Blockchain-Enabled Task Offloading With Energy Harvesting in Multi-UAV-Assisted IoT Networks: A Multi-Agent DRL Approach

Abegaz Mohammed Seid, *Member, IEEE*, Jianfeng Lu, *Member, IEEE*, Hayla Nahom Abishu, and Tewodros Alemu Ayall

***Abstract*— Unmanned Aerial Vehicle (UAV) is a promising technology that can serve as aerial base stations to assist Internet of Things (IoT) networks, solving various problems such as extending network coverage, enhancing network performance, transferring energy to IoT devices (IoTDs), and perform computationally-intensive tasks of IoTDs. Heterogeneous IoTDs connected to IoT networks have limited processing capability, so they cannot perform resource-intensive activities for extended periods. Additionally, IoT network is vulnerable to security threats and natural calamities, limiting the execution of real-time applications. Although there have been many attempts to solve resource scarcity through computational offloading with Energy Harvesting (EH), the emergency and vulnerability issues have still been under-explored so far. This paper proposes a blockchain and multi-agent deep reinforcement learning (MADRL) integrated framework for computation offloading with EH in a multi-UAV-assisted IoT network, where IoTDs obtain computing and energy resources from UAVs. We first formulate the optimization problem as the joint optimization problem of computation offloading and EH problems while considering the optimal resource price. And then, we model the optimization problem as a Stackelberg game to investigate the interaction between IoTDs and UAVs by allowing them to continuously adjust their resource demands and pricing strategies. In particular, the formulated problem can be addressed indirectly by a stochastic game model to minimize computation costs for IoTDs while maximizing the utility of UAVs. The MADRL algorithm solves the defined problem due to its dynamic and large-dimensional properties. Finally, extensive simulation results demonstrate the superiority of our proposed framework compared to the state-of-the-art.**

Manuscript received 16 February 2022; revised 16 June 2022; accepted 30 June 2022. Date of publication 13 October 2022; date of current version 22 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62072411 and in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR21F020001. *(Corresponding author: Jianfeng Lu.)*

Abegaz Mohammed Seid and Jianfeng Lu are with the School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, China (e-mail: mamsied2002@gmail.com; lujianfeng@wust.edu.cn).

Hayla Nahom Abishu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: nahinati@gmail.com).

Tewodros Alemu Ayall is with the Department of Computer Science and Engineering, Zhejiang Normal University, Jinhua, Zhejiang 321004, China (e-mail: meettedy2123@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3213352.

Digital Object Identifier 10.1109/JSAC.2022.3213352

***Index Terms*— Blockchain, computation offloading, energy harvesting, Internet of Things, MADRL, UAV.**

I. Introduction

IN THE fifth-generation (5G) and beyond fifth-generation (B5G) network paradigm, unmanned aerial vehicle (UAV) technology and communication networking have attracted different interest groups and researchers. Due to the flexibility and low-cost deployment, UAVs have numerous benefits in the Internet of Things (IoT) edge network environment. UAVs have been used as flying base stations (BSs) to maximize energy efficiency and throughput, as well as for emergency communication [1], [2], [3]. For instance, when some parts of the IoT edge network malfunctions or infrastructure such as BSs fails due to disaster, UAVs can be immediately deployed as aerial BSs (ABSs) to play a critical role in post-disaster network management [4], [5], [6], [7].

Although most IoT devices (IoTDs) run real-time applications in the UAV-assisted IoT network, their application scope is severely limited due to the limitation of computation resources. With the emergence of computation offloading paradigms, these IoTDs can offload their application execution tasks to the nearest computational nodes. Nevertheless, the task computing or offloading performance may be hindered by inadequate IoTD battery energy, resulting in IoTD application terminations leading to service disruptions. To resolve this bottleneck, the traditional solutions mainly include recharging the battery regularly and employing a high capacity battery. However, installing high large-capacity batteries on IoTDs will incur high hardware costs and thus, not an efficient solution. Most IoTDs are always on the move, thus recharging batteries frequently becomes difficult and sometimes not practicable [8]. The seamless operation of the current wireless network requires prolonged battery life to avail enough energy and the efficient utilization of the available energy by its edge devices. Besides, some parts of the network are vulnerable to disasters. Due to these issues, the network becomes hampered in the processing of real-time applications within a given time slot.

To this end, energy harvesting (EH) technology is a promising approach to prolong the battery life of IoTDs [9].

Also, Wireless power transfer (WPT) is a promising technology to provide low-power IoTDs energy replenishments [10], [11]. Some of the regularly harvested energy are from various sources such as thermal, solar, vibration, and wireless radio frequency (RF) sources [12]. Recent works have been done on machine learning-based offloading with EH/transfer in IoT edge network [13], [14], [15], [16]. Combining the EH approach and computation offloading in a UAV-assisted IoT network environment will increase computation sustainability to achieve quality of service (QoS) satisfaction. However, the WPT technology is vulnerable to security attacks due to irregular energy attackers, which results in high energy consumption and energy loss during transmissions [10]. Blockchain technology is a digital ledger in which transactions are recorded using an immutable cryptographic signature/hashing, making it impossible to alter, hack, or compromise the system. It is now being used to ensure the security and privacy of transactions between untrusted nodes in various industries. Thus, employing blockchain technology as one of the promising paradigms in the wireless networks can empower mobile networks, ensure security, privacy and services. The authors in [17] and [18] integrated blockchain into wireless networks to control and manage resource utilization, secure resource transactions, and ensure security for efficient resource sharing. UAV-assisted network combined with blockchain technology is applied in solving different problems [19], [20], [21], [22].

The combination of blockchain and machine learning methods offers various benefits to 5G and B5G network applications, such as data authenticity assurance, data consumption management, audit trails, and providing new values to business processes through automation facilitated by smart contracts. Blockchain is used to store the deep reinforcement learning (DRL)/reinforcement learning (RL) model-related data on the blockchain network, which reduces the chances of error in the DRL/RL models. The blockchain network also does not allow for duplication, missing, or noisy data, which are essential characteristics for ML-based models [23], [24], [25]. Dynamic resource allocation and pricing in the UAV-enabled mobile edge computing (MEC) network studied in [21], the UAVs act as ABS to collect data from IoTD. They formulated the optimization problem as a Stackelberg game and utilized blockchain technology with unsupervised hierarchical deep learning (HDL) and deep Q-learning (DQL) schemes.

Moreover, there are still some challenges to be solved in the aerial to ground (ATG) network. First, due to the dynamic network environment, it is challenging to perform an efficient task offloading with EH for IoTD during an emergency on the ground network. In the ATG network, when IoTDs request diverse resources, the local ground MEC server may become overburdened. Some IoTDs are out of the network coverage due to their mobile nature. Second, motivating the resource providers (RPs) to allocate efficient resources with the optimal price is a problem that needs to be investigated carefully. Third, security is a critical issue. It is a key enabler of the current and upcoming networks in the ATG network scenarios that require a secure and intelligent system. Therefore, designing a framework that jointly optimizes the computation offloading with EH problems to minimize the computation costs while maximizing the resource provider's income or profit.

Motivated by the limitations of previous works and above mentioned challenges, in this paper, we integrate multi-agent DRL and permissioned blockchain (consortium blockchain) into UAV-enabled IoT networks to propose intelligent and secure computation offloading and EH. We first propose a blockchain-empowered distributed and secure computation offloading with an EH framework where IoTDs act as resource requesters (RRs) and UAV cluster heads (UCHs)/BSs act as RPs to perform computation offloading and EH, and BSs/UCHs act as verifiers to maintain a permissioned blockchain. We utilize a Stackelberg game model to formulate the problem and transformed it into Markov decision proces (MDP) model to obtain optimal decision policies. Due to multi-objective problems and the high dynamics of the UAV-enabled IoT network environment, we exploit the advanced DRL approach to learn the dynamic network environment and time-varying wireless channel conditions of the network status and then design an optimal computation offloading with an EH algorithm between RRs and RPs. The main contributions of this work are summarized as follows.

1) We propose a MADRL framework for dynamic computation offloading for IoTDs with EH in a multi-UAV-assisted IoT network to minimize computation cost and resource price. It can help the IoTD learn about the environment and make the optimal computation offloading and EH decisions while obtaining minimum resource pricing in the ATG network. To control IoTDs' energy-hungry issue, the EH is executed first before the IoTD tasks are offloaded or executed locally.
2) To ensure secure transactions between RPs and RRs, we deploy a consortium blockchain with smart contracts, resulting in decentralized resource allocation among RPs and RRs. In order to participate in computation offloading and EH, entities such as RPs, RRs, and consensus nodes must first create an account with public parameters and cryptographic keys and obtain a unique identification address.
3) We formulate a task offloading problem for IoTDs with EH as a multi-learder multi-follower Stackelberg game to make decisions and control the RP and RR's self-ishness. The formulated Stackelberg game-based optimization problem is transformed into an MDP model (stochastic game) and we adopt a model-free Multi-agent deep deterministic policy gradient (MADDPG) algorithm to control the large and continuous action space to maximize long-term reward and find the optimal offloading decision and EH policies, and pricing.
4) We conduct extensive simulations to analyze the proposed algorithm's performance compared with three benchmarks: deep deterministic policy gradient (DDPG), Asynchronous Advantage Actor Critic (A3C), and Dueling- deep Q-network (DQN) algorithms. The simulation results illustrate that our proposed algorithm achieve minimal system cost and optimal prices in a dynamic ATG network environment.

TABLE I
SUMMARY OF RELATED WORKS

Paper	Scenario	Algorithms and Agent	Problem	Objective
[14]	MEC system	Q-learning, MDP, MEC server: single agent	Computation offloading with EH	Maximize profit of data computation and the cost of execution latency
[21]	Blockchain based UAV-enabled MEC network	Unsupervised HDL with DQL, Blockchain, Stackelberg game, BS and Peer: multi-agent	Resource management and pricing	Maximize payoff
[22]	UAV-assisted M2M Communication	DQN, MDP, MEC Server: single agent	Data transmission, security and reliability	Ensure security, maximize computation capacity and throughput
[26]	UAV-aid WPT in MEC	Successive convex approximation algorithm	Computation offloading and energy transfer	Minimize UAV energy consumption
[27]	UAV-assisted Power Transfer	DQN, MDP, single agent	Data collection and EH	Minimize total packet loss
[28]	WPT in MEC	Federated learning, Stackelberg game, Blockchain, multi-agent	Incentive and power transmission	Maximize system efficiency and user profits
Our work	Multi-UAV-assisted IoT network	MADDPG, Stackelberg game, Stochastic game, Blockchain, IoTDs and RPs: multi-agent	Computation offloading and EH, Minimize resource price	Minimize computation cost, Maximize RP utilities

The rest of the paper is organized as follows: Section II introduces the related works. Section III presents the system model. Section IV discusses the problem formulation and Stackelberg game analysis. The proposed solution discussed in Section V. Section VI presents the simulation results and analysis. Finally, we conclude this work in Section VII.

II. RELATED WORKS

Recently, several studies on EH and computation offloading in current wireless technologies have been conducted. Wei et al. [14], proposed RL-based dynamic edge computation offloading for IoT with EH to extend the lifetime of IoT systems. In this study, the computation offloading process is modeled as the MDP to deal with the unpredictability and complexity of the generated data and harvested energy. The Q-learning algorithm is also used to determine the best offloading policy. To empower IoT networks, UAVs act as ABS to provide computation service and transfer energy to IoTDs. In [26], the authors investigated UAV-assisted wireless powered cooperative MEC network, in which UAVs served as energy transmitters, and MEC servers provided energy and computing services to sensor devices. The authors solved the optimization problem by using a successive convex approximation-based algorithm that minimizes the total required energy of the UAV. They jointly optimized the offloading bits, the CPU frequencies, transmit power at the active sensor devices, and UAV trajectory. The authors in [27] proposed an on-board deep Q-network (DQN) based UAV-assisted ground IoT network to transfer power and collect data from IoTDs. The problems are formulated by the MDP to minimize data packet loss by optimizing charging and data collection decisions.

However, offloading intensive tasks and transferring power in IoT network is vulnerable to attack. In [10], the authors designed a distributed and secure WPT architecture integrated with blockchain. To achieve the prolonged IoTD battery and secure communication between each device by EH, [28] designed a novel wireless power edge intelligent framework. Their framework integrates blockchain application to ensure peer-to-peer (P2P) energy and information sharing and adopted multiple agent learning schemes to optimize the utilities. In [29], the authors proposed a blockchain-based cooperative and distributed computation offloading for data processing and mining tasks in Industrial IoT network to improve scalability and minimize system costs. In [30], the authors proposed a decentralized offloading architecture with blockchain, and [31] ensured trust among UAVs and other IoTDs by blockchain-based architecture in UAV-aided wireless network. In [32], the authors proposed a blockchain-enabled UAV-assisted IoT network framework scenario that includes different technologies such as UAV edge computing, UAV charging, blockchain, and virtual currency. To ensure data transmission, security, and reliability in machine-to-machine (M2M) communication, [22] designed UAV-assisted data transmission with blockchain-enabled M2M communication in the MEC network. The authors formulated optimization problems using the MDP model and solved them using Dueling-DQN to maximize the data computation capacity and throughput of the blockchain system.

In the 5G and B5G networks, the integration of blockchain, game theory, and DRL is applicable in different optimization problems such as computation offloading, resource trading, and energy transfer [28] and [33]. The Stackelberg game is widely used in utility optimization problems between RPs and RRs, allowing them to maximize utility while minimizing cost per unit resource. The authors in [33] studied a multi-agent-based resource trading in blockchain-based Industrial IoT scenario and adopted the Stackelberg game to optimize the resource and price between RPs and RR.

We present a comprehensive summary of some closely related works in TABLE I. These studies have presented resource allocation/computation offloading and energy transfer/EH issues in different scenarios. The works [14], [22], [26], and [27] do not consider the security in the EH and computation offloading/resource allocation, and the problem is multi-objective with a continuous action space environment. Reference [28] does not consider the ATG network environment. Moreover, reference [21] focuses on resource management, secured data collection, and processing tasks with minimum cost, where the UAVs are used to relay tasks of peers to the blockchain server but do not perform computation tasks of peers and transfer energy to the peers in emergency

TABLE II
LIST OF KEY NOTATIONS

Notations	Description	Notations	Description
$\mathcal{U}$	Set of UCHs	$d_{it}(t)$	Distance between UCH and IoTD
$\mathcal{N}$	Set of BSs	Δ_i	Task of IoTD i
$\mathcal{X}$	Set of active computational nodes	$E_I^{iot}(t)$	Overall energy consumption of IoTD i
$\mathcal{I}$	Set of IoTDs	$T_i(t)$	Local execution delay
ω^f	Unit price of computation resource	$E_i(t)$	Local energy consumption
μ^{eh}	Unit price of EH	$T_{iu}^{tr}(t)$	Computation offloading task transmission time of IoTD
$\xi_{ix}(t)$	Connection status in time slot t	$T_{iu}^{exe}(t)$	Computation offloading task execution time of IoTD
$\alpha_{ix}(t)$	Offloading decision variable in time slot t	$E_{iu}^{tr}(t)$	Computation offloading task transmission power of IoTD
$e_{ix}(t)$	EH decision in time slot t	$E_{iu}^{exe}(t)$	Computation offloading task execution power of IoTD
υ_i	Power conversion efficiency	$\mathcal{P}$	Set of resource prices set by RPs
$\Psi_i(t)$	Service price of harvested energy	$\mathcal{D}$	Set of computation resource demand
$\bar{\Psi}_i(t)$	Service price of computation resource	Θ	Set of energy demand
β_i	Decision variable of EH and computation offloading	D_i	Computation resource demand of RR i
R_x	Reward of RP (Utility)	P_x	Price set by RP i
Θ_i	Energy demand of RR i	$\boldsymbol{\beta}$	Set of decision includes computation offloading and EH
δ^t	Weighted parameter of time	β_i^*	Optimal decision variable of IoTD i
δ^e	Weighted parameter of energy	Θ_i^*	Optimal energy demand of RR i
D_i^*	Optimal computation demand of RR i	p_x^*	Optimal price of RP i

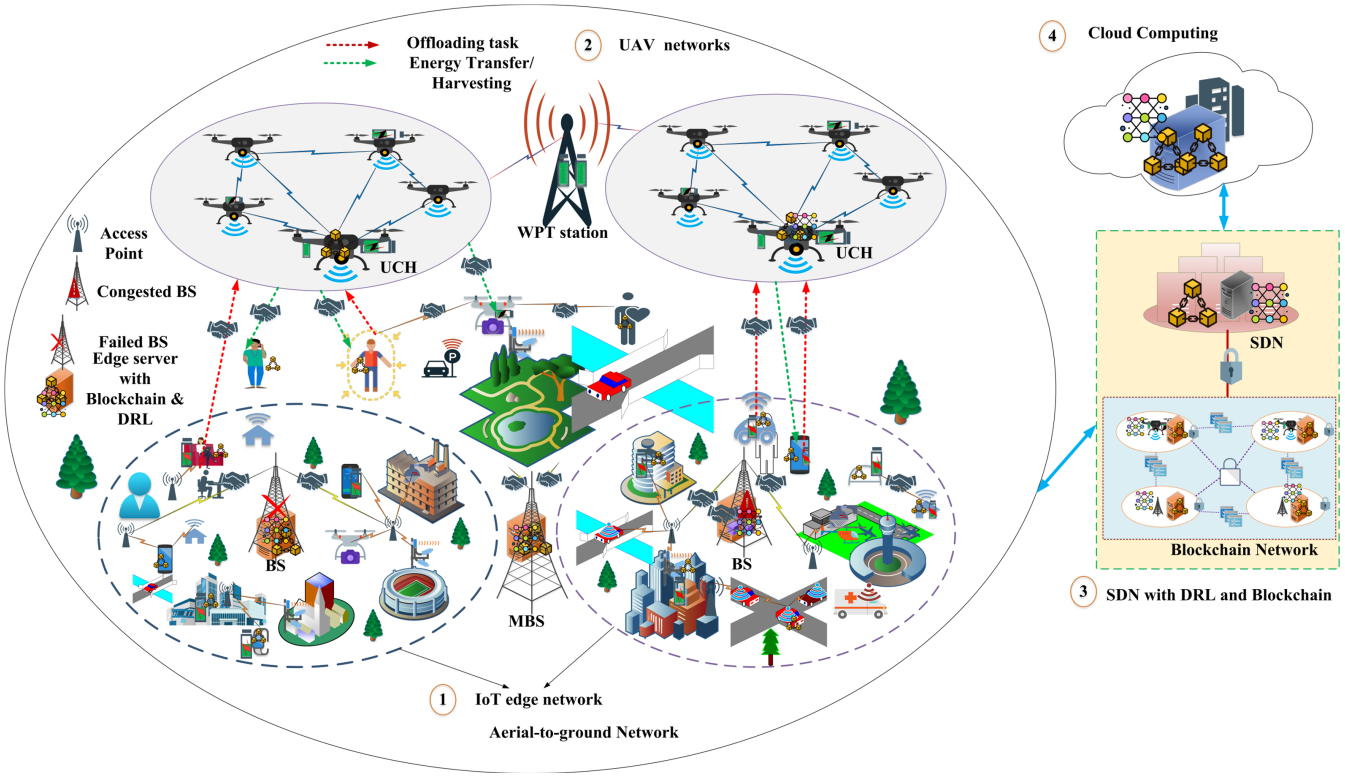


Fig. 1. System architecture.

scenarios. Therefore, our proposed new framework in the ATG network environment is different from the above works. Our proposed ATG framework resolves the ground IoT network optimization problems that arise during an emergency or when the local ground MEC server becomes overburdened. We deploy UAVs to provide computation offloading and EH services to the mobile devices, allowing them to keep their communication in these cases. For efficient and secure computation offloading with EH and handling the interaction between RPs and RRs in a dynamic environment, we proposed to integrate blockchain and DRL. The joint integration of such disruptive technologies in the ATG network can unlock its full potential, resulting in reduced computation costs and increased RPs utilities.

III. SYSTEM MODEL

Fig. 1 depicts the proposed blockchain-enabled multi-UAV assisted IoT network system architecture. We consider a cloud computing, central network controller (SDN), one MBS with MEC server, a clustered UAV network swarm over different small cells consisting of I IoTDs, and access points. Moreover, blockchain is deployed at the edge computing environment, where operations such as consensus, transaction management, and other operations that require powerful servers are per-

formed. Each local BS with a local MEC server and UCH provides resource allocation and computation offloading services to edge IoTDs in each cell. The proposed system is managed by an SDN controller equipped with DRL and consortium blockchain. As shown in Fig. 1, the system architecture consists of the following main components.

(1) *The IoT edge network:* This network consists of heterogeneous IoTDs such as sensor devices, UAVs, smart vehicles, smartphones, smart health monitoring and other devices connected to the network through blockchain. Each IoTD has a blockchain account to join the network to perform different operations such as harvesting energy and task offloading into and from the nearest computational server, either on BS or UCH. The local MEC servers in this network can provide resources for IoTDs and acts as a blockchain entity to establish trust between IoTDs and UAV networks to protect the malicious users. Additionally, there are access points and base stations (BSs) to provide network connections.

(2) *The UAV networks* are controlled by UCH and can access the WPT station to recharge the UAV battery. It has a blockchain network to secure the communication between the UAV network and IoTDs. The edge servers on the aerial and ground networks are also one of the blockchain entities to establish secure communication, transactions with SDNs and IoTDs on the blockchain network. The UCH can act as an RP to provide services such as offloading and EH to IoTDs. (3) *SDN controller* is used to integrate the ATG networks and solve complex computation tasks with EH of IoTDs. In addition, SDN contains MADRL and blockchain network entities to control IoTDs, access, and transaction mining process. (4) *Cloud computing* includes several virtual machines with high-performance computation and storage capacities to solve complex computation tasks and integration management in the ATG network. It consists of blockchain and DRL applications that communicate securely with the SDN controller. (5) *A consortium blockchain* is a lightweight and permissioned blockchain platform that ensures the security and privacy of computation offloading and EH transaction records kept by untrusted nodes in UAV-assisted IoT networks. It requires less resources to reach a consensus, which reduces the computational cost of the consensus process [34] and [35].

A. Network Model

The MEC server serves a massive number of heterogeneous IoTDs. Some of the IoTDs (sensors on the vehicle, body sensors, wearable devices) are dynamically movable. Perhaps, those devices may be far from the network coverage, and malfunctioning may exist at a section of the ground network as shown in Fig.1. Due to the above four possibilities, the network performance will be degraded. Therefore, we deploy UAV as ABS to enable the IoT edge network. The multi-UAV network is organized in clusters, swarming over the IoT edge network and accomplishing their mission-critical tasks. A list of key notations are summarized in TABLE II. Let $u \in \mathcal{U} = \{1, \ldots, U\}$, $n \in \mathcal{N} = \{1, \ldots, N\}$, and $x \in \mathcal{X} = \{1, \ldots, X\}$ denote the set of UCHs, BSs, and active computational nodes, respectively. The set of IoTDs is denoted by $i \in \mathcal{I} = \{1, \ldots, I\}$, where an IoTD i is served by its corresponding active computational node x. The coordinates of an IoTD i and UCH u are denoted by $\{x_i, y_i\}$ and $\{X_u, Y_u, h_u\}$, respectively, where h_u is the altitude of UCH u. The IoTDs generate different tasks but have limited computing capability and battery life to complete them. Thus, it must extend its battery life by harvesting energy from UCHs/RPs and offloading tasks to the nearest computational node for processing. The linear distance between IoTD i and UCH u is expressed as:

$$d_{iu}(t) = \sqrt{(X_u - x_i)^2 + (Y_u - y_i)^2 + h_u^2}, \quad \forall i \in \mathcal{I}, u \in \mathcal{U}. \tag{1}$$

We consider that each IoTD i generates either delay-sensitive or delay-tolerant task Δ_i and each task has three profiles indicated as $[\Lambda_i,\ F_i,\ \Gamma_i]$, where Λ_i denotes the size of the offloading task, F_i is the required computation capacity (CPU cycles), and Γ_i denotes latency constraint of a task. The computation task of IoTD i at time slot t is expressed as:

$$\Delta_i(t) = (\Lambda_i, F_i, \Gamma_i). \tag{2}$$

The IoTD i can be associated with the nearest UCH u on hovering position for task offloading and EH. Hence, UCH u can serve (task offloading and recharging) at most I IoTDs at time slot t. Let the binary variable $\xi_{ix}(t) \in \{0, 1\}$ indicates the connection status between computational node x and IoTD i. If $\xi_{ix}(t) = 1$, the subchannel between the IoTD i and BS or UCH is available, and the IoTD can offload the task and recharge batteries; else, the subchannel is unavailable, which means either the computation node is overloaded or broken down. The IoTD i is associated with UCH u to offload task and/or harvest RF energy from UCH u, it must be under UCH u's coverage range. Let $\mathcal{U}_i$ denote the set of UCHs which cover IoTD i, and is expressed as $\mathcal{U}_i = \{u | d_{iu} \leq h_u tan_{\theta_u}\}, \forall i \in \mathcal{I}$, θ_u is angle of elevation of UCH u. Without loss of generality, we assume that the IoTD with a low battery level can harvest RF energy from UCH before offloading or executing the task, and the UCH transmits the energy before receiving and executing tasks from IoTD. After harvesting energy from UCH, the IoTD will decides to compute tasks locally or offloads to computational nodes by using the offloading decision variable $\alpha_{ix}(t) \in \{0, 1\}$.

The ATG communication channel depends on the altitude, angle of elevation, and type of propagation environment. The ATG communication links can be modeled by a probabilistic path loss (PL) model containing the Line of Sight (LoS) and non-LoS (NLoS) in different probabilities of occurrence [5] and [36]. The LoS, NLoS and path loss between UCH u and IoTD i expression are similar to [5]. To control interference among IoTDs during task offloading, we consider Orthogonal Frequency Division Multiplexing (OFDM) technique. The transmission rate of IoTD i associated with UCH u is expressed as:

$$r_{iu}(t) = \frac{B}{I} log_2\big(1 + \frac{P_i}{\sigma^2 10^{\bar{L}_{iu}(t)}}\big), \quad \forall i \in \mathcal{I}, \quad \forall u \in \mathcal{U}, \tag{3}$$

where $\bar{L}_{iu}(t)$, B, P_i, and σ^2 denote the average pathloss between UCH u and IoTD i at time slot t [5], the

channel bandwidth, the transmission power and the noise power, respectively.

B. Energy Harvesting Model

In our proposed ATG network, when the IoTDs have resource constraints (computation and energy) to compute tasks, then offloads to the edge servers; the EH transaction is processed before deciding whether to offload computation-intensive tasks. Furthermore, the UAVs may have an energy constraint in the long run to process various operations, and as such, the UAVs can obtain energy from the WPT station. The WPT station obtains energy from the traditional power grid and harvests renewable energy from the environment with EH equipment. The energy stored in the WPT station will charge the UAV network via wireless channels. We define the binary variable $e_{ix}(t) \in \{0,1\}$ for all IoTD $i \in \mathcal{I}$, where $e_{ix}(t) = 1$ denotes that IoTD i harvests energy from node x, and $e_{ix}(t) = 0$ denotes that IoTD i does not harvest energy from node x. We further define the EH vector $\mathbf{e}_i = (e_{i,1}, e_{i,2}, \ldots, e_{i,x}), \forall i \in \mathcal{I}$. We also assume that each IoTD can harvest energy from at most one node. The IoTDs may contain RF energy harvester which receives the power from transmitters. Let ψ denote the time used by UCH for transmitting RF energy, then the energy harvested by IoTD i from UCH u at time slot t is expressed as:

$$\Theta_i(t) = \sum_{u \in U} \nu_i P_u \bar{\mathrm{L}}_{iu}(t)\psi, \quad \forall i \in \mathcal{I}, \tag{4}$$

where $\nu_i \in (0,1)$ denotes the power conversion efficiency of IoTD i and P_u denotes the transmission power of UCH u. The initial battery level of IoTD i at time slot t is denoted as $\rho(t)$, thus, the battery level at $t+1$ is given as:

$$\rho_i(t+1) = \min\{\max\{0, \rho_i(t) - (E_i^{iot}(t) + \Theta_i(t))\}, \rho_{\max}\}, \tag{5}$$

where $\rho_i(t)$ indicates battery level. The total energy consumption of IoTD i (i.e., $E_i^{iot}(t) = E_i(t) + E_{iu}(t)$) should not exceed the harvested energy, which is expressed as:

$$E_i^{iot}(t) \leq \Theta_i(t). \tag{6}$$

The service price of transmitted power charged by the energy provider is expressed as:

$$\Psi_{ix}(t) = \Theta_i(t)\mu^{eh}, \tag{7}$$

where μ^{eh} denotes the price per unit for EH service demands of IoTD, and the computational node that charged it.

C. Computation Model

The IoTD i can compute tasks locally using its own resources or offload computation-intensive tasks to be executed by the edge server. The edge servers handle task processing and transaction mining, as well as uploading the results. For a set of $\mathcal{I}$ IoTDs, let us define offloading strategy vector $\boldsymbol{\alpha}_i = (\alpha_{i1}, \ldots, \alpha_{ix}), \forall i \in \mathcal{I}$, in which $\alpha_{ix}(t) \in \{0,1\}$. When $\alpha_{ix}(t) = 1$, the computation-intensive task of IoTD i is offloaded to the UCH/BS; otherwise, it is computed locally, $\alpha_{ix}(t) = 0$. As mentioned above, the EH process is performed before offloading tasks or executed locally. Let $\beta_i = (\boldsymbol{\alpha_i}, \boldsymbol{e_i})$ denote the EH and offloading decision of IoTD i, where $\boldsymbol{\alpha}_i$ and $\boldsymbol{e}_i$ indicate offloading vector and EH vector of IoTD i, respectively. Let the set $\boldsymbol{\beta} = \{\beta_i\}_{i \in \mathcal{I}}$ be a decision profile of all IoTDs. In the decision profile $\boldsymbol{\beta}$, we denote $i_x^{\mathcal{X}}(\boldsymbol{\beta}) = \sum_{i \in \mathcal{I}} \boldsymbol{\alpha}_{ix} | \boldsymbol{e}_{ix}$ as the number of IoTDs that harvest energy or offload tasks from/to edge node x.

1) Local Computation: In the local computational model, the task of IoTD i is executed locally at time slot t. The local execution delay is expressed as:

$$T_i(t) = \frac{F_i}{f_i}, \tag{8}$$

where f_i denotes the computation capacity of IoTD i. The local execution delay of the accomplished task $T_i(t)$ must be less than Γ_i. The local energy consumption of task $\Delta_i(t)$ is expressed as:

$$E_i(t) = \kappa_i (f_i)^{\upsilon} T_i(t), \tag{9}$$

where $\kappa_i \geq 0$ denotes CPU switch capacitance of IoTD i and υ denotes a constant value. In this paper, we set $\upsilon = 3$ and $\kappa_i = 10^{-27}$ [4]. The local computing cost of IoTD i for the completed task Δ_i is expressed as follows:

$$\mathcal{Z}_i^{loc} = \delta^t T_i(t) + \delta^e E_i(t), \tag{10}$$

where δ^t and δ^e denotes the weighted parameters of time and energy, respectively.

2) Offloading Computation Task: IoTDs can offload the computation tasks to the computational node x at time slot t. The transmission delay of IoTD i's task at time slot t is expressed as:

$$T_{iu}^{tr}(t) = \frac{\Lambda_i}{r_{iu}(t)}. \tag{11}$$

The execution time delay of offloaded task is given as:

$$T_{iu}^{exe}(t) = \frac{F_i \Lambda_i}{f_{iu}}, \tag{12}$$

where f_{iu} denotes the computation resource allocated to the IoTD i from UCH u. The IoTD i consumes $E_{iu}^{tr}(t)$ energy to offload the tasks to computational node x at time slot t, which is expressed as:

$$E_{iu}^{tr}(t) = P_i T_{iu}^{tr}(t), \tag{13}$$

where P_i denotes the transmission power of IoTD i. The energy consumed by UCH u during the execution of an offloaded task at time slot t is given as:

$$E_{iu}^{exe}(t) = \kappa_u (f_{iu}^{\upsilon}), \tag{14}$$

where $\kappa_u = 10^{-28}$ denotes CPU switch capacitance of UCH u and $\upsilon = 3$ denotes a constant value. The total computation costs in terms of total energy consumption and the total time delay is expressed as:

$$\hat{\mathcal{Z}}_{ix}(\boldsymbol{\beta}) = \delta^t T_{ix}(t) + \delta^e E_{ix}(t), \quad \forall i \in \mathcal{I}, \tag{15}$$

where $T_{ix}(t) = T_{ix}^{tr}(t) + T_{ix}^{exe}(t)$, $E_{ix}(t) = E_{ix}^{tr}(t) + E_{ix}^{exe}(t)$ indicate total time delay and total energy consumption

including cost of task offloading and execution, respectively. Then, the total computation costs of IoTD i in terms of energy consumption and time delay is calculated as: $\mathcal{Z}_i(\boldsymbol{\beta}) = \mathcal{Z}_i^{loc} + \hat{\mathcal{Z}}_{ix}(\boldsymbol{\beta})$. The product of service demands D_i and service price per unit of demand ω^f can determine IoTD i's service cost $D_i(t)\omega^f$ and is charged by an RP. Therefore, if IoTD i offloads to node x by service price ω^f, then the service cost that is charged by the RP is expressed as:

$$\bar{\Psi}_{ix}(t) = D_i(t)\omega^f. \tag{16}$$

The sum of energy and computation offloading costs of IoTD i ($\lambda_{ix}(\boldsymbol{\beta})$) charged by RP x, which is expressed as:

$$\lambda_{ix}(\boldsymbol{\beta}) = \Psi_{ix} + \bar{\Psi}_{ix}, \quad \forall i \in \mathcal{I}. \tag{17}$$

When IoTD i decides to harvest energy from node x (UCH u) and offloads its computation-intensive task to edge computing server x, then the corresponding energy cost, time cost, and service cost for harvesting $\Theta_i(t)$ energy from node x (UCH u) and offloading task $\Delta_i(t)$ to edge computing server x has a total cost as:

$$\mathcal{Z}_{ix}(\boldsymbol{\beta}) = \lambda_{ix}(\boldsymbol{\beta}) + \hat{\mathcal{Z}}_{ix}(\boldsymbol{\beta}), \quad \forall i \in \mathcal{I}, \quad x \in \mathcal{X}. \tag{18}$$

The total cost of IoTD i with decision profile including the recharged price is calculated as:

$$\bar{\mathcal{Z}}_i(\boldsymbol{\beta}) = \mathcal{Z}_i^{loc} + \mathcal{Z}_{ix}(\boldsymbol{\beta}). \tag{19}$$

We exclude the cost of returning the result from the edge computing server to IoTD i due to the size of the application result since it is much smaller than the input data size. Therefore, the system cost is calculated as:

$$\mathcal{Z}(\boldsymbol{\beta}) = \sum_{i \in \mathcal{I}} \mathcal{Z}_i(\boldsymbol{\beta}), \quad \forall i \in \mathcal{I}. \tag{20}$$

IV. Problem Formulation

In this section, we introduce the consortium blockchain in computation offloading with EH in ATG network, optimization objective function and Stackelberg game model.

A. Blockchain for Computation Offloading With EH

To ensure secure transactions in the dynamic computation offloading with the EH system, we use a consortium blockchain (Hyperledger fabric platform) integrated with the DRL model and smart contact. The consortium blockchain consensus process is completed on preselected nodes with moderate cost, thus making it more relevant and feasible in energy-constrained network scenarios such as UAV and IoT networks. Also, to achieve efficient consensus process, the edge computing nodes are utilized for block mining which minimizes the computation traffic during the creation of a block. The computation offloading and EH transaction records of all RPs and RRs are uploaded to the chain after being validated by the authorized consensus nodes.

Fig. 2 shows the process of consortium blockchain-based computation offloading with EH in a multi-UAV-assisted IoT network. The main entities of the blockchain network are the certificate authority (CA), IoTDs, edge computing servers, and RP. The CA initializes the computation offloading and EH system, generates public parameters, and manages the operating entities. In such a scenario, all the blockchain entities must first obtain a unique identifier from the CA to participate in the blockchain legally. The CA is in charge of entity validation, identity authorization, and certificate issuance. Each node must register and create an account in the EH and computation offloading blockchain before running the blockchain. IoTDs are computation and energy-constrained entities that demand computation offloading and EH services from UCH or BS [4]. These devices perform computation offloading to MEC servers which can provide computing and storage services.

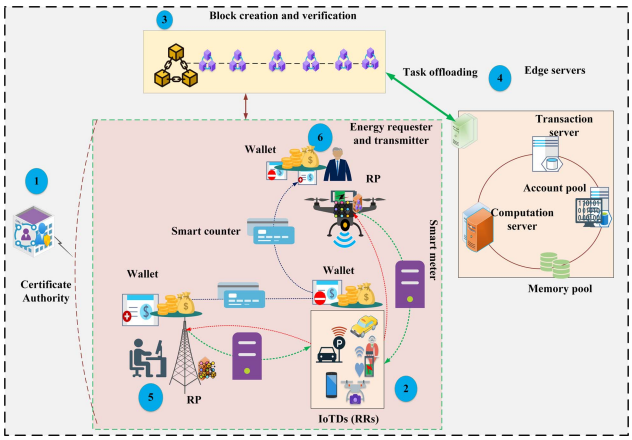


Fig. 2. Consortium blockchain process.

The operation of our proposed blockchain-based computation offloading with the EH framework is presented in five stages in **Algorithm 1**:

(1) System initialization: Once the entities involved in the system have been identified, the blockchain system must be initialized using a cryptography technique to ensure the authenticity and integrity of digital messages. In the proposed framework, each node registers with the CA to become a legitimate entity and get a unique identity (ID) for joining the blockchain system and then obtains the public/private key via elliptic asymmetric cryptography [37] and a certificate using the curve digital signature algorithm [38].

(2) Transaction process: Transaction recording is a critical process in the blockchain system. IoTDs initiate a transaction and add it to the blockchain to request resources. In our system, the IoTD sends offloading and EH requests to nearby MEC servers, which then stores them in the transaction pool. After which the servers then broadcast their demand to all service providers. The UCHs/BSs will then respond to the edge server, providing their inventory and unit price. The edge server matches trading pairs based on IoTD demand and UCH/BS stock following the signed smart contract and control strategy. Then IoTDs and UCHs/BSs can proceed with their transactions.

(3) Consensus nodes selection: In our scenario, we use a hybrid of proof of reputation (PoR) and Practical Byzantine fault-tolerant (PBFT) consensus mechanism [34]. Since we use a consortium blockchain, only selected edge nodes can participate in the consensus process to confirm the validity

Algorithm 1 Blockchain-Enabled Task Offloading and EH

Initialize: Number of RPs and RRs
Registration and certification of RPs, RRs
for time T **do**
  for all x, i **do**
    Verify the certificates
    if the certificates of RPs, RRs are verified
      Set-up Stackelberg game as Eqns. (25) and (26)
      Run SC for computation offloading with EH and create a block
    else
      Terminate SC
    end if
  end for
  for N edge nodes **do** ***/Block creation, consensus/**
    Leader x broadcast block blk to all validator nodes
    Each node receive blk and check its trust value
    for All edge nodes **do**
      if Block blk is verified and $\tau_{ix} \geq 0.5$ **then**
        Set the block blk = true
      else
        Set the block blk =false
      end if
      Send the audit results to the leader for analysis
    end for
    Accept block blk and add to chain or discard if the node is not trusted and the verification is failed
  end for
end for

of transactions and create new blocks. The consensus nodes can generate, validate, and store new blocks in the digital ledger with minimal resources using distributed consensus. This consensus mechanism is suitable for systems containing resource-constrained devices such as IoT devices, UAVs, etc. However, not all edge nodes in the ATG system can be trusted; some malicious edge nodes may discard transaction records during the mining process. To ensure that communication between entities in the system is secure, the communicating parties must have a high level of trust. When there is a lack of trust between the nodes, the transaction is discarded. Let $\tau_{ix} \in [0, 1]$ represent the trust value between blockchain entities. The trust threshold is set to 0.5, according to [29] and [39]. If the trust value τ_{ix} is greater than 0.5, the node is trustworthy; otherwise, it is not trustworthy.

(4) Block generation: When blockchain entities send requests to the blockchain, the nearest edge node collects and stores them in the transaction pool. The leader (the node with the highest reputation) then collects the transaction records, builds a block, and broadcasts it to other consensus nodes. Then the consensus process continues.

(5) Consensus process: The block generated by the leader is legally added to the chain only after other consensus nodes confirm its validity via consensus. Once the leader node has been identified, it packs transactions into a new block, signs, and broadcasts to other consensus nodes for verification and validation. The consensus nodes then audit and cross-check the validity of the newly received block by comparing their audit results to those of other consensus nodes and send a consent or commit message to the leader node. The leader collects and analyzes responses from all consensus nodes. If the leader receives a majority consent message from the consensus nodes, it signs the block, appends the consensus nodes' audit results, and broadcasts the block to all nodes in the system. Finally, the new block is chronologically linked to the main blockchain, and the node's local view of the blockchain state is updated. Once the blockchain consensus process is complete, the transaction details are recorded using a distributed ledger. Each edge computing server consists of a transaction server, an account pool, a memory pool, and a computation server. The transaction server collects the real-time energy and offloading requests from IoTDs and the price announcements from RPs. The transaction digital coins work as IoTDs' digital assets used to purchase resources (energy and computation resources) from RPs. Both RR and RP have a virtual wallet to manage individual resource coins. The account pool at the edge computing nodes records and stores resource coins in the RRs and RPs based on personal wallet. The memory pool stores all transactions executed by computing servers, such as blockchain creation and validation. The computation server provides computing resources for the operation of block creation and confirmation.

The UCH and BS transfer energy and compute offloaded tasks. Then the IoTDs transfer coins to the RP (BS/UCH) wallets. The operators provide the resource and obtain a reward/incentive. The IoTDs (buyer) pays a digital coin to the UCH or BS (seller) based on the record of smart meters. Smart contracts also facilitate smart data sharing and provide automation and transparency in a distributed manner. In our system, all blockchain entities are allowed to access smart contracts and their transaction.

B. Cost Objective Function

We formulate the optimization problem in our proposed framework to minimize the computation costs including energy consumption, time delay, and resource allocation depending on the decision profile $\boldsymbol{\beta}$ and resource of RPs. Therefore, the optimization problem is expressed as follows:

$$\boldsymbol{P_1} \quad \min_{\boldsymbol{\beta},\boldsymbol{F}} \mathcal{Z}(\boldsymbol{\beta}) \tag{21a}$$

s.t.

$$C1: \alpha_{ix}(t), e_{ix}(t), \xi_{ix}(t) \in \{0, 1\}, \ \forall i \in \mathcal{I}, x \in \mathcal{X}, \tag{21b}$$

$$C2: \sum_{x \in \mathcal{X}} \alpha_{ix}(t) \leq 1, \forall i \in \mathcal{I}, \tag{21c}$$

$$C3: \sum_{x \in \mathcal{X}} e_{ix}(t) \leq 1, \quad \forall i \in \mathcal{I}, \tag{21d}$$

$$C4: \sum_{x \in \mathcal{X}} \xi_{ix}(t) \leq 1, \quad \forall i \in \mathcal{I}, \tag{21e}$$

$$C5: \sum_{x \in \mathcal{X}} f_{ix}(t) \leq F_x, \quad \forall i \in \mathcal{I}, \tag{21f}$$

$$C6: f_{ix} \geq 0, \quad \forall i \in \mathcal{I}, \tag{21g}$$

$$C7: E_i^{iot} < \Theta_i(t) + \rho(t), \quad \forall i \in \mathcal{I}. \tag{21h}$$

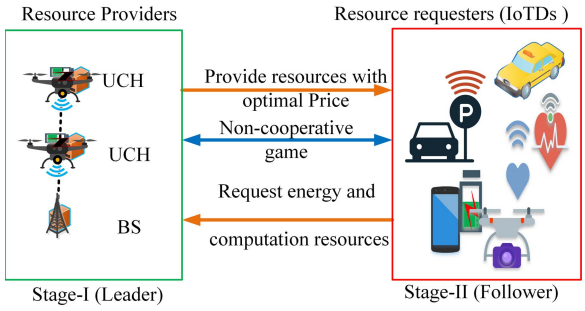


Fig. 3. Stackelberg model for computation offloading with EH.

Here, constraint $C1$ indicates the binary offloading decision strategy, binary EH decision strategy of IoTD i, and connection status between RP x and IoTD i, respectively. Constraints $C2, C3$ and $C4$ indicate that computation task is offloaded to at most one RP, energy is harvested from at most one RP, and IoTD i is connected with one computational node within time slot t, respectively. Constraints $C5$ and $C6$ indicate that the assigned/sold resource does not exceed the RP's total computing capacity. Constraint $C7$ indicates the energy constraint of IoTD i which shows the battery energy should not run out for each computing IoTDs. Currently, the growth of IoTDs is exponential and with dynamic network scalability, then the complexity of problem $\boldsymbol{P_1}$ is high. It is a mixed-integer and non-convex optimization problem. It involves the binary offloading decision vector $\boldsymbol{\alpha}$, EH decision vector $\boldsymbol{e}$, binary connection vector $\boldsymbol{\xi}$, continuous indicator f, and the non-convex objective function, which is NP-hard and challenging to solve directly and obtain an optimal solution. Thus, existing works try to solve such problems by decomposing the optimization problem into different sub-optimization problems [4]. However, solving a complex problem in a multi-UAV assisted IoT network similar to [4], [30], [31] results in curse of dimensionality and raises security concerns.

C. Stackelberg Game Formulation and Analysis

In this paper, we formulate a multi-leader multi-follower Stackelberg game. Fig. 3 depicts the interaction between RPs (leaders) and RRs (followers) of the game. The RPs with computing servers can provide resources to RRs at a low cost. To maximize their utility, the RP first announce the selling price of their resources to the RRs. The RRs determine their resource demands for purchase and pay the RPs the agreed-upon prices. The proposed game in a multi-UAV-assisted IoT network aims to obtain optimal offloading and EH policies with optimal price of resources.

In the first stage, the edge nodes act as RPs, set the EH and computation resources price first, and dynamically adjust their pricing policy. The RPs announce a set of price, where $\mathcal{P} = \{P_1, P_2, \ldots, P_X\}$. Therefore, the expected reward of an RP can be calculated as:

$$R_x = \beta \sum_{i \in I} (\mu^{eh} \Theta_i + \omega^f D_i) - \sum_{i \in I} (\Theta_i D_i E_x^{exe}), \quad (22)$$

where $\mu^{eh}, \omega^f \in \mathcal{P}$ denotes the different resource price, Θ_i represents energy, D_i is the computation resource that IoTD i requests from edge node $x \in \mathcal{X}$, and E_x^{exe} represents energy consumption of edge node. The RPs first set resource prices according to the RRs demand requirements and sells the resources to RRs to increase their reward. Therefore, the optimization problem of RPs at stage-I is expressed as:

$$\boldsymbol{P_2} \quad \max_{P_x} R_{xi}(\beta, \Theta_i, D_i, P_x) \quad (23a)$$

$$\text{s.t.} \quad C_1 \quad P_x \geq 0, \quad (23b)$$

$$C_2 \quad \beta \sum_{i \in I} (\mu^{eh} \Theta_i + \omega^f D_i) \geq \sum_{i \in I} (\Theta_i D_i E_x^{exe}). \quad (23c)$$

According to the optimization problems ($\boldsymbol{P_2}$), the RPs would like to maximize the utilities. Both the RPs and RRs seek to effectively adjust their strategies to maximize the reward and minimize the cost, respectively.

In the second stage, the RRs determine their EH strategies, offloading strategies, and resource (energy and computation) demand based on optimal price and QoS level estimated by the RPs. The action strategies consist of decision strategies $\boldsymbol{\beta}$ and set of price $\mathcal{P}$. In addition, the RRs have a set of decision strategies β, set of energy demand Θ, and set of computation demand $\mathcal{D}$, where $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_I\}$, $\Theta = \{\Theta_1, \ldots, \Theta_I\}$, and $\mathcal{D} = \{D_1, \ldots, D_I\}$.

Each RR determines its energy resource demand $\Theta_i \in [\underline{\Theta}, \overline{\Theta}]$, computation resource demand $D_i \in [\underline{D}, \overline{D}]$, and price offered by the RPs $P_x \in [\underline{P}, \overline{P}]$, where $\underline{\Theta}$ and $\overline{\Theta}$ are the lower and upper bound of energy requirement, $\underline{D}$ and $\overline{D}$ are the lower and upper bound of computation requirement, $\underline{P}$ and $\overline{P}$ are the lower and upper bound of price. Besides, the RRs also need to minimize the service costs as described in Eq. (19). The optimization problem of the RRs is expressed as:

$$\boldsymbol{P_3} \quad \min_{\boldsymbol{\beta}, \boldsymbol{F}} \mathcal{Z}_i(\boldsymbol{\beta}) \quad (24a)$$

$$\text{s.t.} \quad \alpha_{ix}(t), e_{ix}(t), \xi_{ix}(t) \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \quad x \in \mathcal{X}. \quad (24b)$$

Then RRs are interested in ensuring QoS and minimizing costs. The main target of the Stackelberg game is to find the Stackelberg Equilibrium (SE).

Definition 1: Let $\beta_i^*, \Theta_i^*, D_i^*$, and p_x^* denote the optimal decision, optimal energy demand, optimal computation demand for each RR, and the optimal price for each RP, respectively. The Stackelberg game model can be formulated as $\mathbf{S} = (p^*, \beta^*, \Theta^*, D^*)$, where the SE satisfies the following expressions:

$$\begin{aligned} &\text{Follower:} \quad \mathcal{Z}(p^*, \beta^*, \Theta^*, D^*) \geq \mathcal{Z}(p^*, \beta, \Theta^*, D^*), \\ &\text{Leader:} \quad R_x(p^*, \beta^*, \Theta^*, D^*) \geq R_x(p, \beta^*, \Theta^*, D^*). \end{aligned} \quad (25)$$

Theorem 1: For the IoTD (RR), its cost function satisfies Eq. (20). To verify the uniqueness and existence of the SE in our Stackelberg game, we take the second-order derivative of the cost function of Equ. (20) with respect to β, Θ_i, D_i and reward function of Equ. (22) with respect to p, which is

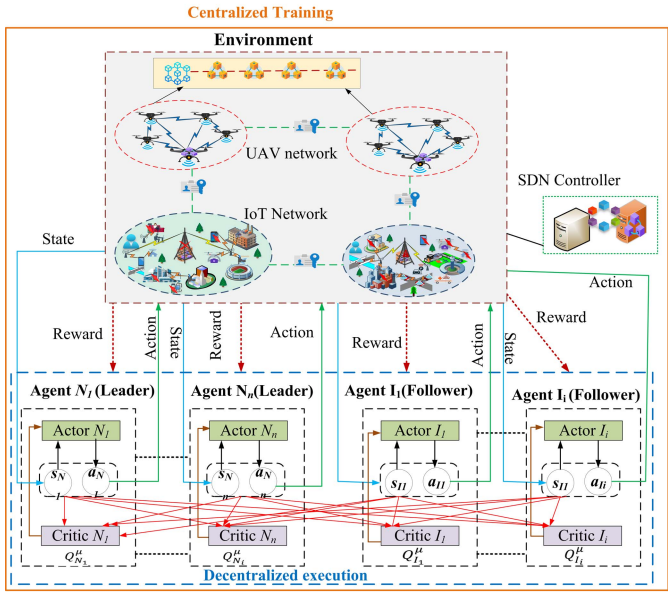


Fig. 4. MADDPG framework in blockchain-based multi-UAV-assisted IoT network.

expressed as:

$$\frac{\partial^2 \mathcal{Z}(\beta)}{\partial(\Theta_i D_i)^2} = -2\frac{Z_i(\beta)}{(\sum_{i\in I}\Theta_i D_i)^3}) \leq 0. \tag{26a}$$

$$\frac{\partial^2 R_x}{\partial(p)^2} = -2\frac{(\Theta_i D_i E_x^{exe})}{p^2}\frac{x-1}{x}R_x \leq 0. \tag{26b}$$

Hence, $\mathcal{Z}$ and R_x are strict concave, and SE exists in this Stackelberg game.

V. MADRL Based Solution

A. MADRL Based Computation Offloading for IoTD With EH

In this paper, we consider computation offloading with EH in blockchain-based multi-UAV-assisted IoT network, where multiple agents are involved. In such a dynamic environment, solving optimization problems using traditional single agent approaches is challenging [5]. Therefore, we proposed a MADRL algorithm to solve multiple problems in a multi-agent system.

To solve the Stackelberg game based problem $(\boldsymbol{P_2}, \boldsymbol{P_3})$, we transform the problem into a MDP game. In a multi-agent environment, the problems are formulated in Markov Games [40]. We introduce a MADRL-based computation offloading with an EH solution to minimize the computation costs of IoTDs and maximize the reward of RPs. The RRs and RPs act as agents and compete each other to achieve their goal. According to Fig. 4, in each stage agents compete with each other, obtain experience from the others by observing the state of the environment. We take the sum of the reward of RPs and RRs as a system reward. To implement a MADRL method, we define state space, action space, and reward at each time slot t as follows:

- ***State space:*** The learning agents learn to get experience and update decision policy through observing states of the environment. Based on the optimization objectives, the agents make their decisions using states in the environment. The state space is denoted as $s(t) = (\xi_{ix}, \tau_{ix}, e_i, f_i, e_x, f_x, p_x^e, p_x^f)$, where ξ_{ix} and τ_{ix} are the connection status and the trust value between IoTD i and edge node x, respectively. Furthermore, e_x, f_x, p_x^e, and p_x^f represent the available energy resource, computing resource of edge node x, service price of transmitted energy and computing task, respectively.
- ***Action space:*** The action space includes a vector of decision (offloading and EH decisions) $\boldsymbol{B} = [\beta_1^{e,f}, \beta_2^{e,f}, \ldots, \beta_I^{e,f}]$, vector of transmitted power allocation from RP $\boldsymbol{E} = [e_1, e_2, \ldots, e_I]$, and computation resource allocation $\boldsymbol{F} = [f_1, f_2, \ldots, f_I]$. The overall action vector is denoted by $a(t) = [\beta_1^{e,f}, e_1, f_1, \ldots, \beta_I^{e,f}, e_I, f_I]$.
- ***Reward function:*** The objective of the reward is to optimize the offloading, EH decisions and minimize the computation costs, which include time delay, energy consumption and price per service, to ensure QoS and prolong battery life of IoTDs. The reward function of RR is expressed as:

$$r_{ix}(t) = -(\delta^t T_{ix}(t) + \delta^e E_{ix}(t)). \tag{27}$$

The reward function of RPs is expressed as:

$$r_x(t) = R_x(t). \tag{28}$$

Then, the overall reward of the system is expressed as:

$$r(t) = \sum_{i\in\mathcal{I}}\sum_{x\in\mathcal{X}}\Big(r_{ix}(t) + r_x(t)\Big). \tag{29}$$

B. Multi-Agent Deep Deterministic Policy Gradient Algorithm

Fig. 4 shows the MADRL framework of blockchain-based multi-UAV-assisted IoT network. The MADDPG algorithm is one of the MADRL algorithms [41], which is a modified actor-critic-network with DQN method that can handle the dynamic environment and it is useful for the cooperative policy learning of multi-agents with continuous action space. This algorithm also supports centralized training of the action-value function or critic-network with decentralized execution. The critic function also uses other agents' action policies, and agents to make a decision independently using their strategies and observations. Policy parameters are updated by an individual.

To be specific, a game with a set of K agents include RPs N_n and RRs I_i with their own observations, actions and rewards. Agent k has a set of states $\mathcal{S} = \{S_1, S_2, \ldots, S_K\}$, a set of action $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_K\}$ and set of observations $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_K\}$. Each agent k can choose an action using stochastic policy $\pi_{(\theta_k)} : \mathcal{O}_k \times \mathcal{A}_k \rightarrow [0, 1]$. The next state is also obtained using state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_k \times, \ldots, \times \mathcal{A}_K \rightarrow S$. Each agent obtains the rewards based on current states and agent actions $r_k : \mathcal{S} \times \mathcal{A}_k \rightarrow \mathcal{R}$, and also receives their own observation $\mathcal{O}_k : \mathcal{S} \rightarrow \mathcal{O}_k$ privately. The main objective of each agent k is to maximize its expected sum of discounted rewards, $\mathbb{E}\{\sum_{j=0}^{\infty}\gamma^j r_k, t+j\}$, where $r_k, t+j$ is the reward received at j episodes into the future

by agent k. As mentioned above, the main idea of MADDPG is to learn a centralized critic function (action-value function) $Q_k^{\pi}(\mathcal{K}, a_1, \ldots, a_K)$ for k-th agent, where $\{a_1, \ldots, a_K\} \in \mathcal{A}$ are the representation of all agents' actions and each agent has the global training weight information. The set of policies of all agents expressed as $\pi = \{\pi_1, \ldots, \pi_K\}$, with policy parameter $\theta = \{\theta_1, \ldots, \theta_K\}$. In the MADDPG algorithm, the actor and critic-network will be updated after each training episode. The actor-network can be updated using the gradient method calculated as follows:

$$\nabla_{\theta_k}(\theta_k) = \mathbb{E}_{\mathcal{K},a\sim\mathcal{D}}[\nabla_{a_k}\theta_k^{\mu}(\mathcal{K}, a_1, \ldots, a_K) \nabla_{\theta_k}\mu\theta_k(\mathcal{O}_k)|_{ak} = \mu\theta_k(\mathcal{O}_k)], \quad (30)$$

where $\mathcal{D}$ is the experience replay buffer which stores all agent experiences including $(\mathcal{K}, \mathcal{K}', a_1, \ldots, a_K, r_1, \ldots, r_K)$. The critic function θ_k^{μ} is updated as the following expression:

$$\mathcal{L}(\theta_k) = \mathbb{E}_{\mathcal{K},\mathcal{K}',a_1,\ldots,a_K,r_1,\ldots,r_K}[(\theta_k^{\mu}(\mathcal{K}, a_1, \ldots, a_K) - y)^2], \quad (31)$$

where $y = r_k + \gamma\theta_k^{\mu'}(\mathcal{K}', a_1', \ldots, a_K')|_{a_j'=\mu_j'(\mathcal{O}_j)}$. The set of target policies with delayed parameters θ_k' is denoted as $\mu' = \{\mu\theta_1', \ldots, \mu\theta_K'\}$.

Algorithm 2 describes the proposed MADDPG algorithm for computation offloading with EH in blockchain-based multi-UAV-assisted IoT network and it has two procedures: Collecting observed data and training procedures. The replay buffer, actor-network, and critic-network parameters with weights are initialized (lines 1-2), and the number of episodes and training time steps are defined. Agents collect observed data (lines 3-24). The agents execute the action, obtain the reward, and generate a new state (line 12-15). The experience is stored in the experience replay buffer. In the training step (line 24-31) use a policy training is chosen, which uses a batch sampling from replay buffer. Then the actor-network and the critic-network updated based on a randomly selected sample.

Algorithm 2 MADDPG Algorithm for Computation Offloading With EH

1: **Initialize** the local replay buffer $\mathcal{D}_k^{loc}$ at RP and RR, and global replay buffer $\mathcal{D}$ at the controller layer
2: **Initialize** the parameters of actor and critic-network with random weights θ
3: **for** episode = $1, 2, \ldots, 3000$ **do**
4: Initialize state space $S = \{s_1, s_2, \ldots, s_K\}$
5: Via blockchain, execute **Algorithm 1**
6: **for** $t = 1, 2, \ldots, 200$ **do**
7: Estimate the connection status $\xi_{ix}(t)$,
8: Estimate the resources demands of RR (e_i, f_i)
9: Estimate the available resource of RP (e_x, f_x)
10: Estimate the price per service (p_x^e, p_x^f)
11: All agents receive initial state $s(t) = (\xi_{ix}, \tau_{ix}, e_i, f_i, e_x, f_x, p_x^e, p_x^f)$
12: Each agent k selects a random action $a_k = \pi_{\theta_k}(s_k)$ based on the probability ε
13: All agents execute action $a(t) = \{a_1(t), a_2(t), \ldots, a_K(t)\}$ including $a = [\beta_1^{e,f}, e_1, f_1, \ldots, \beta_I^{e,f}, e_I, f_I]$
14: Observe rewards $r(t) = \{r_1(t), r_2(t), \ldots, r_K(t)\}$
15: The new state $s_k(t+1) \sim s_k'$
16: Save the tuples $\{s_k(t), a_k(t), r_k(t), s_k'\}$ in $\mathcal{D}_k^{loc}$
17: Agent k uploads the tuple value from $\mathcal{D}_k^{loc}$ to SDN controller $\mathcal{D}$
18: Merge $\mathcal{D}_k^{loc}$ into $\mathcal{D}$ at SDN controller
19: Download $\mathcal{D}$ from SDN controller
20: $s_K \leftarrow s_k'$
21: **for** agent $k = 1$ to K **do**
22: Sample a random mini-batch of H samples tuples (s^j, a^j, r^j, s'^j) from $\mathcal{D}$
23: Set $y^j = r_k^j + \gamma Q_k^{\pi'}\left(S'^j, a_1', \ldots, a_K'\right)|_{a_k'=\pi_k'(s_k^j)}$
24: Update the critic-network as (31)
25: Update actor-network as (30)
26: **end for**
27: Update target network parameters for each agent k
28: $\theta_k' \leftarrow \tau\theta_k + (1-\tau)\theta_k'$
29: **end for**
30: **end for**

C. Algorithm Analysis

Computational Complexity Analysis: The proposed MADDPG algorithm for computation offloading and EH has centralized training and decentralized execution; the complexity analysis is discussed as follows: Let us define the number of neurons in the g-th layer of the actor-network as M_g^a, the actor-network is fully connected. Then, the computational complexity of g-th layer can be expressed as $\mathcal{O}\left(M_{g-1}^a M_g^a + M_g^a M_{g+1}^a\right)$. Therefore, for G number of layers in the actor-network, the computational complexity of the actor-network is $\mathcal{O}\Big(\sum_{g=2}^{G-1}\left(M_{g-1}^a M_g^a + M_g^a M_{g+1}^a\right)\Big)$. For the critic-network with L layers, the number of neurons in the l-th layer of the critic-network is M_l^c, and the computational complexity of l-th layer critic-network is calculated as $\mathcal{O}\left(M_{l-1}^c M_l^c + M_l^c M_{l+1}^c\right)$. The overall computational complexity of the critic-network is expressed as $\mathcal{O}\Big(\sum_{l=2}^{L-1}\left(M_{l-1}^c M_l^c + M_l^c M_{l+1}^c\right)\Big)$. Therefore, the complexity of the proposed MADDPG algorithm for training is $\mathcal{O}\Big(K * T * E * H * \big(\sum_{g=2}^{G-1}\left(M_{g-1}^a M_g^a + M_g^a M_{g+1}^a\right) + \sum_{l=2}^{L-1}\left(M_{l-1}^c M_l^c + M_l^c M_{l+1}^c\right)\big)\Big)$, where K, T, E, and H denote number of agents, maximum training steps of each episode, number of episodes and mini-batch sampling size, respectively. The execution complexity of each agent is $\mathcal{O}\Big(K * E * \big(\sum_{g=2}^{G-1}\left(M_{g-1}^a M_g^a + M_g^a M_{g+1}^a\right)\big)\Big)$. In the proposed MADDPG algorithm, we analyze the convergence based on the aforementioned analysis when the interaction of RPs agents provides the best response to RRs agents based on current RR policies. This can impact the action selection of RR agents in subsequent steps. However, the actions taken by each RR agent have an impact on the policies of other agents. All agents alternatively support policy changes by agreeing to accept each other's action choices.

VI. Performance Evaluation

A. Experiment Setup

In this section, we evaluate the simulation results of our proposed algorithm through different parameter settings. The deployment and parameters configuration of multi-UAV networks and IoT networks commonly depend on the works in [5] and [4]. We implement the simulations using Python 3.7 environment with Tensorflow 2.0 on a computer with a Core i7 CPU running on a processor speed of 2.4GHz and 16GB RAM. We considered 12 edge nodes (6-UCH, 6-BS), a computing server as RP, one centralized SDN controller with MEC server, and 60 IoTDs in different cells. The UCH's network coverage radius (r_u) is set at 600m, an altitude $h_u = 100$m and the BS coverage is set at 500m. We consider small cells with BS and deployed UAV clusters to assist IoT networks in three cells. The computation capability of IoTD, the flying MEC server (on UCH), and local MEC server (BS) are set at $f_i = 1GHz, f_u = 45GHz$, and $f_n = 60GHz$, respectively. The probabilistic model parameters are as in [5].

In the proposed multi-agent algorithm, we set 3000 episodes with 200 steps in the training stage and use a fully-connected neural network with critic-network and actor-network. For each agent, we use two hidden layers in both the actor and critic neural networks, with 256 in the first hidden layer and 128 in the second layer. The mini-batch size was set to 256, and the replay buffer size was set to 10^5. We set an agent action selection probability $\epsilon \in (0, 1)$, weighted parameter of energy $\delta^e \in [0, 1]$, and weighted parameter of time delay $\delta^t = \delta^e - 1$. The transactions block size is set at 1.0 MB, with a 0.72 seconds block propagation delay. The size of each transaction in the block ranges from 0 to 0.000573 MB. In our analysis, we consider RPs from 2 to 12, the number of resource requestors from 10 to 60, and the number of validators (consensus nodes) from 10 to 70.

We consider the following evaluation metrics to evaluate the performance of the proposed algorithm.

(1) Average cost: This is the mean ratio of system cost to the number of agents' (IoTD). It is used as a matrix to evaluate the performance of the proposed algorithm. The smaller the average cost, the better performance of the algorithm.

(2) Computation time delay and energy consumption cost: These play a vital role in network performance and improve the QoS. Both metrics indicate from transmission up to the completion of offloading tasks.

(3) Optimal price and optimal transmission power: This indicates the availability of RP with optimal price per resource, and the IoTD can easily access the energy from RPs. The smaller optimal price and transmission power indicate the better completion of the algorithm.

(4) Other matrices like RP rewards/utilities, offloading rate of IoT, and scalability are also used to analyze the performance of the proposed algorithm.

To evaluate the proposed framework, we use DDPG [4], A3C [4], and Dueling-DQN [5] algorithms as a baseline. Other simulation parameters are summarized in Table III.

TABLE III
Simulation Parameters

Parameters	Value
The input task size of IoTD i	[150-15000]KB
The transaction size of IoTDs i [42]	300-1500
The number of CPU cycles required by task Δ_i	[150-2000] Mcycle
The maximum processing time delay Γ_i	0.1-2 Sec
Transmission power of IoTD P_i	0.1 W
The energy conversion efficiency ν_i	[0, 1]
Transmission power of node x	[1.5,3] W
Channel bandwidth B	40MHz
The unit price of energy μ^{eh}	[0,1] token/J
The unit price of computing resource ω^f	[0,1] token/Gcycle
The trust value τ_{ix}	[0,1]
Learning rate of actor-network, critic-network	$1e^{-4}$, $3e^{-4}$
Discount factor γ	0.99
Soft update factor τ	$1e^{-3}$

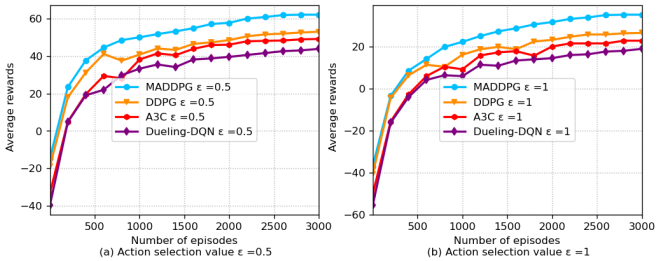


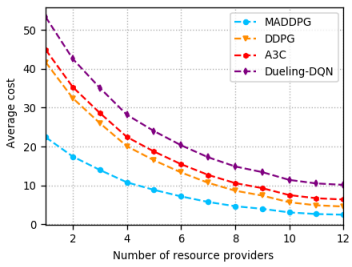
Fig. 5. Average reward with training episodes.

B. Convergence Analysis

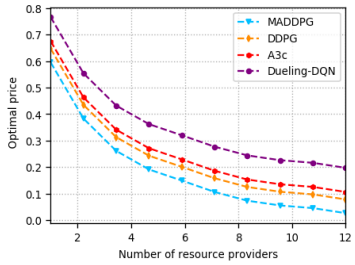
Fig. 5 shows the convergence on average rewards of different DRL algorithms with varying action selection probabilities. As shown in Fig. 5 (a), with action selection probabilities $\epsilon = 0.5$, IoTDs can access both ABS and BS, whereas with $\epsilon = 1$, IoTDs can only access BS (Fig. 5 (b)). The proposed MADDPG algorithm achieves better performance than the other three baseline algorithms. The proposed MADDPG algorithm converges before 500 episodes, and the other three also converge after about 600 episodes. This implies that the RPs and the RRs achieve their interests in maximizing utilities/rewards and minimizing computation costs with optimal offloading decision and EH policies, and pricing. We observed that the average rewards increase when the number of episodes and the number of agents are increasing. The proposed MADDPG algorithm increases the overall reward value by 37.68%, 43.36%, and 66.272% compared with DDPG, A3C, and Dueling-DQN, respectively. Therefore, both RPs and RRs achieved better rewards and minimized computation costs in a dynamic ATG network environment.

C. Performance Analysis

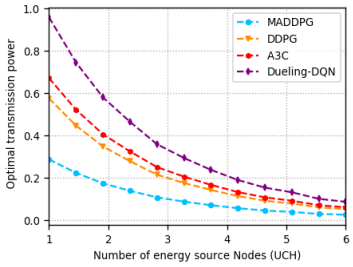
(1) Performance comparison with respect to the number of RPs: In Fig. 6, we first analyze the performance of our proposed algorithm as the number of RPs increases. Fig. 6 (a) shows the average long-term cost of all algorithms decreases as the RPs increase. We observed that the average cost of MADDPG algorithm reduces faster and outperforms the other three baseline algorithms. It implies that under the proposed MADDPG algorithm, the agents cooperate and minimize



(a) Average cost with number of RPs.

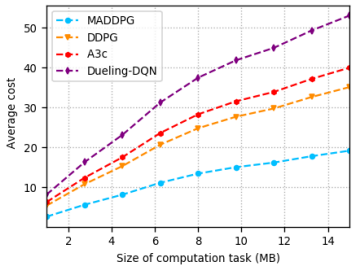


(b) Optimal price with number of RPs.

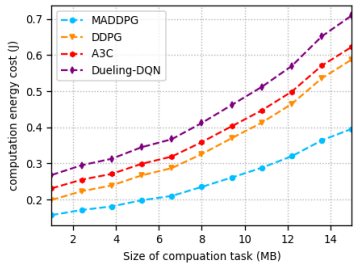


(c) Optimal transmission power.

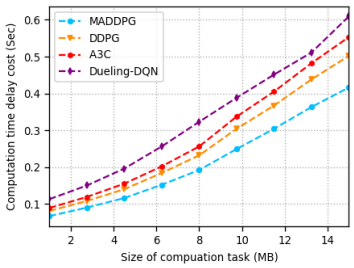
Fig. 6. Effects of RP on cost, price and transmission power.



(a) Average cost with number of data size.



(b) Energy consumption with data size.



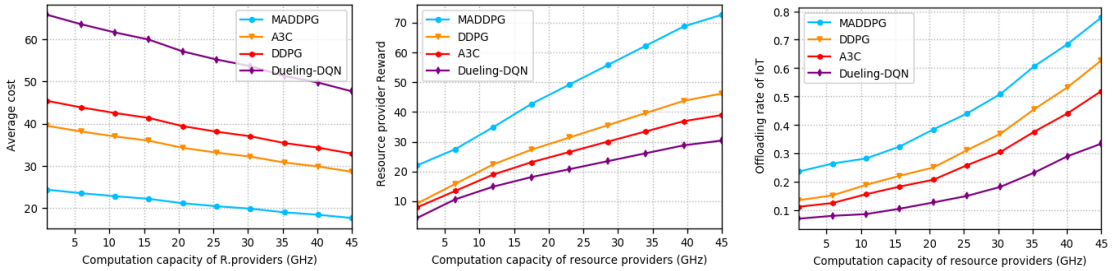
(c) Time delay cost with data size.

Fig. 7. Effects of data size on cost, energy and time.

resource consumption and price per service than other algorithms. DDPG and A3C algorithms are better than Dueling-DQN because they can support distributed learning more than Dueling-DQN. More specifically, MADDPG reduces the long-term average cost by 46.32%, 58.54%, and 67.98% compared with DDPG, A3C, and Dueling-DQN, respectively. Therefore, the proposed MADDPG algorithm in a multi-agent system with blockchain technology outperforms the baselines and optimize the decision of the agents. Fig. 6 (b) and 6 (c) show the optimal price per service and transmission power decrease when the number of RPs increases in the proposed framework. It implies that as the number of RPs in proposed MADDPG algorithm increases, the price per service decreases because nodes compete to sell their resources in a trading mechanism. Due to this, the IoTDs (agents at stage-II) will obtain a low price per service, and the transmission power also reduces. DDPG, A3C, and Dueling-DQN decrease the optimal price and power slowly. Therefore, The proposed MADDPG algorithm in both figures outperforms the three baseline algorithms.

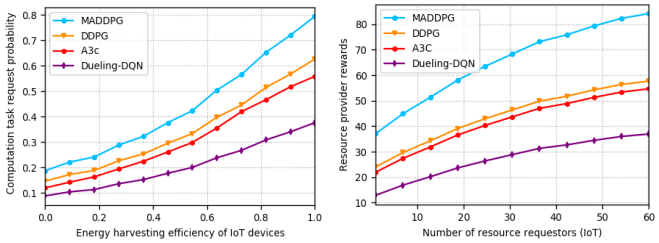
(2) Performance comparison based on offloading data size: In this simulation, we examine the average long-term cost, energy consumption, and time delay using different offloading data sizes ranging from 1 to 15MB to evaluate the impact of offloading data sizes on performance. Fig. 7 (a) shows the average long-term cost under different offloading data sizes. We observe that the average long-term cost increases gradually in all algorithms. This is because, more offloading data mining tasks need more transmission and execution power. It takes high computing time and IoTDs purchase more resources from RPs. When the offloading task size is 5MB, the average cost of the three baseline algorithms increases rapidly. However, the average long-term cost under the proposed MADDPG algorithm rises much slower than the three baseline algorithms. In this algorithm, the agents collaborate and share information, and due to this, the agent can minimize extra costs and price. The average long-term cost of the MADDPG algorithm decreases by 45.42%, 52.07%, and 63.88% compared with the DDPG, A3C, and Dueling-DQN algorithms, respectively. In Fig. 7 (b) and 7 (c), we analyze the computation energy consumption and time delay, including the transmission and execution of IoTD tasks. Fig. 7 (b) shows that total energy consumption increases with increasing offloading tasks in all algorithms, but the proposed MADDPG algorithm increases slower than the other three baseline algorithms. Fig. 7 (c) shows the computation time delay also increases in parallel with offloading data size increasing. However, the proposed MADDPG algorithm's time delay is lower than the other three baseline algorithms. Therefore, the proposed MADDPG algorithm with blockchain technology can minimize the computation costs in terms of energy consumption and time delay, simultaneously ensure security in a dynamic ATG network for computation offloading with EH algorithm in different situations.

(3) Performance based on RPs' computation capacity: We present the impact of RPs' computation capacity in average long-term cost, offloading rate of IoTDs, and maximizing rewards/utilities of RPs. Fig. 8 (a) shows that when the RP computation capacity increases, the average long-term cost decreases slowly in all algorithms. Even if the computation capacity increases, the IoTD charging prices per resource also increase. Thus, the average long-term cost decreases gradually. Specifically, the average long-term cost of the three baseline algorithms is higher than the proposed MADDPG algorithm. When we compare each algorithm's percentage



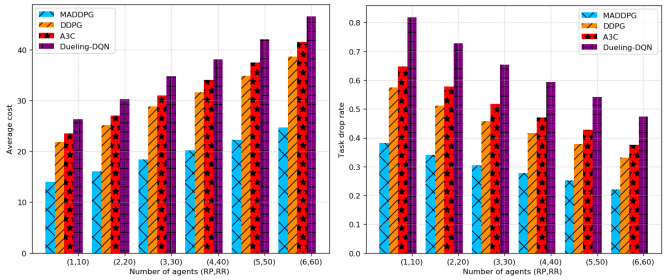
(a) RP comp. capacity vs average cost. (b) RP comp. capacity vs reward. (c) RP comp. capacity vs offl. rate.

Fig. 8. The impacts of CPU on average cost, offloading rate and RP reward.



(a) EH efficiency of IoTDs vs of-floading rate. (b) Number of IoTDs vs RP re-wards.

Fig. 9. The impacts of EH efficiency and number of IoTDs.



(a) Agents vs average costs. (b) Agents vs task drop rate.

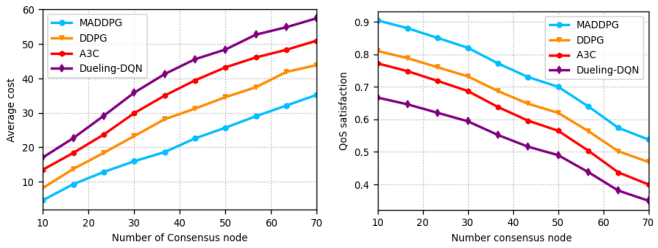
Fig. 10. Scalability performance.

according to the maximum computation capacity, the proposed MADDPG algorithm reduced the average long-term cost by 38.33%, 46.376%, and 63% for DDPG, A3C, and Dueling-DQN algorithms, respectively. The computation capacity of RPs also impacts the offloading rate of IoTDs and RPs' utilities, as shown in Fig. 8 (b) and 8 (c). With an increase in the computation capacity of an RP, the offloading rate and RP reward also increase. The agents in MADRL in both stages cooperate and share information. Due to this, the computation costs are lower than others, and the offloading rate is also high. The proposed MADDPG algorithm has a better offloading rate and RP utilities.

(4) Performance analysis under different parameters: The EH efficiency of IoTD impacts on task offloading rate/resource requesting probability of IoTDs, and the number of IoTDs in the system affects the RP rewards. Fig. 9 (a) shows that when the EH efficiency increases, the IoTDs can request more resources, offloading rate increases, or task drop rate decreases because the energy-hungry problem is somehow solved. We observed that the proposed MADDPG algorithm could create more opportunities to access more resources than the three baseline algorithms.

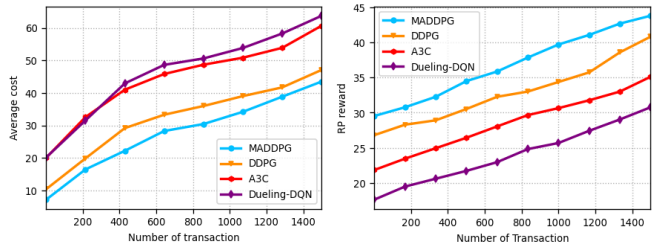
With increasing number of IoTDs requesting resources from the RPs in the system, the average long-term cost of RP increases but with more incentive gain. Fig. 9 (b) shows that the RP rewards gradually increases with the increasing number of IoTDs that request resources. With DDPG, A3C, and Dueling-DQN algorithms, the RP reward is smaller than in the proposed MADDPG algorithm. Therefore, our proposed MADDPG algorithm has a vital role, such as minimizing computation cost, optimizing price and maximizing RPs' rewards, in a secure manner. When the number of agents (RP, RR) increases, the cooperation between them increases in the dynamic ATG network. Due to this, the RP can obtain better rewards and the RR also minimizes costs. Based on these concepts, we validate the proposed framework's scalability in our scenario where the number of agents is varied. As shown in Fig. 10 (a), the number of agents (RP, RR) range from (1, 10) to (6, 60) in all algorithms in our scenario. The proposed MADDPG algorithm achieves more eminent performance than the three algorithms. We observed that as the number of agents increases, the average costs increase, but the proposed MADDPG algorithm cost is lower than others and increases slowly. Fig. 10 (b) shows the task drop rate in different algorithms where the agents are setting from (1, 10) to (6, 60). The task drop rate decreases by all algorithms at different rates. However, the proposed MADDPG algorithm task drop rate is the smallest of the other three algorithms. Under the proposed MADDPG algorithm, the agents are more cooperative and have better experience than DDPG, A3C, and Dueling-DQN algorithms, where the number of participants increases. Therefore, the proposed MADDPG algorithm in our proposed scenario has a better scalability than the other three algorithms and is applicable in multi-agent system scenarios.

(5) Blockchain Analysis: In Figs. 11 and 12, we discuss the effects of blockchain using the number of consensus nodes and transactions in our proposed MADDPG algorithm. We measured the performance of the blockchain utilized in this system based on costs and QoS satisfaction of IoTDs as shown in Fig. 11. As the number of consensus nodes increases, the average cost increases simultaneously, as shown in Fig. 11 (a). The nodes compete with each other until they



(a) Consensus node vs average cost. (b) Consensus node vs QoS satisfaction.

Fig. 11. The impacts of consensus nodes.



(a) Transaction vs system cost. (b) Transaction vs RR reward.

Fig. 12. The impacts of transaction.

reach an agreement or consensus. Due to this, the energy consumption of the MEC server is high, and the latency is also high. The proposed MADDPG algorithm outperformed the baseline algorithms. Fig. 11 (b) shows that as the number of consensus nodes increases, the QoS satisfaction of IoTDs decreases in all compared algorithms. This is due to the fact that transaction confirmation latency increases as the number of consensus nodes increases, which results in a decreased QoS satisfaction of IoTDs.

Fig. 12 shows the average cost of the system and the RPs' reward as the transaction size increases. As shown in Fig. 12 (a), the average cost of the system increases as the number of transactions in a single block increases. The possible reasons are that when the number of transactions in a single block increases, it require more resources, i.e., power, and high processing latency, affecting the network performance. The average cost of the system increases rapidly in the baseline algorithms compared to the proposed algorithms. The proposed MADDPG algorithm can minimize costs because agents can share experiences and cooperate. As shown in Fig. 12 (b), the impact of the number of transactions on RP rewards and RP rewards increases slowly as the number of transactions increases in all algorithms. The proposed MADDPG algorithm RPs reward is better than other baseline algorithms.

VII. Conclusion

In this paper, we studied a new framework for task offloading with EH in multi-UAV-assisted IoT networks that combines MADRL algorithms, Stackelberg game and consortium blockchain to secure transactions between RRs and RPs and optimize the utilities of agents in the system (minimize RR computation costs and maximize RP utilities). We deployed a cluster-based multi-UAV network, with each cluster being controlled by UCH for computing services, energy transmission to IoTDs, and information relaying to SDN. The edge servers on BS or UCH are used as an RP to sell resources to IoTDs. We formulated the optimization problems of RP and IoTDs as a Stackelberg game model and it transformed into MDP games to handle the complexity and dynamics in the system. We then applied a MADRL algorithm called the MADDPG to solve the problems that adopted centralized training with a decentralized execution algorithm. Each RP and RR acts as an agent to obtain an optimal offloading policy with EH and better utilities. To achieve the objective, the agent cooperates and shares information, while making an independent decision. The MADDPG algorithm ensures a better convergence than baseline DRL approaches such as DDPG, A3C, and Dueling-DQN. The simulation results showed that the proposed MADDPG algorithm in blockchain-based computation offloading with EH minimizes the computation costs and improves RPs' utilities. Future work will integrate blockchain technology, distributed MADRL (federated learning), and virtualization technology in the hierarchical non-terrestrial network (NTN) network infrastructure.

References

[1] F. Qi, X. Zhu, G. Mang, M. Kadoch, and W. Li, "UAV network and IoT in the sky for future smart cities," *IEEE Netw.*, vol. 33, no. 2, pp. 96–101, Mar./Apr. 2019.

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

[3] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile Internet of Things: Can UAVs provide an energy-efficient mobile architecture?" in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[4] A. M. Seid, G. O. Boateng, S. Anokye, T. Kwantwi, G. Sun, and G. Liu, "Collaborative computation offloading and resource allocation in multi-UAV-assisted IoT networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12203–12218, Aug. 2021.

[5] A. M. Seid, G. O. Boateng, B. Mareri, G. Sun, and W. Jiang, "Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4531–4547, Dec. 2021.

[6] G. Tuna, B. Nefzi, and G. Conte, "Unmanned aerial vehicle-aided communications system for disaster recovery," *J. Netw. Comput. Appl.*, vol. 41, pp. 27–36, May 2014.

[7] N. Zhao et al., "UAV-assisted emergency networks in disasters," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 45–51, Feb. 2019.

[8] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Sep. 2016.

[9] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 443–461, 3rd Quart., 2011.

[10] L. Jiang, S. Xie, S. Maharjan, and Y. Zhang, "Blockchain empowered wireless power transfer for green and secure Internet of Things," *IEEE Netw.*, vol. 33, no. 6, pp. 164–171, Nov. 2019.

[11] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless charging technologies: Fundamentals, standards, and network applications," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1413–1452, 2nd Quart., 2016.

[12] G. Yang, C. K. Ho, and Y. L. Guan, "Dynamic resource allocation for multiple-antenna wireless power transfer," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3565–3577, Jul. 2014.

[13] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

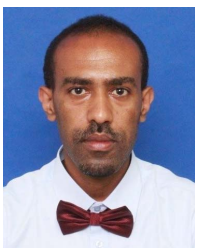
[14] Z. Wei, B. Zhao, J. Su, and X. Lu, "Dynamic edge computation offloading for Internet of Things with energy harvesting: A learning method," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4436–4447, Jun. 2019.

[15] J. Zhang, J. Du, Y. Shen, and J. Wang, "Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9303–9317, Oct. 2020.
[16] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cogn. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.
[17] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5G and beyond networks: A state of the art survey," *J. Netw. Comput. Appl.*, vol. 166, Sep. 2020, Art. no. 102693.
[18] D. V. Medhane, A. K. Sangaiah, M. S. Hossain, G. Muhammad, and J. Wang, "Blockchain-enabled distributed security framework for next-generation IoT: An edge cloud and software-defined network-integrated approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6143–6149, Jul. 2020.
[19] X. Xu, H. Zhao, H. Yao, and S. Wang, "A blockchain-enabled energy-efficient data collection system for UAV-assisted IoT," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2431–2443, Feb. 2021.
[20] P. Mehta, R. Gupta, and S. Tanwar, "Blockchain envisioned UAV networks: Challenges, solutions, and comparisons," *Comput. Commun.*, vol. 151, pp. 518–538, Feb. 2020.
[21] A. Asheralieva and D. Niyato, "Distributed dynamic resource management and pricing in the IoT systems with blockchain-as-a-service and UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1974–1993, Mar. 2020.
[22] M. Li, F. R. Yu, P. Si, R. Yang, Z. Wang, and Y. Zhang, "UAV-assisted data transmission in blockchain-enabled M2M communications with mobile edge computing," *IEEE Netw.*, vol. 34, no. 6, pp. 242–249, Nov./Dec. 2020.
[23] T. N. Dinh and M. T. Thai, "AI and blockchain: A disruptive integration," *Computer*, vol. 51, no. 9, pp. 48–53, Sep. 2018.
[24] T. N. Dinh and M. T. Thai, "AI and blockchain: A disruptive integration," *Computer*, vol. 51, no. 9, pp. 48–53, Sep. 2018.
[25] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5G beyond," *IEEE Netw.*, vol. 33, no. 3, pp. 10–17, May/Jun. 2019.
[26] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, "UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2777–2790, Apr. 2020.
[27] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "On-board deep Q-network for UAV-assisted online power transfer and data collection," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12215–12226, Dec. 2019.
[28] X. Lin, J. Wu, A. K. Bashir, J. Li, W. Yang, and M. J. Piran, "Blockchain-based incentive energy-knowledge trading in IoT: Joint power transfer and AI design," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14685–14698, Aug. 2022.
[29] W. Chen et al., "Cooperative and distributed computation offloading for blockchain-empowered industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8433–8446, Oct. 2019.
[30] S. Luo et al., "Blockchain-based task offloading in drone-aided mobile edge computing," *IEEE Netw.*, vol. 35, no. 1, pp. 124–129, Jan. 2021.
[31] Z. Guan, H. Lyu, D. Li, Y. Hei, and T. Wang, "Blockchain: A distributed solution to UAV-enabled mobile edge computing," *IET Commun.*, vol. 14, no. 15, pp. 2420–2426, Sep. 2020.
[32] X. Xu, H. Zhao, H. Yao, and S. Wang, "A blockchain-enabled energy-efficient data collection system for UAV-assisted IoT," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2431–2443, Feb. 2021.
[33] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3602–3609, Jun. 2019.
[34] H. N. Abishu, A. M. Seid, Y. H. Yacob, T. Ayall, G. Sun, and G. Liu, "Consensus mechanism for blockchain-enabled vehicle-to-vehicle energy trading in the internet of electric vehicles," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 946–960, Jan. 2022.
[35] M. Belotti, N. Bozic, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3796–3838, 4th Quart., 2019.
[36] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.
[37] Z. Su, Y. Wang, Q. Xu, M. Fei, Y.-C. Tian, and N. Zhang, "A secure charging scheme for electric vehicles with smart communities in energy blockchain," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4601–4613, Jun. 2019.
[38] N. Malik and B. Joshi, "ECDSA approach for reliable data sharing and document verification using two level QR code," in *Proc. 2nd Int. Conf. I-SMAC (IoT Social, Mobile, Anal. Cloud) (I-SMAC)I-SMAC (IoT Social, Mobile, Analytics Cloud) (I-SMAC), 2nd Int. Conf.*, Aug. 2018, pp. 434–437.
[39] J. Kang, Z. Xiong, D. Ye, D. I. Kim, J. Zhao, and D. Niyato, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2906–2920, Mar. 2019.
[40] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*. Amsterdam, The Netherlands: Elsevier, 1994, pp. 157–163.
[41] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2017, pp. 6382–6393.
[42] Y. Zuo, S. Jin, S. Zhang, and Y. Zhang, "Blockchain storage and computation offloading for cooperative mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9084–9098, Jun. 2021.

Abegaz Mohammed Seid (Member, IEEE) received the B.Sc. degree in computer science from Ambo University, Ethiopia, in 2010, the M.Sc. degree in computer science from Addis Ababa University, Ethiopia, in 2015, and the Ph.D. degree in computer science and technology from the University of Electronic Science and Technology of China (UESTC) in 2021. He is currently a Research Assistant with the School of Computer Science and Engineering, Wuhan University of Science and Technology. His research interests include wireless networks, mobile edge computing, and blockchain.

Jianfeng Lu (Member, IEEE) received the Ph.D. degree in computer application technology from the Huazhong University of Science and Technology in 2010. He was a Visiting Researcher with the University of Pittsburgh, Pittsburgh, PA, USA, in 2013. He is currently a Professor with the School of Computer Science and Engineering, Wuhan University of Science and Technology, Wuhan, China. His major research interests include game theory and incentive mechanism with applications to sensor networks and mobile computing.

Hayla Nahom Abishu received the B.Sc. degree in computer science and information technology from Haramaya University in 2007, and the M.Sc. degree in computer science and networking from Dilla University, Ethiopia, in 2017. He is currently pursuing the Ph.D. degree in computer science and technology with the University of Electronic Science and Technology of China (UESTC). He is also a member of the Mobile Cloud-Network Research Team, UESTC. His research interests include mobile computing, wireless networks, blockchain, UAV, the IoT, and machine learning.



Tewodros Alemu Ayall received the B.Sc. degree in computer science from the University of Gondar, Ethiopia, in 2010, the M.Sc. degree in computer science from Andhra University, India, in 2015, and the Ph.D. degree in computer science and technology from the University of Electronic Science and Technology of China (UESTC), China, in 2021. He is engaged in research of distributed graph processing, distributed graph database, big data processing, big graph partitioning, and blockchain research.