for heterogeneous satellite-terrestrial networks," *IEEE Network*, vol. 31, no. 6, pp. 86–92, Nov./Dec. 2017.

[2] Y. Gong, H. Yao, J. Wang, M. Li, and S. Guo, "Edge intelligence-driven joint offloading and resource allocation for future 6G industrial Internet of Things," *IEEE Transactions on Network Science and Engineering, Early Access*, 2022.

[3] Y. Gong, H. Yao, D. Wu, W. Yuan, T. Dong, and F. R. Yu, "Computation offloading for rechargeable users in space-air-ground networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3805–3818, Mar. 2022.

[4] L. Song, Z. Han, B. Di, and H. Zhang, *Aerial Access Networks: Integration of UAVs, HAPs, and Satellites*. Cambridge, UK: Cambridge Univ. Press, 2021.

[5] R. Luo, H. Jin, Q. He, S. Wu, and X. Xia, "Cost-effective edge server network design in mobile edge computing environment," *IEEE Transactions on Sustainable Computing, Early Access*, 2022.

[6] A. Alnoman and A. Anpalagan, "Computing-aware base station sleeping mechanism in H-CRAN-cloud-edge networks," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 958–967, July-Sept. 2019.

[7] W. Fan, L. Yao, J. Han, F. Wu, and Y. Liu, "Game-based multitype task offloading among mobile-edge-computing-enabled base stations," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 691–17 704, Dec. 2021.

[8] S. Mihai, M. Yaqoob, D. V. Hung, W. Davis, P. Towakel, M. Raza, M. Karamanoglu, B. Barn, D. Shetve, R. V. Prasad *et al.*, "Digital twins: a survey on enabling technologies, challenges, trends and future prospects," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2255–2291, 4th Quart., 2022.

[9] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-twin-enabled 6G: Vision, architectural trends, and future directions," *IEEE Communications Magazine*, vol. 60, no. 1, pp. 74–80, Jan. 2022.

[10] Y. Lu, S. Maharjan, and Y. Zhang, "Adaptive edge association for wireless digital twin networks in 6G," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16 219–16 230, Nov. 2021.

[11] D. Van Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre, and T. Q. Duong, "Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse," *IEEE Wireless Communications Letters*, vol. 11, no. 8, pp. 1733–1737, Aug. 2022.

[12] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud–edge–end in IoT: A blockchain-assisted collective Q-learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 694–12 704, Aug. 2020.

[13] Y. Cao, X. Ren, C. Qiu, and X. Wang, "Hierarchical reinforcement learning for blockchain-assisted software defined industrial energy market," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6100–6108, Sep. 2022.

[14] J. Li, D. Niyato, and Z. Han, *Cryptoeconomics: Economic Mechanisms behind Blockchains*. Cambridge, UK: Cambridge Univ. Press, 2023.

[15] N. Q. Hieu, T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and E. Elmroth, "Deep reinforcement learning for resource management in blockchain-enabled federated learning network," *IEEE Networking Letters*, vol. 4, no. 3, pp. 137–141, Sep. 2022.

[16] J. Du, W. Cheng, G. Lu, H. Cao, X. Chu, Z. Zhang, and J. Wang, "Resource pricing and allocation in MEC enabled blockchain systems: An A3C deep reinforcement learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 33–44, Jan.-Feb. 2022.

[17] R. Ma, Z. Yi, Y. Xiang, D. Shi, C. Xu, and H. Wu, "A blockchain-enabled demand management and control framework driven by deep reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 1, pp. 430–440, Jan. 2023.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ACM International Conference on Machine Learning*, Sydney, Australia, pp. 1126–1135.

[19] X. Cao, B. Yang, C. Yuen, and Z. Han, "HAP-reserved communications in space-air-ground integrated networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8286–8291, Aug. 2021.

[20] H. Guo, J. Li, J. Liu, N. Tian, and N. Kato, "A survey on space-air-ground-sea integrated network security in 6G," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 53–87, 1st Quart., 2021.

[21] K. Fan, B. Feng, X. Zhang, and Q. Zhang, "Network selection based on evolutionary game and deep reinforcement learning in space-air-ground integrated network," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1802–1812, May–Jun. 2022.

[22] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5098–5107, Jul. 2020.

[23] P. Bellavista, C. Giannelli, M. Mamei, M. Mendula, and M. Picone, "Application-driven network-aware digital twin management in industrial edge environments," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7791–7801, Nov. 2021.

[24] Z. Lei, H. Zhou, W. Hu, G.-P. Liu, S. Guan, and X. Feng, "Toward a web-based digital twin thermal power plant," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1716–1725, Mar. 2021.

[25] C. Ma, J. Li, L. Shi, M. Ding, T. Wang, Z. Han, and H. V. Poor, "When federated learning meets blockchain: A new distributed learning paradigm," *IEEE Computational Intelligence Magazine*, vol. 17, no. 3, pp. 26–33, Aug. 2022.

[26] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2964–2973, Apr. 2020.

[27] L. Cui, X. Su, and Y. Zhou, "A fast blockchain-based federated learning framework with compressed communications," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3358–3372, Dec. 2022.

[28] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-Edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, Sept.-Oct. 2019.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[30] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, "Federated multiagent actor–critic learning for age sensitive mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1053–1067, Jan. 2021.

[31] Q. Li, J. Wu, J. Quan, J. Shi, and S. Zhang, "Efficient quantum blockchain with a consensus mechanism QDPoS," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3264–3276, Aug. 2021.

[32] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, Nov. 2010.

[33] Y. Gong, H. Yao, J. Wang, L. Jiang, and F. R. Yu, "Multi-agent driven resource allocation and interference management for deep edge networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 2018–2030, Feb. 2022.