EEDTO: An Energy-Efficient Dynamic Task Offloading Algorithm for Blockchain-Enabled IoT-Edge-Cloud Orchestrated Computing

Huaming Wu, Katinka Wolter, Pengfei Jiao, Yingjun Deng, Yubin Zhao and Minxian Xu

***Abstract*—With the proliferation of compute-intensive and delay-sensitive mobile applications, large amounts of computational resources with stringent latency requirements are required on Internet of Things (IoT) devices. One promising solution is to offload complex computing tasks from IoT devices either to Mobile Edge Computing (MEC) or Mobile Cloud Computing (MCC) servers. MEC servers are much closer to IoT devices and thus have lower latency, while MCC servers can provide flexible and scalable computing capability to support complicated applications. To address the tradeoff between limited computing capacity and high latency, and meanwhile, ensure the data integrity during the offloading process, we consider a blockchain scenario where edge computing and cloud computing can collaborate towards secure task offloading. We further propose a blockchain-enabled IoT-Edge-Cloud computing architecture that benefits both from MCC and MEC, where MEC servers offer lower latency computing services, while MCC servers provide stronger computation power. Moreover, we develop an Energy-Efficient Dynamic Task Offloading (EEDTO) algorithm by choosing the optimal computing place in an online way, either on the IoT device, the MEC server or the MCC server with the goal of jointly minimizing the energy consumption and task response time. The Lyapunov optimization technique is applied to control computation and communication costs incurred by different types of applications and the dynamic changes of wireless environments. During the optimization the best computing location for each task is chosen adaptively without requiring future system information as prior knowledge. Compared with previous offloading schemes with/without MEC and MCC cooperation, EEDTO can achieve energy-efficient offloading decisions with relatively lower computational complexity.**

***Index Terms*—Task offloading, Mobile cloud computing, Mobile edge computing, Lyapunov optimization, Blockchain.**

I. Introduction

DUE to recent advances in hardware and software technologies, the number of Internet of Things (IoT) devices (e.g. smartphones, wearable devices, Unmanned Aerial Vehicles (UAV), smart vehicles) has increased significantly. According to Cisco, more than 30 billion active IoT devices generate approximately 2.5 exabytes of data per day [1], which needs to be further processed and stored. Besides, there is a substantial increase in the number of real-time and latency-sensitive applications, e.g., smart transportation, smart health-care, augmented reality, and smart buildings that require large amounts of computing and network resources. However, IoT devices are constrained by limited resources, such as CPU computing power, storage space, energy capacity, and environmental awareness, complex computing tasks, e.g., UAV-based Virtual Reality/Augmented Reality (VR/AR) gaming [2]–[4], are inefficient when run locally. Furthermore, a variety of emerging IoT applications, e.g., delay-sensitive and delay-tolerant applications, will incur a different amount of computation and communication costs.

To overcome the above contradictions, one effective way is to take advantage of Mobile Cloud Computing (MCC) by offloading complex computing tasks from IoT devices to a remote cloud via a Wide Area Network (WAN). By exploiting the benefits of the rich virtual resources and computing capacity of the cloud server, we can release the computing burden on IoT devices in handling tasks locally, and thus reduce task response time; at the same time, when tasks are handled in the cloud, the main modules of IoT devices are in an idle state, thereby reducing the energy consumption of devices. However, it is not always appropriate to offload computation and data to the cloud, especially for data-intensive and delay-sensitive tasks. The excessive pressure on cloud computing services leads to high latency and significant network bandwidth usage. IoT devices suffer from high latency and low bandwidth when communicating with distant MCC servers. In addition, the large amount of data arriving at MCC servers and resource-hungry applications requires more computing and storage, which may cause an overload of the MCC servers. Hence, MCC servers not only fail to efficiently satisfy the requirements of real-time and latency-sensitive applications, but also increase the energy consumption of IoT devices due to low bandwidth.

Rather than relying on this centralized service mode, we can also seek the help of Mobile Edge Computing (MEC), in which migrating computations and data to network edge servers can enhance the processing capabilities of IoT devices and alleviate their resource shortages. IoT devices connect to nearby MEC servers rather than directly to distant MCC servers. Due to their proximity to mobile users, the communication cost during task offloading will drop significantly, which can greatly reduce the network latency. With the development of the Internet of Everything (IoE), it is gradually realized that a single computing paradigm cannot solve all problems, while

H. Wu and Y. Deng are with Center for Applied Mathematics, Tianjin University, Tianjin 300072, China. E-mail: {whming, yingjun.deng}@tju.edu.cn

K. Wolter is with Institut für Informatik, Freie Universität Berlin, Berlin 14195, Germany. E-mail: katinka.wolter@fu-berlin.de

P. Jiao is with Center of Biosafety Research and Strategy, Tianjin University, Tianjin 300072, China. E-mail: pjiao@tju.edu.cn

Y. Zhao and M.Xu are with Shenzhen Institutes of Advanced Technology, CAS, Shenzhen 518055, China. E-mail: {zhaoyb, mx.xu}@siat.ac.cn

(Corresponding author: Huaming Wu)

most existing researches still focus on either MCC or MEC to offload tasks with stringent delay requirements. In fact, we can take the heterogeneity of MCC and MEC into account by constructing a hybrid computing environment for offloading destination selection. However, due to the heterogeneous resources among MEC and MCC servers, how to make the optimal offloading decision effectively and efficiently in an edge-cloud computing environment remains a challenge.

In addition, it is generally unsafe for mobile users to offload application tasks to MEC/MCC servers in an untrusted and non-transparent environment. Data loss or privacy leakage is likely to occur during the task offloading process [5]. Due to the unique security and decentralized features, blockchain technology can be introduced into edge computing as a potential solution for ensuring data integrity and prevent illegal offloading behaviors. Therefore, we develop an IoT-Edge-Cloud computing model that supports blockchain, where MEC servers provide a low-latency computing service for latency-intensive applications, while MCC servers offer powerful computing capacity for resource-intensive applications while maintaining the offloading security. We try to derive an adaptive offloading decision algorithm based on Lyapunov optimization, which globally determines when to offload, through which network, and where to process each task (i.e., IoT device, MEC server or MCC server) such that the total energy consumption can be minimized by leveraging delay tolerance. The main contributions of this paper are threefold:

- Considering the characteristics of the abundant computing resources in MCC and the low transmission delay in MEC comprehensively, we design a blockchain-enabled IoT-Edge-Cloud offloading architecture that benefits both from MCC and MEC, where MEC servers can provide lower latency computing services, and MCC servers can provide stronger computation power while ensuring the offloading security in IoT.
- In order to extend the battery lifetime of IoT devices, a mathematical model is established by minimizing energy consumption under a given delay constraint. Multi-criteria are applied to decide whether each task should run on the local device, the nearby MEC server or the remote MCC server.
- We design a cost-driven scheduling strategy between communication and computation according to the IoT-Edge-Cloud hybrid task offloading model. We propose an online offloading algorithm based on *Lyapunov optimization* for decision-making over the cloud, edge, and IoT devices. Simulation results verify that EEDTO significantly reduces the energy consumption of IoT devices with a lower delay penalty.

The rest of this paper is organized as follows. Section II discusses relevant studies. Section III introduces a collaborative edge-cloud computation offloading scenario. The proposed EEDTO algorithm is described in Section IV. Section V analyzes the simulation results from different aspects. Finally, Section VI concludes the paper.

II. Related Work

To address important challenges over IoT systems and emerging computing paradigms that consist of cloud and edge computing (where MCC and MEC servers work collaboratively), effective offloading decision-making has been studied to maximize the performance gain from different perspectives.

A. Traditional Offloading Decisions

Several papers [18]–[22] have focused on solving the multi-objective optimization problem of task offloading decisions either in MEC or MCC systems.

A non-dominated sorting genetic algorithm was proposed in [6] to address the multi-objective computation offloading problem in IoT-enabled cloud-edge computing environments. An end-edge-cloud computing offloading method was proposed in [7] to tackle the optimization problem of offloading decision-making in heterogeneous IoT environments with multiple MEC servers, which employed a modified strength Pareto evolutionary algorithm. A computing offloading game theory was proposed in [8], which employed C-SGA (a fast Stackelberg game algorithm) and F-SGA (a complex Stackelberg game algorithm) to solve the problem. A weighted cost model was designed in [22] to minimize the execution time and energy consumption of IoT applications in a computing environment with multiple IoT devices, multiple fog/edge servers, and MCC servers. And then a Memetic algorithm-based offloading technique was proposed to make batch application offloading decision for finding suitable servers in time.

However, we still need to address these heterogeneity challenges in the integrated MEC and MCC systems. To enable mobile applications smoothly running on IoT devices in heterogeneous edge and cloud computing environments, the graph partition of application tasks between IoT devices and MEC/MCC servers should be made adaptively [23] and offloading decisions should be taken dynamically in a timely manner.

B. Advanced Offloading Decisions

Apart from that, enabler technologies like blockchain [16], [17], [24], Deep Reinforcement Learning (DRL) [13], [25], [26] and federated learning [27]–[29] are also being incorporated in the MCC or MEC system for offloading decision-making.

Deep learning is playing an increasingly important role in solving real-world IoT scenarios, e.g., to effectively enhance the utilization of available resources for the maximization of the system performance and minimization of energy consumption [29]. Moreover, intelligence decision-making shows significant promise for dealing with task offloading with the collaboration of MEC and MCC [30]. For example, DDLO [13] and DDTO [14] are distributed deep learning-based offloading schemes, both of which use multiple parallel DNNs to generate near-optimal offloading decisions in real-time. Neurosurgeon [31], a fine-grained partitioning method based on neural networks, can find the optimal dividing point according to different factors, and take advantage of abundant

TABLE I: Related work and comparison to our proposed EEDTO

Categories	Offloading Schemes	Offloading Mode	Architectural Properties			Optimization Objective(s)		
			MCC	MEC	MCC & MEC	Latency	Energy	Security
Traditional Offloading Decision-Making	COM [6]	Partial	✓	✗	✗	✓	✓	✗
	EECCCO [7]	Partial	✓	✓	✗	✓	✓	✗
	F-SGA&C-SGA [8]	Partial	✗	✓	✗	✓	✗	✗
	SDTO [9]	Full	✗	✓	✗	✓	✓	✗
	EEDOA [10]	Full	✗	✓	✗	✗	✓	✗
	MCCG [11]	Partial	✗	✗	✓	✓	✗	✗
	Scheme in [12]	Full	✓	✗	✗	✓	✗	✓
Advanced Offloading Decision-Making	DROO [13]	Partial	✗	✓	✗	✗	✓	✗
	DDTO [14]	Partial	✗	✗	✓	✓	✓	✗
	DRLO [15]	Partial	✗	✓	✗	✗	✓	✗
	BeCome [16]	Partial	✗	✓	✗	✗	✓	✓
	CQL [17]	Partial	✗	✗	✓	✓	✗	✓
	Proposed EEDTO	Partial	✓	✓	✓	✓	✓	✓

resources of cloud servers to minimize the execution time or energy consumption. DRL has been claimed to solve problems in different areas, e.g. Internet of Vehicles (IoV) [25], UAVs [4] and industrial IoTs [32]. The vehicular edge computing problem has been solved by DRL-based algorithms in [26], [33], [34]. In these works, however, the heterogeneity of diverse servers is still neglected in making offloading decisions.

Mobile users may mistakenly migrate their computing tasks from IoT devices to nearby MEC servers or a remote MCC system intruded by some network attackers, resulting in data loss and privacy and security problems during task offloading. As the core technology of bitcoin digital currency, blockchain is an emerging paradigm for guaranteeing data integrity [35]. BeCome [16] is a blockchain-enabled computation offloading method designed specifically for MEC. By utilizing additive weighting and multi-criteria decision-making, the optimal offloading strategy is identified to ensure data integrity during task offloading. Nguyen et al. [24] presented a novel Reinforcement Learning (RL)-based offloading scheme, which enables IoT devices to make optimal decisions according to blockchain transaction states and wireless channel quality between IoT devices and MEC servers. Moreover, a multihop collaborative and distributed offloading-decision algorithm was designed in [32], which combined data processing tasks with mining tasks and minimized the economic cost of blockchain-empowered IoT devices. However, many recent studies [36]–[38] fail to consider the joint optimization of task offloading and blockchain mining, which has a significant impact on effectively balancing the minimum computing cost and improving user privacy level. Offloading schemes that support blockchain can also be jointly optimized on the basis of task execution time, energy consumption, load balance, and data integrity.

C. A Qualitative Comparison

Table I provides a comparison between the proposed EEDTO with other related work. Edge and cloud hybrid computing systems could be the future of next-generation computing models. However, all of these studies are still difficult to balance complexity, security, and optimality, ignoring the cooperation between MEC and MCC. Therefore, a lightweight offloading decision algorithm with a relatively low computational complexity needs to be developed.

Many previous optimization-based offloading schemes [10], [39], [40] apply stochastic optimization techniques such as Lyapunov optimization, which typically chooses the energy queue as Lyapunov drift. However, the energy queue is normally time-dependent, which makes the allowable offloading action sets not i.i.d., to cope with this issue, we select the response time as the drift instead. Besides, to eliminate the potential data loss or privacy leakage during the task offloading process, it is necessary to develop a lightweight and secure task offloading framework by integrating the blockchain technology.

In view of the serious limitations of current centralized IoT systems, e.g., a single point of failure, data privacy, security and robustness, blockchain-assisted edge computing and cloud computing can provide attractive solutions. Thus, it is necessary to study blockchain-enabled task offloading methods in MCC and MEC heterogeneous environments. By integrating the blockchain and edge computing, it can improve the query delay and offloading security. We design a task offloading framework based on blockchain to ensure data integrity in the process of task offloading, while optimizing application response time, energy consumption and maintaining load balance. To improve the performance of the combined MEC and MCC systems, a joint scheme of task offloading and blockchain mining has been proposed. For instance, smart contracts can be used to support transparent resource transactions to reduce the computing burden, and reputation mechanisms can also be used to accelerate the mining process. On this basis, aiming at the QoS service demand of mobile users and severe energy consumption constraints, a Lyapunov optimization-based framework is proposed to maximize the revenue of edge services and the reward of blockchain mining, while minimizing the energy consumption of service computing.

III. System Model and Problem Formulation

In this section, we first introduce the system architecture design and then formulate the dynamic decision problem for edge-cloud hybrid task offloading.

A. System Overview

According to different computing capabilities, energy costs, and latency, diverse types of computing resources, e.g., local clouds, micro clouds, remote clouds, or nearby MEC servers can be effectively integrated and utilized. Even if a computing resource (e.g., a MCC server) cannot meet the task offloading requirement, the mobile user can still use the offloading service of other computing resources (e.g., a MEC server), and the usability is greatly enhanced. In fact, MCC and MEC are complementary to each other, MCC has abundant computing resources and applications, while MEC has the characteristics of short delays, strong stability, and adaptability to diverse network environments, providing strong support for delay-sensitive services [41].

We intend to leverage the advantages of MCC and MEC, and offload tasks to locations that have different computing and communication capabilities when satisfying the condition of delay constraints. Thus, we try to integrate heterogeneous computing resources and build a collaborative IoT-Edge-Cloud computing environment. A blockchain-based mining task offloading environment is depicted in Fig. 1. It consists of a three-tier hierarchy, i.e., an IoT device layer, a MEC server layer, and a MCC server layer, which is deployed and managed by the blockchain network. The main features of each layer are as follows:

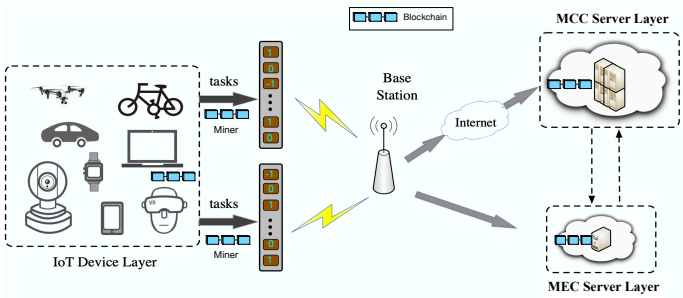


Fig. 1: A blockchain-based task offloading framework in IoT-edge-cloud computing environment

- *IoT device layer*: This layer includes a network of IoT devices connected together on the blockchain. Each device owns a blockchain account to join the network and connect to MEC and MCC servers through wireless Access Points (APs). In each offload cycle, the IoT user needs to make dynamic task offloading decisions based on Quality of Service (QoS) requirements and current network conditions (task size, available edge resources, transmission bandwidth resources, etc.) [42] to decide where to execute mining tasks, either executing the mining task on the local IoT device (*decision*: 0), offloading it to a nearby MEC server (*decision*: -1) or offloading it to a remote MCC server (*decision*: 1), to achieve the best computing benefit (i.e. the minimum offloading cost).
- *MEC server layer*: This layer includes lightweight MEC servers for real-time task processing, which can provide low-latency computing services at the network edge. However, for complex computational tasks, MEC servers need to forward them to the resource-rich MCC servers through wired lines to avoid task overload. In addition, MEC servers also act as blockchain entities, establishing reliable communication with the IoT device layer and the MCC server layer, to ensure the security of offloading activities [43].
- *MCC server layer*: This layer includes multiple virtual machines with powerful computing and storage capabilities for solving complex computing tasks of local IoT devices. All cloud nodes operate in a decentralized and secure manner, and are securely linked to MEC servers and IoT devices through the blockchain network. Data offloaded from IoT devices can be safely stored in distributed cloud storage on the blockchain.

In this three-tier model, each layer has distinct processing capabilities. The blockchain application is first partitioned into multiple tasks and then offloaded to the MEC/MCC server. On one hand, running mining tasks locally may cause a huge amount of energy consumption of IoT devices; on the other hand, offloading tasks to MEC servers or MCC servers will significantly reduce the energy cost, but will introduce data transmission delay and server queuing delay. To balance the trade-offs between latency and energy consumption, which greatly affect the application performance, we dynamically make offloading decisions of whether to offload (IoT devices/servers), and where to offload (MEC/MCC servers). We perform task offloading according to the task complexity and the realistic IoT environment, i.e., compute-intensive tasks are offloaded to the MCC server and data-intensive tasks are offloaded to the MEC server, thereby alleviating load bottlenecks, delays, and fault tolerance. Furthermore, optimization issues are developed to effectively address the challenge of heterogeneous IoT resources. We make full use of the complementary advantages of different types of wireless networks (e.g., wireless LANs and cellular networks) to perform task offloading. The blockchain guarantees the security and data integrity of the MEC server and the MCC server.

B. Offloading Decision Model

The IoT-Edge-Cloud computing model can be described as a weighted call graph $G = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \{v_1, v_2, \cdots, v_N\}$ is the set of nodes that represent there are multiple computing tasks to be allocated to the IoT devices, MEC servers, or MCC servers, which provide different computation capabilities. We assume that there are N computing tasks, once there is an application execution request, a controller in the IoT device will determine which tasks should be handled locally on the IoT device, which ones to be processed on the MEC server, and which ones to be executed on the MCC server. The weight of each node w_{v_i} represents the computational cost when the task is executed in one of these places. Generally, the computing power of IoT devices, MEC servers and MCC servers are satisfying: $f_{\text{IoT}} < f_{\text{edge}} < f_{\text{cloud}}$, implying that

TABLE II: Symbols and Definitions

Symbols	Definition
$I_{v_i}(t)$	The offloading strategy of computing task v_i
w_{v_i}	The computation workload of computing task v_i
D_{v_i,v_j}	The communication data to transfer from task v_i to v_j
p_{ex}	The CPU power of the IoT device
p_{idle}	The idle power of the IoT device
p_{tr}	The transmission power of the IoT device
T_{wan}	The latency of WAN
T_{lan}	The latency of LAN
B_{wan}	The bandwidth of WAN
B_{lan}	The bandwidth of LAN
f_{IoT}	The computing frequency of the IoT device
f_{edge}	The computing frequency of the MEC server
f_{cloud}	The computing frequency of the MCC server
$T_{v_i}^{\text{IoT}}$	The time taken when processing task v_i on the IoT device
$T_{v_i}^{\text{cloud}}$	The time taken when processing task v_i on the MCC server
$T_{v_i}^{\text{edge}}$	The time taken when processing task v_i on the MEC server
$E_{v_i}^{\text{IoT}}$	The energy consumption when processing task v_i on the IoT device
$E_{v_i}^{\text{cloud}}$	The energy consumption when processing task v_i on the MCC server
$E_{v_i}^{\text{edge}}$	The energy consumption when processing task v_i on the MEC server
$T_{v_i,v_j}^{\text{cloud}}$	The communication time between task v_i and task v_j when offloading task v_i to the MCC server
$T_{v_i,v_j}^{\text{edge}}$	The communication time between task v_i and task v_j when offloading task v_i to the MEC server

the MCC server has the strongest computing power, followed by the MEC server and then the IoT devices [6]. Moreover, $\mathbb{E} = \{e_{v_i,v_j} | v_i, v_j \in \mathbb{V}\}$ is the edge set, where e_{v_i,v_j} is a one-hop link between nodes v_i and v_j. Each link provides the communication cost for data transmission. Furthermore, D_{v_i,v_j} along the edge e_{v_i,v_j} represents the size of data that migrates from node v_i to node v_j. Table II summarizes key notations and their respective definitions.

Constructing the weighted call graph G is critical for offloading decision-making among the IoT devices, MEC servers and MCC servers, which closely depends on profiling techniques. To achieve this, we can use a program profiler, a network profiler, and an energy profiler to collect detailed information about the application tasks, IoT devices and network characteristics [23]. We try to find the optimal offloading decision that allocates N tasks to the IoT device, MEC server and MCC server such that the IoT device consumes the least energy when satisfying the following constraints [44]:

- Minimizing energy consumption on IoT devices.
- Satisfying the given deadline for executing each IoT application.
- Partitioning the application tasks into different categories (e.g., execute on IoT devices, MEC servers or MCC servers) dynamically.

At the t^{th} execution, the indicator $I_{v_i}(t)$ represents diverse offloading strategies for task v_i as follows:

$$I_{v_i}(t) = \begin{cases} 0, & \text{if task } v_i \text{ is executed on IoT device,} \\ 1, & \text{if task } v_i \text{ is offloaded to MCC server,} \\ -1, & \text{if task } v_i \text{ is offloaded to MEC server,} \end{cases} \quad (1)$$

where $I_{v_i}(t) = 0$ denotes task v_i is executed locally, $I_{v_i}(t) = 1$ denotes task v_i is offloaded to the remote MCC server, and $I_{v_i}(t) = -1$ denotes task v_i is offloaded to the nearby MEC server.

Here, we define an offloading-decision matrix as follows:

$$\mathbb{I}(t) = \{I_{v_1}(t), I_{v_2}(t), \cdots, I_{v_i}(t), \cdots, I_{v_N}(t)\}, \quad (2)$$

where each $I_{v_i}(t)$ can be selected from $\{0, 1, -1\}$. For each execution, the search for the optimal solution (i.e., determining whether $I_{v_i}(t)$ should be 0, 1 or -1) grows exponentially with the number of tasks [45], denoted by $|\mathbf{\Phi}| = 3^N$, where $\mathbf{\Phi}$ is the set of all possible offloading decision combinations. Hence, obtaining the optimal decision combination directly in a timely manner is intractable.

1) IoT Computing Model: Due to resource constraints like battery capacity, IoT devices can only perform basic tasks. As for latency-intolerant applications, they should make real-time decisions with limited capabilities.

The execution time spent when processing the computing task v_i locally on the IoT device, is expressed as:

$$T_{v_i}^{\text{IoT}} = \frac{w_{v_i}}{f_{\text{IoT}}}, \quad \forall v_i \in \mathbb{V}, \quad (3)$$

where w_{v_i} is the computation workload of the computing task v_i and f_{IoT} denotes the computing frequency of the IoT device.

Similarly, the amount of energy spent on the IoT device for executing task v is formulated as:

$$E_{v_i}^{\text{IoT}} = p_{\text{ex}} \cdot T_{v_i}^{\text{IoT}}, \quad \forall v_i \in \mathbb{V}, \quad (4)$$

where p_{ex} is the power for task execution on the IoT device.

2) Cloud Computing Model: Compared to IoT devices and MEC servers, MCC servers can provide more powerful computing capacities.

The time spent when offloading the computing task v_i to the MCC server for execution is expressed as:

$$T_{v_i}^{\text{cloud}} = \frac{w_{v_i}}{f_{\text{cloud}}} + T_{\text{wan}}, \quad \forall v_i \in \mathbb{V}, \quad (5)$$

where f_{cloud} denotes the computing frequency of the MCC server and T_{wan} represents the latency of WAN.

The communication time between tasks v_i and v_j can be calculated by:

$$T_{v_i,v_j}^{\text{cloud}} = \frac{D_{v_i,v_j}}{B_{\text{wan}}}, \quad \forall (v_i, v_j) \in \mathbb{E}, \quad (6)$$

where D_{v_i,v_j} is the transferred data between task v_i and v_j and B_{wan} represents the bandwidth of WAN. The communication time closely depends on the amount of data to be transmitted and the network bandwidth between the IoT device and the MCC server, which is relatively low and sometimes even unstable.

Further, the energy consumption spent on the IoT device when task v_i is offloaded to the MCC server for execution is given by:

$$E_{v_i}^{\text{cloud}} = p_{\text{idle}} \cdot T_{v_i}^{\text{cloud}}, \quad \forall v_i \in \mathbb{V}, \tag{7}$$

where p_{idle} is the power when the IoT device is in the idle state, i.e., the task is being executed outside the IoT device.

3) Edge Computing Model: MEC servers in the edge layer located in the proximity of IoT devices and communicate with them through different wireless communication technologies, e.g., Bluetooth and WiFi. By using a Local Area Network (LAN) together with IoT devices [46], MEC servers can provide a low-latency computing service.

The execution time spent when offloading task v_i to the MEC server for execution is expressed as:

$$T_{v_i}^{\text{edge}} = \frac{w_{v_i}}{f_{\text{edge}}} + T_{\text{lan}}, \quad \forall v_i \in \mathbb{V}, \tag{8}$$

where f_{edge} denotes the computing frequency of the MEC server and T_{lan} represents the latency of LAN.

The transmission time between tasks v_i and v_j can be calculated by:

$$T_{v_i,v_j}^{\text{edge}} = \frac{D_{v_i,v_j}}{B_{\text{lan}}}, \quad \forall (v_i, v_j) \in \mathbb{E}, \tag{9}$$

where B_{lan} represents the bandwidth of LAN. Mostly, we have $B_{\text{wan}} \leq B_{\text{lan}}$.

Further, the energy consumption spent on the IoT device when offloading task v_i to the MEC server for execution, is given by:

$$E_{v_i}^{\text{edge}} = p_{\text{idle}} \cdot T_{v_i}^{\text{edge}}, \quad \forall v_i \in \mathbb{V}. \tag{10}$$

C. Problem Formulation

1) Response Time Model: The total response time includes the execution time of tasks running on the IoT device, the MEC server and the MCC server, respectively, as well as extra communication time for data transferring when locating in different places, which is formulated as:

$$\begin{aligned} T(\mathbb{I}(t)) = &\underbrace{\sum_{v_i \in \mathbb{V}} (1 - |I_{v_i}(t)|) \cdot T^{\text{ex}}(I_{v_i}(t))}_{\text{IoT device}} + \\ &\underbrace{\sum_{v_i \in \mathbb{V}} |I_{v_i}(t)| \cdot T^{\text{ex}}(I_{v_i}(t))}_{\text{MEC/MCC server}} + \\ &\underbrace{\sum_{(v_i,v_j) \in \mathbb{E}} \left[2 - |I_{v_i}(t) - I_{v_j}(t)|\right] \cdot T^{\text{tr}}(I_{v_i}(t), I_{v_j}(t))}_{\text{communication}}, \end{aligned} \tag{11}$$

where $I_{v_i}(t)$ and $I_{v_j}(t)$ are elements from Eq. (2). The execution time is:

$$T^{\text{ex}}(I_{v_i}(t)) = \begin{cases} T_{v_i}^{\text{IoT}}, & \text{if } I_{v_i}(t) = 0 \\ T_{v_i}^{\text{cloud}}, & \text{if } I_{v_i}(t) = 1 \\ T_{v_i}^{\text{edge}}, & \text{if } I_{v_i}(t) = -1 \end{cases} \tag{12}$$

The communication time between task v_i to v_j can be calculated by:

$$T^{\text{tr}}(I_{v_i}(t), I_{v_j}(t)) = \begin{cases} T_{v_i,v_j}^{\text{cloud}}, & \text{if } (I_{v_i}(t)+1) \odot (I_{v_j}(t)+1) = 0 \\ T_{v_i,v_j}^{\text{edge}}, & \text{if } (I_{v_i}(t)+1) \odot (I_{v_j}(t)+1) = 2 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where $\odot$ is NOR computation for binary variable.

2) Energy Consumption Model: The total energy consumption is composed of the energy consumed by the tasks running locally, the energy consumed in the idle state when performing certain tasks on the server, and the energy spent for communication, which is formulated as:

$$\begin{aligned} E(\mathbb{I}(t)) = &\underbrace{\sum_{v_i \in \mathbb{V}} (1 - |I_{v_i}(t)|) \cdot E^{\text{ex}}(I_{v_i}(t))}_{\text{IoT device}} + \\ &\underbrace{\sum_{v_i \in \mathbb{V}} |I_v(t)| \cdot E^{\text{ex}}(I_{v_i}(t))}_{\text{MEC/MCC server}} + \\ &\underbrace{\sum_{(v_i,v_j) \in \mathbb{E}} \left[2 - |I_{v_i}(t) - I_{v_j}(t)|\right] \cdot E^{\text{tr}}(I_{v_i}(t), I_{v_j}(t))}_{\text{communication}}, \end{aligned} \tag{14}$$

where the energy consumed for task execution on the IoT device can be expressed by:

$$E^{\text{ex}}(I_{v_i}(t)) = \begin{cases} E_{v_i}^{\text{IoT}}, & \text{if } I_{v_i}(t) = 0 \\ E_{v_i}^{\text{cloud}}, & \text{if } I_{v_i}(t) = 1 \\ E_{v_i}^{\text{edge}}, & \text{if } I_{v_i}(t) = -1 \end{cases} \tag{15}$$

and if we assume that the transmission power of the IoT device is fixed, the energy consumption for sending and receiving data between task v_i and v_j can be calculated by:

$$E^{\text{tr}}(I_{v_i}(t), I_{v_j}(t)) = p_{\text{tr}} \cdot T^{\text{tr}}(I_{v_i}(t), I_{v_j}(t)), \tag{16}$$

where p_{tr} is the power for data transferring.

3) Optimization Problem: To satisfy the requirements of resource-intensive and delay-sensitive applications, IoT devices can either offload part of their computational workload to a nearby MEC server, or to a remote MCC server under varying wireless environmental conditions. For a given offloading decision combination vector $\mathbb{I}(t)$, we define the execution indicator variable as:

$$\sigma\big(\mathbb{I}(t)\big) = \begin{cases} 0, & \text{if } T\big(\mathbb{I}(t)\big) \leq T_d, \\ 1, & \text{otherwise.} \end{cases} \tag{17}$$

where $\sigma\big(\mathbb{I}(t)\big) = 0$ if the total response time satisfies the deadline T_d, otherwise $\sigma\big(\mathbb{I}(t)\big) = 1$ if it fails to meet the delay constraint.

A stochastic optimization problem is then formulated for offloading decision-making for IoT-Edge-Cloud orchestrated computing. Formally, we have:

$$\min_{\mathbb{I}(t)} \quad \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{E\left(\mathbb{I}(\tau)\right)\big\}, \tag{18}$$

$$\text{s.t.} \quad \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{\sigma\left(\mathbb{I}(\tau)\right)\big\} \le \rho, \tag{19}$$

where ρ is the violation ratio, that is, the ratio of the number of executions that violate the deadline to the total number of executions. The intuitive idea of offloading all computing tasks to the MEC/MCC server does not work well, because it may cause too many tasks competing for the constrained resources and destabilizing the system. In order to minimize the average queue backlog and maintain system stability, we seek to bound the average queue backlog, as described in Eq. (19), which ensures that the system is stable, i.e., with finite upper bounds.

The considered optimization problem formulated in Eqs. (18) and (19) is Mixed-Integer Linear Programming (MILP) [47] and non-convex [48], typically known as NP-hard problem. Finding the optimal offloading solution is generally prohibitively due to the computation complexity, which will significantly increase owing to the binary variables.

IV. Energy-Efficient Dynamic Task Offloading Algorithm

A. Problem Transformation

To effectively solve Problem (18), we derive an adaptive offloading-decision algorithm by utilizing the Lyapunov optimization problem, which determines where to perform each task (i.e., IoT device, MEC server or MCC server) such that the energy consumption is minimized by leveraging delay tolerance. The optimal offloading-decision is made under which circumstances offloading is beneficial. Taking the average energy consumption as a penalty function, we dynamically determine when to offload tasks, while accepting small delays.

The system queue model is defined as:

$$Q(t+1) = \max[Q(t)-\rho, 0] + \sigma\big(\mathbb{I}(t)\big), \tag{20}$$

where $Q(t)$ is the system state at the t^{th} execution.

According to Lyapunov optimization theory [49], Lyapunov function is defined as:

$$L(Q(t)) = \frac{1}{2}Q^2(t). \tag{21}$$

Moreover, Lyapunov drift is defined as the change in Lyapunov function from one execution to the next, which is expressed as:

$$\begin{aligned} L(Q(t+1)) - L(Q(t)) &= \frac{1}{2}\big[Q^2(t+1) - Q^2(t)\big] \\ &= \frac{1}{2}\Big\{\big[\max[Q(t)-\rho,0] + \sigma\big(\mathbb{I}(t)\big)\big]^2 - Q^2(t)\Big\} \\ &\le \frac{\rho^2+\sigma^2(\mathbb{I}(t))}{2} + Q(t)\cdot\big[\sigma\big(\mathbb{I}(t)\big)-\rho\big]. \end{aligned} \tag{22}$$

Given the current state $Q(t)$, we define conditional Lyapunov drift $\Delta(Q(t))$ as the expected change in the continuous execution of the Lyapunov function:

$$\begin{aligned} \Delta(Q(t)) &\triangleq \mathbb{E}\big\{L(Q(t+1)) - L(Q(t))|Q(t)\big\}, \\ &\le C - \rho Q(t) + \mathbb{E}\Big\{Q(t)\sigma\big(\mathbb{I}(t)\big)|Q(t)\Big\}, \end{aligned} \tag{23}$$

where $C \triangleq \mathbb{E}\Big\{\frac{\rho^2+\sigma^2(\mathbb{I}(t))}{2}|Q(t)\Big\} = \frac{\rho^2}{2} + \mathbb{E}\Big\{\frac{\sigma^2(\mathbb{I}(t))}{2}|Q(t)\Big\}$.

In order to balance the average energy consumption and response time, we perform control actions at each execution to greedily minimize bounds of the weighted energy-plus-latency expression [49] as follows:

$$\Delta(Q(t)) + V\mathbb{E}\big\{E\big(\mathbb{I}(t)\big)|Q(t)\big\}, \tag{24}$$

where V is a weight control parameter, indicating the relative importance of minimizing energy consumption compared to the violation rate of the deadline.

After substituting Eq. (23) into Eq. (24), yields:

$$\begin{aligned} &\Delta(Q(t)) + V\mathbb{E}\big\{E\big(\mathbb{I}(t)\big)|Q(t)\big\} \le C - \rho Q(t) \\ &+ V\mathbb{E}\big\{E\big(\mathbb{I}(t)\big)|Q(t)\big\} + \mathbb{E}\big\{Q(t)\sigma\big(\mathbb{I}(t)\big)|Q(t)\big\} \\ &= C - \rho Q(t) + \mathbb{E}\Big\{\big[VE\big(\mathbb{I}(t)\big) + Q(t)\sigma\big(\mathbb{I}(t)\big)\big]|Q(t)\Big\}. \end{aligned} \tag{25}$$

We try to search for a feasible decision combination vector $\mathbb{I}(t)$ that greedily minimizes the decision criterion at the right-hand side of the Lyapunov function, which is described as follows:

$$\arg\min_{\mathbb{I}(t)} \big[VE\big(\mathbb{I}(t)\big) + Q(t)\sigma\big(\mathbb{I}(t)\big)\big]. \tag{26}$$

Furthermore, the offloading decision function is defined as:

$$\mathbb{D}\big(Q(t), \mathbb{I}(t)\big) = VE\big(\mathbb{I}(t)\big) + \sigma\big(\mathbb{I}(t)\big)Q(t). \tag{27}$$

where a decision combination vector $\mathbb{I}^*(t)$ is selected such that $\mathbb{D}\big(Q(t), \mathbb{I}^*(t)\big)$ is minimized. Therefore, $\mathbb{I}^*(t)$ is the optimal offloading solution among all feasible decision vectors. Based on Lyapunov optimization, the long-term stochastic optimization problem can be transformed into a deterministic optimization in each time slot. To solve the NP-hard problem (18), we then decouple it into sub-problems by transforming it into (27), which is much simpler than the original problem.

The EEDTO algorithm is designed to make offloading decisions greedily so that the upper bound will also be minimized at each time slot t. The EEDTO algorithmic process is described in detail, as shown in Algorithm 1. Firstly, we calculate $T\big(\mathbb{I}(t)\big)$ and $E\big(\mathbb{I}(t)\big)$ according to Eqs. (11) and (14). Then, we obtain the decision combination vector $\sigma\big(\mathbb{I}(t)\big)$ by judging whether the deadline T_d is violated or not. After that, the system queue in Eq. (20) is updated at each time slot. Obviously, when the waiting tasks in the buffer are minimized, the total energy consumption will be simultaneously minimized. Finally, the optimal offloading solution among all feasible decision vectors will be found.

Algorithm 1: An Energy-Efficient Dynamic Task Offloading (EEDTO) Algorithm

Input : $I_{v_i}(t)$: the offloading strategy for task v_i at the t^{th} execution.
Parameter: ρ: the violation ratio
T_d: the deadline;
$maxIter$: the maximum number of iterations.
Output : $\mathbb{I}^*(t)$: the optimal offloading solution among all feasible decision vectors.

1 **begin**
2 **for** $t = 0$ *to* $maxIter$ **do**
3 Calculate $T\big(\mathbb{I}(t)\big)$ and $E\big(\mathbb{I}(t)\big)$ according to Eq. (11) and Eq. (14);
4 **if** $T\big(\mathbb{I}(t)\big) \le T_d$ **then**
5 $\sigma\big(\mathbb{I}(t)\big) = 0$
6 **else**
7 $\sigma\big(\mathbb{I}(t)\big) = 1$
8 **end**
9 Update the system queue: $Q(t+1) = \max[Q(t) - \rho, 0] + \sigma\big(\mathbb{I}(t)\big)$; `/* buffering algorithm */`
10 **end**
11 Search for a decision combination vector $\mathbb{I}^*(t)$ that minimizes: $\mathbb{D}\big(Q(t), \mathbb{I}(t)\big)$; `/* scheduling algorithm */`
12 **end**
13 return $(\mathbb{I}^*(t))$

B. Performance Bounds Analysis

For any $V > 0$, average energy consumption and average queue backlog can be achieved when satisfying the following constraints [49]:

$$\bar{E} = \limsup_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\big\{E\left(\mathbb{I}(\tau)\right)\big\} \le \frac{C}{V} + E^*, \tag{28}$$

$$\bar{Q} = \limsup_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{Q(\tau)\} \le \frac{C + V(E^* - \bar{E})}{\varepsilon}, \tag{29}$$

where $\bar{E}$ is average energy consumption that can be arbitrarily close to the minimum energy consumption E^* with a diminishing gap $(1/V)$, while maintaining queue stability. However, this reduction is achieved at the expense of a larger delay since the average queue length $\bar{Q}$, i.e., an effective measure of average response time or latency, increases linearly with the rise of V.

Proof: The energy consumption can be reduced if we minimize the right-hand-side of Eq. (25) while ensuring the stability of the queue in Eq. (20). Given the observed $Q(\tau)$, we have:

$$\begin{aligned}\Delta\big(Q(\tau)\big) + V\mathbb{E}\big\{E\big(\mathbb{I}(\tau)\big)|Q(\tau)\big\} &\le C - \rho Q(\tau) \\ &+ V\mathbb{E}\big\{E\big(\mathbb{I}^*(\tau)\big)|Q(\tau)\big\} + \mathbb{E}\big\{Q(\tau)\sigma\big(\mathbb{I}^*(\tau)\big)|Q(\tau)\big\} \\ &\le C - \rho Q(\tau) + VE^* + Q(\tau)(\rho - \varepsilon) \\ &= C + VE^* - \varepsilon Q(\tau),\end{aligned}$$

where there exists an arbitrarily small $\varepsilon > 0$ that satisfies $\mathbb{E}\big\{\sigma\big(\mathbb{I}^*(\tau)\big)\big\} \le \rho - \varepsilon$ because the average violation rate $\mathbb{E}\big\{\sigma\big(\mathbb{I}^*(\tau)\big)\big\} \le \rho$.

Taking expectations of the above inequality and using the law of iterated expectations yields:

$$\begin{aligned}\mathbb{E}\big\{L\big(Q(\tau+1)\big)\big\} - \mathbb{E}\big\{L\big(Q(\tau)\big)\big\} + V\mathbb{E}\big\{E\big(\mathbb{I}(\tau)\big)\big\} \\ \le C + VE^* - \varepsilon\mathbb{E}\{Q(\tau)\}.\end{aligned}$$

We sum up the above inequality over $\tau \in \{0, 1, \cdots, T-1\}$ for some positive integer T and obtain as follows:

$$\begin{aligned}\mathbb{E}\big\{L\big(Q(t)\big)\big\} - \mathbb{E}\big\{L\big(Q(0)\big)\big\} + V\sum_{\tau=0}^{T-1}\mathbb{E}\big\{E\big(\mathbb{I}(\tau)\big)\big\} \le Ct + \\ VE^*T - \varepsilon\sum_{\tau=0}^{T-1}\mathbb{E}\{Q(\tau)\}.\end{aligned}$$

Since $\mathbb{E}\big\{L\big(Q(t)\big)\big\}$ and $\mathbb{E}\{Q(\tau)\}$ are non-negative, after eliminating one or both terms from the above inequality, we can get:

$$\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}\big\{E\big(\mathbb{I}(\tau)\big)\big\} \le E^* + \frac{C}{V} + \frac{\mathbb{E}\big\{L\big(Q(0)\big)\big\}}{Vt},$$

$$\begin{aligned}\frac{1}{t}\sum_{\tau=0}^{T-1}\mathbb{E}\{Q(\tau)\} \le \frac{C + V\Big[E^* - \frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}\big\{E\big(\mathbb{I}(\tau)\big)\big\}\Big]}{\varepsilon} \\ + \frac{\mathbb{E}\big\{L\big(Q(0)\big)\big\}}{\varepsilon}.\end{aligned}$$

Finally, we take limits as $t \to \infty$, and then Eqs. (28) and (29) are obtained. □

Remark: From Eqs. (28) and (29), it can be found that there exists an $[O(1/V), O(V)]$-tradeoff between average energy consumption and average response time (since the queue length is closely related to response time). The performance of the EEDTO algorithm depends on the control parameter V, which can be adjusted flexibly to reduce energy consumption by increasing the response time, or the other way around.

C. Computational Complexity Analysis

According to Algorithm 1, for the loop (line 2-10), EEDTO traverses all the application tasks once. Thus, each loop terminates in $O(N \cdot maxIter)$ operations, where $maxIter$ is the maximum number of iterations. For line 11, we apply 1-opt LS algorithm to reduce the exponential computational complexity, which is a local search algorithm that seeks an optimal solution around the initial estimate, whose vectors of candidates consist of all possible solutions with unitary Hamming distance [50]. Its computational complexity is in terms of run time $O(|\mathbb{D}|^3)$, which is polynomial.

Thus, the EEDTO algorithm for task offloading in the hybrid MEC and MCC environment can make offloading decisions with polynomial time complexity, which is relatively low than exponential.

V. Performance Evaluation

In this section, we will evaluate the performance of the proposed EEDTO algorithm in comparison with other offloading-decision schemes under various parameter changes.

A. Parameter Settings

In our simulation, a mobile blockchain network is considered with one IoT device (miner) and mining tasks, one MEC server and one MCC server. The computation capabilities of the IoT device, the MEC server and the MCC server are set to $f_{\text{IoT}} = 500$ MHz, $f_{\text{edge}} = 3$ GHz, and $f_{\text{cloud}} = 5$ GHz, respectively, which satisfy: $f_{\text{IoT}} < f_{\text{edge}} < f_{\text{cloud}}$ [6]. As an example, we consider the application on the IoT device consists of $N = 5$ tasks. We assume that $p_{\text{tr}} > p_{\text{ex}} > p_{\text{idle}}$ according to the measurements in [51], the transmission power of the IoT device, its power in processing and idle states are set as 1.3 W, 0.9 W and 0.3 W, respectively. In addition, we add one dummy node v_0 as task 0, and the workload for this node is 0. The data communication for each mining task is $D = \{10, 20, 4, 100, 200\}$ KB. The workload for each task is $\omega = \{300, 100, 20, 1000, 200\}$ G cycles. Our evaluation parameters and corresponding values are listed in Table III. To improve the reliability of the experiments, we run 10,000 times for each setting and then the results are averaged.

TABLE III: Evaluation Parameters

Evaluation Parameters	Value
The number of IoT application tasks	$N = 5$
The average violation rate	$\rho = 0.2$
The bandwidth of LAN	$B_{\text{lan}} \in (0, 1000)$ KB/s
The bandwidth of WAN	$B_{\text{wan}} \in (0, 500)$ KB/s
The delay of LAN	$T_{\text{lan}} = 0.5$ ms
The delay of WAN	$T_{\text{wan}} = 30$ ms
The computing frequency of IoT device	$f_{\text{IoT}} = 500$ MHz
The computing frequency of MEC server	$f_{\text{edge}} = 3$ GHz
The computing frequency of MCC server	$f_{\text{cloud}} = 5$ GHz
The idle power of IoT device	$p_{\text{idle}} = 0.3$ W
The CPU computation power of IoT device	$p_{\text{ex}} = 0.9$ W
The data transmission power of IoT device	$p_{\text{tr}} = 1.3$ W

B. Experimental Comparison

To reveal the effectiveness of the proposed EEDTO algorithm, the following offloading-decision approaches are implemented under the same MEC and MCC heterogeneous environment for comparison:

- *IoT-only scheme*: This can be treated as a zero offloading scheme. All computing tasks are executed locally on the IoT device.
- *Edge-only scheme*: This is a full offloading scheme. All computing tasks are fully offloaded to the MEC servers for further processing.
- *Cloud-only scheme*: This is a full offloading scheme. All computing tasks are fully offloaded to the MCC server for further processing.
- *LARAC scheme* [40], [52]: This is a partial offloading scheme with LAgrangian Relaxation based Aggregated Cost (LARAC) [53]. It is an energy-efficient offloading algorithm, aiming to find the optimal offloading solution that can minimize the mean energy consumption while satisfying the deadline constraint.
- *EEDTO scheme*: This is a partial offloading scheme based on the proposed EEDTO algorithm. In this method, we use the proposed Lyapunov optimization-based dynamic offloading-decision algorithm. This energy-efficient algorithm is designed to (asymptotically) minimize the energy consumption of a distributed computing system, while maintaining queue stability across the system.

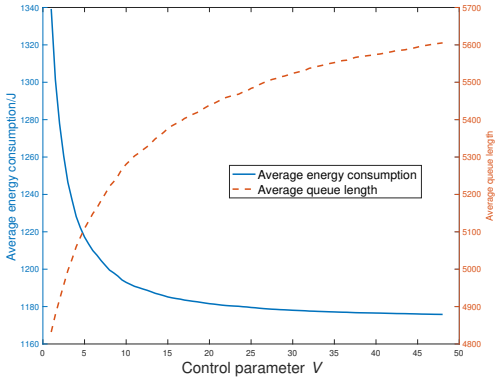
Actually, it's not always energy-efficient when offloading computing tasks from IoT devices to MEC/MCC servers. Offloading decisions regarding where to perform computation closely depend on whether the computational cost saved from task offloading covers the extra communication cost.

C. Effect of Tradeoff Parameter

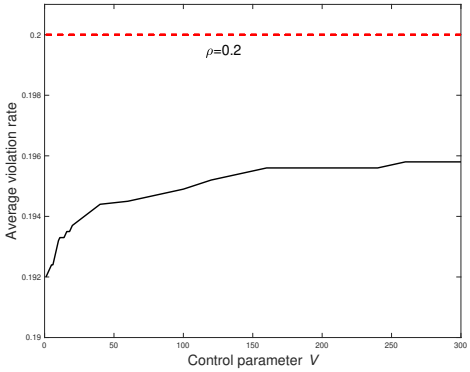
From Fig. 2(a), we can see that as the control parameter V rises, the average energy consumption drops dramatically at the beginning and then tends to the minimum value. However, the average queue length initially grows linearly and then converges gradually as V increases. By adjusting V, the *EEDTO scheme* can balance the average energy consumption and queue length, which confirms that an $[O(1/V), O(V)]$-tradeoff establishes between the average energy consumption and the average response time. This relationship is because V adjusts the weight of computation and communication costs in Lyapunov optimization. In Fig. 2(b), the average violation rate $\mathbb{E}\{\sigma(\mathbb{I}(t))\}$ increases gradually and approaches to a fixed ratio $\rho = 0.2$, denoted by the dotted red line. This finding satisfies the stable condition defined in Eq. (19), demonstrating that the queue length would be bounded. Thus, setting a larger deadline can reduce average energy consumption, but it will increase average response time.

As a result, by increasing V to a sufficiently large value, the proposed *EEDTO scheme* can approach to the optimal energy consumption and stabilize the queuing system state. This is because a larger V means prioritizing the optimization of energy consumption, and then the *Lyapunov* scheme would dynamically adjust offloading decisions to reduce average energy consumption. We can select a suitable V, where an increase in V will cause a small decrease in $\bar{Q}$. For example, we can choose $V = 20$ and beyond this value, increasing V brings marginal energy-saving, but will result in continuous growth of average response time.

Figure 3 demonstrates the performance comparison for different offloading-decision schemes. We can observe that the *edge-only scheme* costs less response time but consumes more energy than the *IoT-only scheme*. On the contrary, when compared with the *IoT-only scheme*, the proposed *EEDTO scheme* ($V = 20$) with heterogeneous edge-cloud servers can save around 60% energy consumption of IoT devices. This is because, unlike the *cloud-only scheme* and the *edge-only scheme* that ignore the relative relationship between communication and computational costs, the *EEDTO scheme* dynamically makes offloading decisions in a heterogeneous computing environment according to conditions such as task workloads, communication data and network bandwidths. Furthermore, when compared with the *LARAC scheme*, the *EEDTO scheme* also reduces more energy while only sacrificing a small portion of response time. Moreover, the computational complexity of the *LARAC scheme* is much higher than the *EEDTO scheme* [40]. Therefore, the proposed scheme is proved to be effective

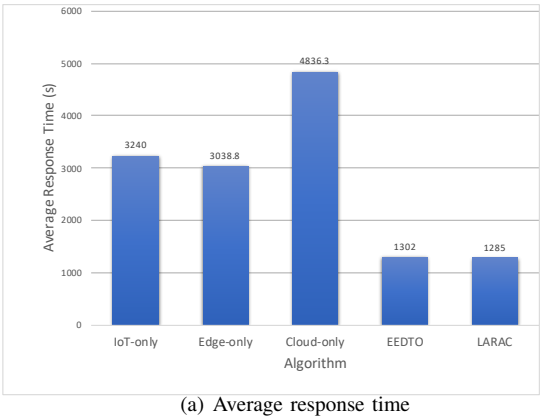


(a) Average energy consumption and queue length

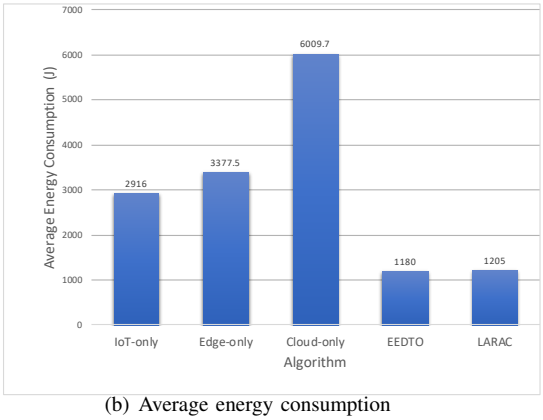


(b) Average violation rate

Fig. 2: Impact of V on average energy consumption, queue length and violation rate



(a) Average response time



(b) Average energy consumption

Fig. 3: Comparison of different offloading schemes

for the hybrid offloading decision-making problem with large-scale computing tasks, since it is much more energy-efficient with relatively lower computational complexity.

D. Effect of Computation-Intensive Tasks

In this case study, we will investigate the behavior of offloading techniques for tasks with various amounts of workload, which are divided into three groups, namely, *light* ($\omega/2$), *medium* (ω) and *heavy* (10ω), respectively.

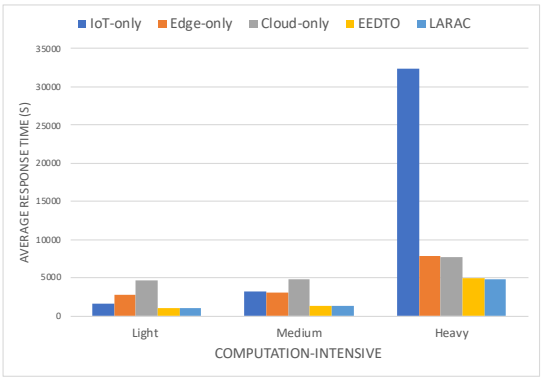
It can be observed from Fig. 4 that with the increase of the task workload, the average response time and energy consumption also arise. The reason is that when the computing capacities of the IoT device, the MEC server, and the MCC server remain unchanged, the increase in the amount of task workload will inevitably lead to longer execution time and larger energy consumption. More specifically, the cost for the *IoT-only scheme* arises dramatically as the task workload increases, owing to the fact the computing resources in IoT devices are very limited. For the *heavy* case with computation-intensive tasks, the *cloud-only scheme* achieves smaller response time costs as shown in Fig. 4(a) since it has the strongest computing capacity. However, it can be observed from Fig. 4(b) that the *cloud-only scheme* spends more energy for transmitting data compared with the *edge-only scheme* that has higher bandwidth.

In spite of different amounts of task workload, the cost obtained by the *EEDTO scheme* is always significantly less than those of the *local-only scheme*, the *cloud-only scheme*, and the *edge-only scheme*. The reason is that the *EEDTO scheme* can dynamically adjust the offloading decisions by migrating more computation-intensive tasks to the MEC/MCC server. Furthermore, Figure 4 demonstrates that the *EEDTO scheme* is robust enough to handle not only computation-heavy tasks that are suitable for computation offloading, but also tasks with smaller workloads.

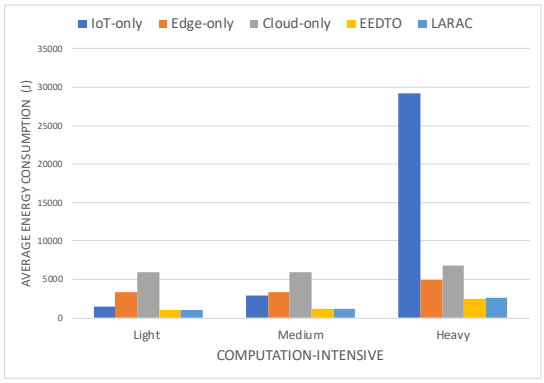
E. Effect of Communication-Heavy Tasks

In this case study, we investigate the behavior of offloading techniques for tasks with various amounts of communication data, which are divided into three groups, namely, *light* ($D/5$), *medium* (D) and *heavy* ($5D$), respectively.

Figure 5 demonstrates that as the communication data increases, the average response time and energy consumption are unchanged for *IoT-only scheme*. This is because, in the *IoT-only scheme*, all tasks in the IoT service are performed locally, there is no data upload / download process. However, for the *edge-only scheme*, the *cloud-only scheme* and the *EEDTO scheme*, the larger the amount of communication data of the task, the greater the average cost. This is because the amount of communication data of the task has a great influence on

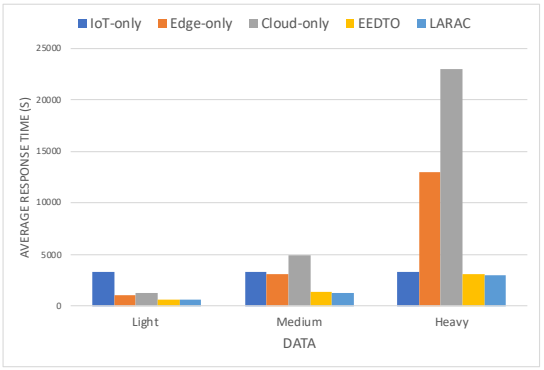


(a) Average response time

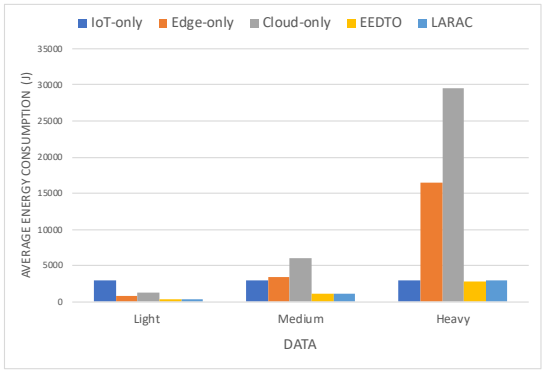


(b) Average energy consumption

Fig. 4: Comparison of different offloading schemes with computation-intensive tasks

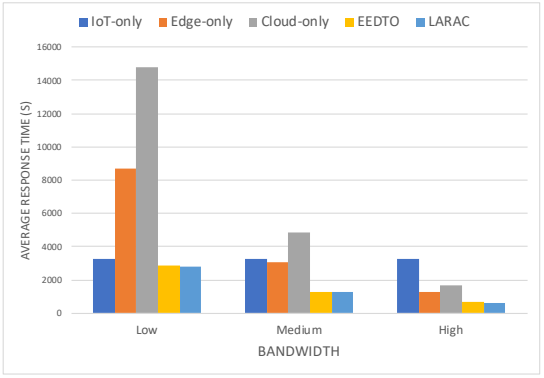


(a) Average response time

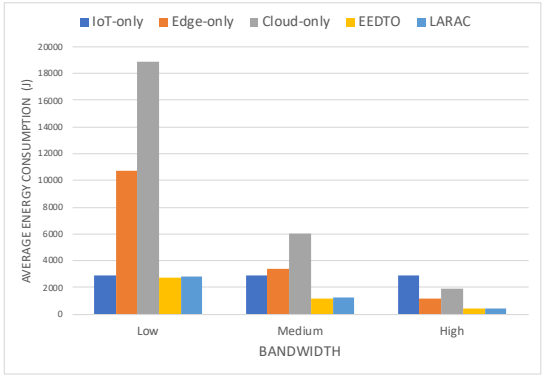


(b) Average energy consumption

Fig. 5: Comparison of different offloading schemes with communication-heavy tasks



(a) Average response time



(b) Average energy consumption

Fig. 6: Comparison of different offloading schemes with various bandwidths

the transmission delay and energy consumption in the process of task execution. Under the same offload strategy, larger data transfers take longer. Similarly, a larger data transmission volume will bring more transmission energy consumption to the IoT device.

As depicted in Fig. 5(a), the *EEDTO scheme* achieves the lowest response time among all the four offloading schemes regardless of different amounts of communication data. As shown in Fig. 5(b), offloading to the MEC/MCC server can also save energy for the *light* and *medium* cases. However, we cannot benefit from task offloading especially when D is very large because it will result in high transmission time. In this case, we would rather perform the computation locally on the IoT device than offload it to the MEC/MCC server, e.g., the *heavy* case with large communication data. This finding aligns well with the fact that applications suitable for computation offloading should not have heavy communication between tasks.

F. Effect of Network Bandwidths

In this case study, we will investigate the behavior of different offloading techniques for various bandwidth values.

Both the LAN and WAN bandwidths are divided into three groups, namely, *low* ($B_{\text{lan}}/4$ and $B_{\text{wan}}/4$), *medium* (B_{lan} and B_{wan}) and *high* ($4B_{\text{lan}}$ and $4B_{\text{wan}}$), respectively.

IoT users are vulnerable to dynamically changing network conditions due to their mobility, making it difficult to take high-quality offloading decisions in mobile environments [40]. Especially when the network bandwidth is very low, it may not be beneficial to offload tasks from IoT devices to MEC/MCC servers. Thus, the offloadable tasks are preferably transmitted via energy-efficient wireless channels so as to reduce transmission delay and save energy consumption.

Figure 6 demonstrates that network bandwidths affect the performance of offloading systems. With the increase of bandwidth, the average response time and energy consumption decrease, which means better task offloading gain in comparison to the *IoT-only scheme*. Moreover, in most cases, the *edge-only scheme* outperforms the *cloud-only scheme* since MEC servers are distributed at the proximity of IoT devices, they can be accessed with higher bandwidth and less latency. However, it fails to obtain the best possible outcome since the computing resources of MEC servers are still constrained when compared with MCC servers.

From Fig. 6, the proposed *EEDTO scheme* is superior to all other methods since it makes full use of the resources of MEC and MCC servers simultaneously, including the mobile network environments and the computing capacities, and hence it converges to the optimal solution quickly. In addition, the *EEDTO scheme* can save more energy and achieve lower computational complexity than the *LARAC scheme*, but with a small delay penalty.

VI. Conclusion and Future Work

In this paper, we identify and address key challenges of task offloading in blockchain-enabled heterogeneous IoT-Edge-Cloud computing environments, where MCC and MEC can work collaboratively to minimize the energy consumption of the IoT device with delay constraints. For deciding how to split and orchestrate the IoT applications across the edge and the cloud, we have formulated the offloading decision problem as an optimization problem and further derived an online and polynomial-time-complexity algorithm by taking advantage of the Lyapunov optimization technique, which determines whether and where to offload such that the energy consumption of the IoT device can be minimized when only sacrificing a little delay. The EEDTO scheme can dynamically offload application tasks to different places and balance the delay and energy consumption with an adjustable parameter V, allowing us to specify a combined optimization target by offering energy-efficient computing services. Through experiments, we further show that MCC can serve as a long-term data processor for computation-intensive tasks while MEC can be treated as a short-time data processor for delay-sensitive applications.

Although rigorous theoretical analysis and extensive numerical simulations have verified the efficiency of the proposed solution, validation based on real-world blockchain networks with intelligent offloading scenarios will be considered in the future.

Acknowledgment

This work is sponsored by National Natural Science Foundation of China (61801325, 62071327, 71701143), Natural Science Foundation of Tianjin City (18JCQNJC00600) and CCF-Tencent Open Research Fund.

References

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[2] L. Hu, Y. Tian, J. Yang, T. Taleb, L. Xiang, and Y. Hao, "Ready player one: UAV-clustering-based multi-task offloading for vehicular vr/ar gaming," *IEEE Network*, vol. 33, no. 3, pp. 42–48, 2019.

[3] N. H. Motlagh, M. Bagaa, and T. Taleb, "Energy and delay aware task assignment mechanism for UAV-based IoT platform," *IEEE Internet of Things Journal*, vol. 6, pp. 6523–6536, Aug. 2019.

[4] A. Asheralieva and D. Niyato, "Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8753–8769, 2019.

[5] K. Zhang, Y. Zhu, S. Maharjan, and Y. Zhang, "Edge intelligence and blockchain empowered 5G beyond for the industrial internet of things," *IEEE Network*, vol. 33, no. 5, pp. 12–19, 2019.

[6] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[7] K. Peng, H. Huang, S. Wan, and V. C. M. Leung, "End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment," *Wireless Networks*, June 2020.

[8] M. Li, Q. Wu, J. Zhu, R. Zheng, and M. Zhang, "A computing offloading game for mobile devices and edge cloud servers," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[9] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[10] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Energy efficient dynamic offloading in mobile edge computing for internet of things," *IEEE Transactions on Cloud Computing*, 2019.

[11] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[12] T. Meng, K. Wolter, H. Wu, and Q. Wang, "A secure and cost-efficient offloading policy for mobile cloud computing against timing attacks," *Pervasive and Mobile Computing*, vol. 45, pp. 4–18, Apr. 2018.

[13] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, pp. 2581–2593, Nov. 2020.

[14] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8099–8110, 2020.

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

[16] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, and W. Dou, "BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4187–4195, 2020.

[17] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud-edge-end in IoT: A blockchain-assisted collective Q-learning approach," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[18] H. Ko and S. Pack, "Distributed device-to-device offloading system: Design and performance optimization," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[19] Y. Han, D. Guo, W. Cai, X. Wang, and V. Leung, "Virtual machine placement optimization in mobile cloud gaming through QoE-oriented resource competition," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[20] X. Li, X. Wang, P.-J. Wan, Z. Han, and V. C. M. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 1768–1785, Aug. 2018.

[21] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, 2018.

[22] M. Goudarzi, H. Wu, M. S. Palaniswami, and R. Buyya, "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, 2020.

[23] H. Wu, W. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[24] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2020.

[25] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. P. C. Rodrigues, X. Kong, J. Huang, and R. Y. K. Kwok, "Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, pp. 1060–1072, Dec. 2019.

[26] S. Shen, Y. Han, X. Wang, and Y. Wang, "Computation offloading with multiple agents in edge-computing–supported IoT," *ACM Transactions on Sensor Networks*, vol. 16, pp. 1–27, Feb. 2020.

[27] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

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

[29] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[30] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4192–4203, 2019.

[31] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[32] W. Chen, Z. Zhang, Z. Hong, C. Chen, J. Wu, S. Maharjan, Z. Zheng, and Y. Zhang, "Cooperative and distributed computation offloading for blockchain-empowered industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8433–8446, 2019.

[33] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 6, p. 60, 2019.

[34] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, pp. 6214–6228, July 2020.

[35] Z. Zhang, Z. Hong, W. Chen, Z. Zheng, and X. Chen, "Joint computation offloading and coin loaning for blockchain-empowered mobile-edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9934–9950, 2019.

[36] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5G beyond," *IEEE Network*, vol. 33, no. 3, pp. 10–17, 2019.

[37] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, pp. 4660–4670, June 2019.

[38] Y. Jiao, P. Wang, D. Niyato, and K. Suankaewmanee, "Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, pp. 1975–1989, Sept. 2019.

[39] Z. Zhou, S. Yu, W. Chen, and X. Chen, "CE-IoT: Cost-effective cloud-edge resource provisioning for heterogeneous IoT applications," *IEEE Internet of Things Journal*, vol. 7, pp. 8600–8614, Sept. 2020.

[40] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 570–584, 2020.

[41] T. Zhao, S. Zhou, X. Guo, and Z. Niu, "Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2017.

[42] X. Wang, X. Li, S. Pack, Z. Han, and V. C. M. Leung, "STCS: Spatial-temporal collaborative sampling in flow-aware software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, pp. 999–1013, June 2020.

[43] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Secure computation offloading in blockchain based IoT networks with deep reinforcement learning," *arXiv preprint arXiv:1908.07466*, 2019.

[44] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 6, pp. 1991–1995, 2012.

[45] B.-G. Chun and P. Maniatis, "Dynamically partitioning applications between weak devices and clouds," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond*, p. 7, ACM, 2010.

[46] K. Peng, M. Zhu, Y. Zhang, L. Liu, J. Zhang, V. C. Leung, and L. Zheng, "An energy-and cost-aware computation offloading method for workflow applications in mobile edge computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 207, 2019.

[47] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "Optimized placement of scalable iot services in edge computing," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 189–197, 2019.

[48] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97–106, 2012.

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

[50] E. H. Aarts and J. K. Lenstra, *Local search in combinatorial optimization*. Princeton University Press, 2003.

[51] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?," *Computer*, no. 4, pp. 51–56, 2010.

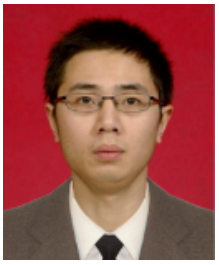
[52] V. Haghighi and N. S. Moayedian, "An offloading strategy in mobile cloud computing considering energy and delay constraints," *IEEE Access*, vol. 6, pp. 11849–11861, 2018.

[53] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the qos routing problem," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications*, vol. 2, pp. 859–868, IEEE, 2001.

Huaming Wu received the B.E. and M.S. degrees from Harbin Institute of Technology, China in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science at Freie Universität Berlin, Germany in 2015. He is currently an Associate Professor in the Center for Applied Mathematics, Tianjin University, China. His research interests include wireless networks, mobile edge computing, internet of things and deep learning.

Katinka Wolter received her PhD degree from Technische Universität Berlin in 1999. She has been Assistant professor at Humboldt-University Berlin and lecturer at Newcastle University before joining Freie Universität Berlin as a professor for dependable systems in 2012. Her research interests are model-based evaluation and improvement of dependability, security and performance of distributed systems and networks.

Pengfei Jiao received the Ph.D. degrees in computer science from Tianjin University, Tianjin, China, in 2018. He is a lecture with the Center of Biosafety Research and Strategy of Tianjin University. His current research interests include complex network analysis and data mining, and currently working on community detection and link predication, community evolution in dynamic networks, network embedding and applications of statistical network model.



Yingjun Deng received the B.S. in Applied Mathematics (2009) and M.S. in Computational Mathematics (2011) from Harbin Institute of Technology, China. He got his Ph.D. in Systems Optimization and Dependability from Troyes University of Technology, France in 2015. He became a lecturer since 2016 in the Center for Applied Mathematics, Tianjin University, China. His current research interests include applied statistics, deep learning, prognostic and health management, and predictive maintenance.

Yubin Zhao received the B.S. and M.S. degrees from the Beijing University of Posts and Telecommunications, China, in 2007 and 2010, respectively, and the Ph.D. degree in computer science from Freie Universität Berlin, Germany, in 2014. He is currently an Associate Professor in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His current research interests include wireless power transfer, indoor localization, and target tracking.

Minxian Xu is currently an assistant professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He received the B.S. and M.S. degrees in 2012 and 2015, respectively, both in software engineering from University of Electronic Science and Technology of China. He obtained his PhD degree from the University of Melbourne in 2019. His research interests include resource scheduling and optimization in cloud computing.