GreenEdge: Joint Green Energy Scheduling and Dynamic Task Offloading in Multi-Tier Edge Computing Systems

Huirong Ma, *Graduate Student Member, IEEE*, Peng Huang, *Graduate Student Member, IEEE*, Zhi Zhou, *Member, IEEE*, Xiaoxi Zhang, *Member, IEEE*, and Xu Chen, *Senior Member, IEEE*

***Abstract*—As mobile edge computing (MEC) emerges as a paradigm to meet the ever-increasing computation demands from real-time Internet of Things (IoT) applications in 5 G era, the development trends of which are mainly divided into two, with one being MEC with advanced computing architectures, and the other being MEC with high efficiency for sustainable operations. We are committed to taking advantage of these two trends to explore a novel multi-tier edge computing scenario with hierarchical task offloading and green energy provisioning via leveraging the energy harvesting (EH) technique. Specifically, we focus on the key problem of joint task offloading and energy scheduling in such green multi-tier edge computing systems. We aim to minimize the task execution cost by jointly considering the system cost that covers latency, energy consumption, and cloud rental fees. By formulating the problem as a stochastic optimization problem, we invoke the Lyapunov technique to decompose the long-term optimization problem into a series of one-slot optimization problems which only use the current system information. To solve the one-slot optimization problem which is a mixed-integer linear problem (MILP) proved to be *NP-hard*, we first relax the integer variables into real ones to obtain the optimal fractional solutions. Considering the capacity of the physical resources of each edge server, we propose a resource-constrained randomized dependent rounding algorithm to properly round up or down the fractional variables to get a feasible yet near-optimal solution. We conduct rigorous theoretical analysis and extensive simulations to verify the superior performance of the proposed schemes.**

***Index Terms*—Multi-tier edge computing, task offloading, green energy scheduling, randomized dependent rounding.**

I. Introduction

IN THE face of the proliferation of real-time IoT applications in 5 G era [1], vast computing capacity and energy supply are required. However, for devices, they usually have limited computing capacity and scant energy supply. Besides, offloading tasks to the cloud to execute may incur long end-to-end latency by the wide-area network (WAN) which is far beyond the stringent latency required by emerging IoT applications such as augmented reality [2]–[4] and interactive gaming [5]. As a remedy to the above limitations, the recently proposed technology, mobile edge computing (MEC) [6]–[9] can extend the cloud to the network edge thus to satisfy the stringent timeliness requirement and intensive computation demands of emerging IoT applications [10].

A typical MEC system consists of a set of geographically distributed edge servers which are deployed on the periphery of a network, providing elastic resources such as storage, computing, and network bandwidth [11]. Considering the distance between the edge servers and IoT devices, edge servers are often organized in a hierarchical fashion. Therefore, the resource-limited IoT devices which are heavily loaded can offload the tasks to nearby edge servers, thus to reduce the energy consumption and processing latency; meanwhile, in such a way, edge servers at the lower tier can flexibly share the stringent workload with the edge servers with greater capacities in the upper tier. With all the effectiveness of edge servers on reducing the energy consumption, resource requirement and perceived latency of the IoT devices, however, the low energy efficiency at the edge side may become the bottleneck towards sustainable edge computing. Besides, a major limitation of edge servers is that the computing capacity of edge servers is still significantly limited.

As a last consideration, 5 G technology involves the deployment of a large number of base stations (BSs), to increase the network coverage and provide higher throughput to the users. This however results in higher energy consumption, which is expected to considerably contribute to carbon emissions [12]. Taking China as an example, the annual network power consumption is expected to exceed 100 billion kWh. As thermal power prevails at present, this power consumption soar will result in 27.2 billion kilograms of carbon emissions. Therefore, there is an urgent need to increase the energy efficiency of edge servers and reduce the carbon emissions in edge tiers [13]. Thanks to the recent advancements of energy harvesting techniques [14], [15], off-grid green energy harvested from the environment (such as solar radiation, wind power generation, and kinetic human motion) is embraced as an important type of

Manuscript received October 23, 2021; revised December 24, 2021; accepted January 23, 2022. Date of publication January 31, 2022; date of current version May 2, 2022. This work was supported in part by the National Science Foundation of China under Grants U20A20159, 61972432 and 62102460, in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X355, and in part by Pearl River Talent Recruitment Program under Grant 2017GC010465. The review of this article was coordinated by Dr. Zehui Xiong. *(Corresponding author: Xu Chen.)*

The authors are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, Guangdong 510006, China (e-mail: mahr3@mail2.sysu.edu.cn; huangp57@mail2.sysu.edu.cn; zhouzhi9@mail.sysu.edu.cn; zhangxx89@mail.sysu.edu.cn; chenxu35@mail.sysu.edu.cn).

Digital Object Identifier 10.1109/TVT.2022.3147027

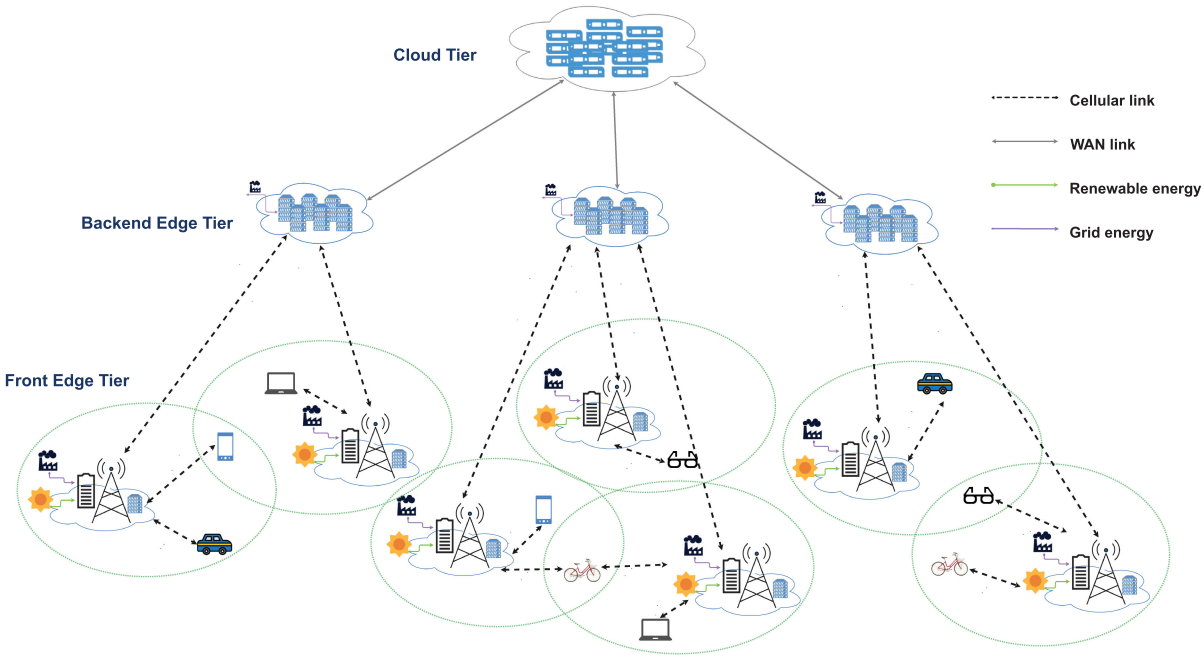


Fig. 1. An illustration of the multi-tier edge computing system with green energy and grid energy.

energy supply for the edge servers in edge tiers in the field. In addition, the emergence of energy harvesting technique helps to achieve green computing [16]. As shown in Fig. 1, in our paper, the edge servers in front edge tier can not only harvest the green energy but also use the grid power to supply energy such as the smart street light system [17] [18] in the smart city which employs green energy powered street lighting as the IoT network communication node device. Despite the advantages, the randomness and suddenness of the arrival of green energy introduce new challenges for fully reaping the benefits of edge servers in edge tiers to achieve green computing.

Therefore, to embrace the profound benefits enabled by MEC-EH in multi-tier edge computing scenario, a task provider needs to address the following major issues jointly: 1) task offloading, which determines where to offload the tasks subject to the computing capacity of edge servers in different edge tiers; and 2) green energy scheduling, how much green energy is used to compute the tasks to minimize the system cost that covers latency, energy consumption, and cloud rental fees. The timely decision making regarding these two issues is critical but challenging, due to temporal variations of system dynamics in a wireless environment, uncertainty in the resulting offloading costs, and the randomness of the arrival of green energy.

In this paper, we address the challenge of incorporating green energy into edge servers and focus on the joint task offloading and green energy scheduling problem for multi-tiered edge systems. We propose an efficient approximated optimization framework, which can make online decisions for our joint optimization problem without the need to access future information and has a good theoretical performance guarantee. We summarize the key results and main contributions of our paper as follows:

- *Problem Formulation:* We formulate the problem of joint task offloading and green energy scheduling in the multi-tier edge computing system as a stochastic optimization problem to increase the energy efficiency of edge servers and reduce the operational cost. We introduce harvested energy into the edge servers when managing a multi-tier edge computing system and consider minimizing the system cost that covers latency, energy consumption, and cloud rental fees. In addition, our objective function is more general than those proposed by many related works in that we include various types of energy consumption of two tiers of edge servers (including green energy and fossil energy), the cloud rental fees, and the latency within the edge servers.
- *Online Algorithm Design:* We propose a novel online algorithm framework that nicely integrates the Lyapunov optimization technique [19] and an efficient optimization rounding algorithm. More specifically, through a non-trivial transformation, we decouple the problem into a series of one-slot online decision problems which do not require the future system states as a prior. However, the decoupled one-slot problem is a mixed-integer linear programming (MILP) problem which is proved to be NP-hard. To efficiently solve it, we further relax the problem, and then use a novel resource-constrained randomized dependent rounding (RCRDR) algorithm to convert the fractional solution into a feasible and near-optimal solution of the original problem.
- *Theoretical Analysis and Experimental Verification:* We conduct theoretical analysis and trace-driven simulations to evaluate the performance of our proposed algorithms. The results show that our algorithms achieve a tunable trade-off between system minimizing long-term costs and the queue length while effectively reducing the costs. Besides, through extensive performance evaluations, we show that our sub-routine algorithm (RCRDR) significantly outperforms existing optimization rounding algorithms, and the overall scheme can not only achieve superior performance gain over existing benchmarks but is also very close to

the offline optimum with at most 4% higher system cost incurred.

The rest of the paper is organized as follows. We discuss the related work in Section II and present the system model in Section III. Then we show the online algorithms in Section IV and the performance analysis of our algorithms in Section V. Section VI evaluates the performance of our proposed algorithms via extensive simulations, while Section VII concludes this paper.

II. Related Work

The computing latency minimizing and the energy consumption minimizing for mobile edge computing systems have attracted significant attention in recent years.

Many researchers studying mobile edge computing (MEC) have devoted themselves to solving the problem of computing efficiency maximizing and the IoT devices' energy consumption minimizing [20]–[24] in the past few years. The authors in [20] mainly study the energy efficiency maximization problem of the MEC systems by combing local computing and data offloading into a joint computation algorithm by using a new computing efficiency metric. To solve the problem of computation offloading and resource allocation in a multi-user wireless powered MEC system, [21] proposes an iterative algorithm by using a Lagrangian dual method and transformed the offloaded delay into an offloaded data rate constraint. A unified mobile-edge computing (MEC) and wireless power transfer (WPT) design framework have been proposed in [22], where the proposed unified design framework MEC-WPT can pave the way to facilitate ubiquitous computing for IoT devices in a self-sustainable way. To jointly find optimal caching and offloading scheme, authors in [23] exploited resource-based utility function and device-number-based relative distance, which helps to effectively reduce the average service latency and keep a low average leasing cost. To achieve long-term performance for cooperative computation offloading, authors in [24] propose a multiagent deep reinforcement learning (DRL) framework in which a scatter network is adopted to improve its stability and league learning is introduced for agents to explore the environment collaboratively for fast convergence and robustness. The above researches [20]–[24] focus on the energy efficiency of the IoT devices while this paper focuses on that of MEC server.

With the introduction of green computing [16] and the advancements of energy harvesting technology [14], [15], the researchers study the IoT devices equipped with energy harvested module [25]–[27]. Authors in [25] model the computation offloading process as a Markov decision process (MDP) to cope with the challenge of the unpredictability nature of the generated data and the harvested energy. The work [26] proposes a deep reinforcement learning (DRL) based offloading scheme for an IoT device with EH and a DRL based offloading scheme to further accelerate the learning speed. A green MEC system with EH devices and an efficient computation offloading strategy are investigated in [27], in which the execution latency and task failure are adopted as the performance metric. However, these algorithms are not suitable for our proposed EH MEC environment where the edge servers in the front edge tier are equipped with energy harvesting module, i.e., previous studies aimed at the sustainability of IoT devices while our proposed scheme aims at the stable operation of edge server in energy harvesting environment.

Definitely, the researches of MEC systems with energy harvesting module also has attracted great attention from researches [28], [29]. Authors in [28] address the dynamic workload offloading and the edge server provisioning of the energy harvesting MEC system by proposing an efficient reinforcement learning-based resource management algorithm. This algorithm uses a decomposition of the (offline) value iteration and (online) reinforcement learning, thus achieving a significant improvement of learning rate and run-time performance. By integrating renewable energy into the MEC system, authors of [29] propose an offloading scheduling algorithm that considers the MEC battery and quality of service experience (QoE). The algorithm in [29] includes the admission control of the offload request and the calculation of the MEC server frequency. However, none of the above algorithms are suitable for our proposed scheme that incorporates harvested energy into edge servers and focuses on the joint task offloading and green energy scheduling problem for multi-tiered edge systems. Moreover, different from such works that focus on edge computing systems with flat or two-tiered architectures, our solution is applicable to general multi-tiered edge computing systems with time-varying wireless channel states and the randomness and suddenness of the arrival of green energy.

III. System Model

We consider a multi-tiered edge computing system, as shown in Fig. 1. We assume that time is slotted, and denote the time slot length and the time slot index set by τ_0 and $\mathcal{T} \triangleq \{0, 1, \ldots, T-1\}$, respectively. In Fig. 1, inside the front edge tier (FET) are a set of front EH-edge servers (FESs-EH) that offer low latency access to a set of wireless IoT devices. Each FES-EH is equipped with an EH module to collect green energy and store it in its battery. On the other hand, the backend edge tier (BET) comprises a set of backend edge servers (BESs).

In our paper, we mainly focus on the energy consumption of edge servers (in both FET and BET), the cloud rental fees, and communication latency within the edge servers (in both FET and BET) and cloud server. First, the energy (including both green energy and grid power energy, i.e., fossil energy) consumption we consider include two parts: processing energy and transmission energy. The processing energy is induced by the task processing on FET and BET. In practice, data transmissions from FESs-EH to BESs and from BESs to the cloud can incur energy costs. However, the transmission energy from FESs-EH to BESs is much smaller comparing to the processing energy so we ignore it. Considering that we assume the BESs communicate with the cloud through wireline connections, so we ignore the transmission energy from BESs to the cloud. Second, the cloud rental fees we consider only include the task's processing costs in the cloud server. Third, the latency we consider include two parts: processing latency and transmission latency. We focus on the transmission latency from IoT devices to FESs-EH, to BESs, and to cloud server. And we assume that the task processing in

TABLE I
MAIN PARAMETERS AND THEIR DEFINITIONS

Parameter	Definition
$\mathcal{T}$, τ_0	the set of time slots, the length of each time slot
$\mathcal{F}$, $\mathcal{B}$, $\mathcal{N}$	the set of FET, BET, tasks
λ_i^e, λ_i^t	$\in$ [0,1], means the weighting parameters of energy consumption costs and the cloud rental fees, or means the weighting parameters of latency
V	Lyapunov control parameter
$I_i(t)$, $O_i(t)$	the input and output size of the task of device i
$Y_i(t)$	the amount of computing resource required for the task of device i
$\pi_i(t)$	that is 1 if device i has a task to be offloaded, and 0 otherwise
$P_f^{Front}(t)$	the physical resources (in CPU cycles) of FES-EH f
$P_b^{Back}(t)$	the physical resources (in CPU cycles) of BES b
$D_{if}^{Fronttra}(t)$	the upload data rates from device i to its associated FES-EH f
$D_{if}^{Frontrec}(t)$	the download data rates from device i to its associated FES-EH f
$D_{ib}^{Backtra}(t)$	the upload data rates from device i to its associated BES b
$D_{ib}^{Backrec}(t)$	the download data rates from device i to its associated BES b
$D_i^{Cloudtra}(t)$	the upload data rates from device i to cloud server
$D_i^{Cloudrec}(t)$	the download data rates from device i to cloud server
$e_f^{Front}(t)$	the energy consumption in FES-EH f generated by receiving one unit of input data, calculating the data and transmitting the calculated result data back to device i
$e_b^{Back}(t)$	the energy consumption in BES b generated by receiving and calculating one unit of data, and transmitting the calculated result data back to device i
c^{cloud}	the resource rental fees (costs) for receiving one unit of input data, calculating the data and transmitting the result data back to device i
$p_f^{Front}(t)$	the electricity price of FES-EH f
$p_b^{Back}(t)$	the electricity price of BES b
α	the cost coefficient of using green energy
β	the cost coefficient of using grid power energy (fossil energy)
$Q_f(t)$	the energy level of the EH module at time slot t
$r_f(t)$	the amount of harvested green energy in FES-EH f at time slot t

Variable	Definition
$x_{ij}(t)$	the task offloading decision making at time slot t
$E_f^{green}(t)$	the consumed green energy in FES-EH f

each time slot can be completed by the end of the same time slot, thus we ignore the processing latency. Particularly, the task can be divided into multiple sub-tasks which can be finished during each time slot. Table I summarizes the key notations used throughout this paper.

A. Basic Resource Model

In the following, we first illustrate the basic settings.

- *IoT Device Resource Model:* The edge system consists of N IoT devices, and each IoT device $i \in \mathcal{N}$ can establish a cellular link with its associated base station. During a time slot, the locations of devices are assumed to be unchanged and the wireless channels are assumed to be stable. However, considering the mobility of users, they may change across different time slots.
- *Mobile Task Model:* To characterize the task of a device $i \in \mathcal{N}$, for each time slot $t \in \mathcal{T}$, we adopt a parameter tuple $< I_i(t), Y_i(t), O_i(t) >$, in which $I_i(t)$ is the data size of task, $Y_i(t)$ is the amount of computing resource (i.e., the number of CPU cycles) required when processing the task, and $O_i(t)$ indicates the output data size of the task.
- *FES-EH Resource Model:* The edge computing system consists of F FESs-EH in FET, let $\mathcal{F} = \{1, 2, \ldots, F\}$ be the sets of FESs-EH. Each FES-EH $f \in \mathcal{F}$ has the dynamic and restricted amount of physical resources $P_f^{Front}(t)$ (in CPU cycles) in a time slot. We use $D_{if}^{Fronttra}(t)$ and $D_{if}^{Frontrec}(t)$ to denote the corresponding upload and download data rates from device i to its associated FES-EH f respectively.
- *BES Resource Model:* The edge computing system consists of B BESs in BET. Let $\mathcal{B} = \{1, 2, \ldots, B\}$ be the sets of BESs. Each BES $b \in \mathcal{B}$ has the dynamic and restricted amount of physical resources $P_b^{Back}(t)$ (in CPU cycles) in a time slot. $D_{ib}^{Backtra}(t)$ and $D_{ib}^{Backrec}(t)$ are used to denote the corresponding upload and download data rates from device i to its associated BES b respectively.
- *Cloud Resource Model:* We assumed that the operator will deploy a cloud server at the network to further facilitate the execution of tasks. Accordingly, we use $D_i^{Cloudtra}(t)$ and $D_i^{Cloudrec}(t)$ to denote the corresponding upload and download data rates from device i to cloud server respectively.

B. Task Execution Model

The task execution model is introduced as follows. As described above, we assume that at each time slot $t \in \mathcal{T}$, each device will generate a task that needs to be offloaded and executed.

- *FES-EH Offloaded Execution:* A device i can offload its own task to FESs-EH via the cellular link. In this case, we use $e_f^{Front}(t)$ ($f \in \mathcal{F}$) to represent the energy consumption in FES-EH f generated by receiving one unit of input data, calculating the data and transmitting the calculated result data back to device i. Given the amount of input data that device i offloads to FES-EH f, the energy consumption is given by $E_{if}^{Front}(t) = I_i(t) * e_f^{Front}(t)$. And the transmission latency is given by $T_{if}^{Front}(t) = I_i(t)/D_{if}^{Fronttra}(t) + O_i(t)/D_{if}^{Frontrec}(t)$.
- *BES Offloaded Execution:* The task of device i on each FES-EH can be offloaded to and computed by any of its accessible BESs. In this case, we use $e_b^{Back}(t)$ ($b \in \mathcal{B}$) to represent the energy consumption in BES b generated by receiving one unit of input data, calculating the data and transmitting the calculated result data back to device i. Given the amount of input data that device i offloads to BES b, the energy consumption is given by $E_{ib}^{Back}(t) = I_i(t) * e_b^{Back}(t)$. And the transmission latency is given by $T_{ib}^{Back}(t) = I_i(t)/D_{ib}^{Backtra}(t) + O_i(t)/D_{ib}^{Backrec}(t)$.
- *Cloud Offloaded Execution:* The task of device i on each BES can be offloaded to the cloud. In this case, we use

c^{cloud} to represent the cost of receiving one unit of input data, calculating the data and transmitting the calculated result data back to device i. Considering the bandwidth usage cost and the performance cost of the WAN is high, the cost in cloud server is typically more expensive than EFSs-EH and BESs. Given the amount of input data that device i offloads to the cloud server, the resource rental cost in time slot t of the cloud server can be expressed as: $C_i^{Cloud}(t) = I_i(t) * c^{cloud}$, and the transmission latency is given by $T_i^{Cloud}(t) = I_i(t)/D_i^{Cloudtra}(t) + O_i(t)/D_i^{Cloudrec}(t)$.

C. Task Offloading Constraints

Next, we will formulate the constraints for the task offloading problem. Specifically, for each time slot $t \in \mathcal{T}$, let $\pi_i(t)$ $(\forall i \in \mathcal{N})$ be a binary indicator that is 1 if device i has a task to be offloaded, and 0 otherwise (i.e., the task queue of device i is empty). In addition, we denote a binary vector $x_i(t) = [x_{i1}(t), \ldots, x_{ii}(t), \ldots, x_{iF}(t), x_{iF+1}(t), x_{iF+2}(t), \ldots, x_{iF+B}(t), x_{iF+B+1}(t)]$ for each device i to denote the task offloading decision making at time slot t. Specifically, when $j \in \{1, 2, \ldots, F\}$ and $x_{ij}(t) = 1$, it represents the task is executed on the FET; when $j \in \{F+1, F+2, \ldots, F+B\}$ and $x_{ij}(t) = 1$, it represents the task is executed on the BET; otherwise, $j = F+B+1$ and $x_{ij}(t) = 1$, it represents the task is executed on the cloud. Note that at each time slot, the task of device i is assigned to one and only one execution node. We then have the task offloading constraints as follow:

$$x_{ij}(t) \in \{0, 1\}, \forall i \in \mathcal{N}, \forall j \in \{1, 2, \ldots, F+B+1\}. \tag{1}$$

$$\sum_{i=1}^{N} x_{ij}(t) Y_i(t) \leq P_j^{Front}(t), \forall j \in \{1, 2, \ldots, F\}. \tag{2}$$

$$\sum_{i=1}^{N} x_{ij}(t) Y_i(t) \leq P_j^{Back}(t), \forall j \in \{F{+}1, F+2, \ldots, F{+}B\}. \tag{3}$$

$$\sum_{j \in \{1,2,\ldots,F+B+1\}} x_{ij}(t) = \pi_i(t), \forall i \in \mathcal{N}. \tag{4}$$

The constraint (2) represents that the physical resources in the FES-EH j ($j \in \{1, 2, \ldots, F\}$) is restricted by $P_j^{Front}(t)$. The constraint (3) represents that the physical resources in the BES j ($j \in \{F+1, F+2, \ldots, F+B\}$) is restricted by $P_j^{Back}(t)$. The constraint (4) represents that during a task offloading round a device will offload at most one task.

D. FESs-EH Battery Model and Green Energy Constraints

For the EH battery model, we assume that each FESs-EH in FET is equipped with an EH module (which can be a solar panel, a wind turbine, a kinetic tile, or a combination of them) as shown in Fig. 1. Before powering FES-EH, the harvested energy is buffered at battery to ensure sustained energy supply, since the ambient renewable energy sources appear to be random and bursty in nature.

For each FES-EH $f \in \{1, 2, \ldots, F\}$, we use $Q_f(t)$ to denote the energy level of the EH module at the beginning of time slot t. Besides, $r_f(t)$ is denoted as the amount of harvested green energy at time slot t. Note that $r_f(t)$ is a time-varying and random variable depending on the ambient environment. And the green energy $r_f(t)$ harvested by the EH module of FES-EH f is subjected to the availability constraint

$$r_f(t) \in [0, r_f^{max}], \tag{5}$$

where r_f^{max} denotes the upper-bound of the possible amount of green energy that can be harvested by FES-EH f at each time slot t.

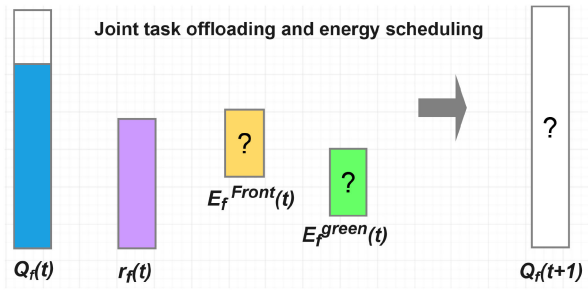


Fig. 2. Battery state dynamics. With unknown value of $E_f^{Front}(t)$ and $E_f^{green}(t)$, therefore the system needs to solve the joint task offloading and energy scheduling problem.

For realistic considerations, the collected green energy is not the sole source of energy for each FES-EH's battery which also can use the grid electricity (fossil energy) when the harvested energy cannot fulfill the task executions' needed energy. And we use the variable $E_f^{green}(t)$ to denote the harvested green energy used to execute the tasks in FES-EH f which is also an energy scheduling decision. Therefore, we have

$$E_f^{green}(t) \geq 0, \forall f \in \{1, 2, \ldots, F\}. \tag{6}$$

Besides, in FES-EH f, the needed green energy for execution does not exceed the total energy consumption (i.e., $E_f^{Front}(t)$, where we have $E_f^{Front}(t) = \sum_{i=1}^{N} x_{if}(t) E_{if}^{Front}(t)$):

$$E_f^{green}(t) \leq E_f^{Front}(t), \forall f \in \{1, 2, \ldots, F\}. \tag{7}$$

Considering the energy level of the EH module at time slot t, we further have:

$$E_f^{green}(t) \leq Q_f(t), \forall f \in \{1, 2, \ldots, F\}. \tag{8}$$

Thus, the dynamics of the renewable energy level of each EH module in FES-EH f follows:

$$Q_f(t+1) = Q_f(t) - E_f^{green}(t) + r_f(t), \forall f \in \{1, 2, \ldots, F\}, \tag{9}$$

where $r_f(t)$ can be viewed as the "arrivals" of the energy queue $Q_f(t)$, and $E_f^{green}(t)$ can be viewed as the "service rate" of such a queue. Note that, if the harvested green energy cannot be fully utilized, the virtual energy queue may accumulate and even grow indefinitely over time. To avoid this, we must keep the queue stable, i.e., $\lim_{T \to \infty} \mathbb{E}\{Q_f(T)\}/T = 0$ $(Q_f(T) < \infty)$. The FESs-EH battery state dynamics is illustrated in Fig. 2.

E. System Cost Minimization Problem

In this paper, we assume that for each FES-EH $f \in \mathcal{F}$, the cost coefficient of using green energy is α and the cost coefficient of using grid power energy (fossil energy) is β. Note that, α is smaller than β. Besides, we use λ_i^e, $\lambda_i^t \in \{0, 1\}$ to denote the weighting parameters of the costs of energy consumption and the cloud rental fees, and the execution latency for devices' decisions making, respectively.

Therefore, for each time slot $t \in \mathcal{T}$, the FESs-EH' costs caused by executed tasks of all IoT devices including transmission latency costs and energy costs in FES-EH f can be denoted as $C^{Front}(t) = \sum_{i=1}^{N}\sum_{f=1}^{F} x_{if}(t)\lambda_i^t T_{if}^{Front}(t) + \sum_{f=1}^{F}\lambda_i^e p_f^{Front}(t)[\alpha E_f^{green}(t) + \beta(E_f^{Front}(t) - E_f^{green}(t))]$; the BES' costs caused by executed all IoT devices' tasks including transmission latency costs and energy costs in BES b can be denoted as $C^{Back}(t) = \sum_{i=1}^{N}\sum_{b=F+1}^{F+B} x_{ib}(t)[\lambda_i^t T_{ib}^{Back}(t) + \lambda_i^e p_b^{Back}(t) E_{ib}^{Back}(t)]$; the cloud costs caused by executed all IoT devices' tasks include transmission latency costs and cloud resource rental costs, and can be denoted as $C^{Cloud}(t) = \sum_{i=1,c=F+B+1}^{N} x_{ic}(t)[\lambda_i^t T_i^{Cloud}(t) + \lambda_i^e C_i^{Cloud}(t)]$.

Note that with the capability of energy harvesting, our objective is to make joint task offloading and green energy scheduling decisions to minimize the system total costs, which can be formulated as:

$$\min \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[C^{Front}(t) + C^{Back}(t) + C^{Cloud}(t)] \quad \text{(P1)}$$
$$s.t. \quad (1) - (9).$$

IV. Algorithm Design

To solve the problem $\mathcal{P}1$, a direct way is to gradually optimize each individual time slot to achieve the optimization goal of minimizing the total system costs. However, considering the dynamic coupling between the EH module buffer of each FES-EH in different time slots as well as the dynamic system states (such as the arrival of available green energy and the connections between devices and each FES-EH), it is not realistic to solve problem $\mathcal{P}1$ offline. Therefore, we devote to propose an online approach that can efficiently make task offloading and green energy scheduling decisions on-the-fly without foreseeing the future. We first apply the Lyapunov optimization technique [19] to decouple problem $\mathcal{P}1$ into one-slot deterministic MILP subproblem $\mathcal{P}2$. Unfortunately, the resulted decomposed subproblem $\mathcal{P}2$ in each time slot is NP-hard. As a result, the standard Lyapunov optimization approach relying on getting optimal solution from the decomposed problems fails to work. Hence, we will develop an approximated Lyapunov optimization scheme to overcome this key issue. Specifically, we further relax the integer constraint to get problem $\mathcal{P}3$, and by using the simplex method [30], we get the optimal fractional solution and finally propose an efficient resource-constrained randomized dependent rounding RCRDR algorithm to get a feasible yet near-optimal solution. The proposed optimization algorithm workflow is summarized in Fig. 3.

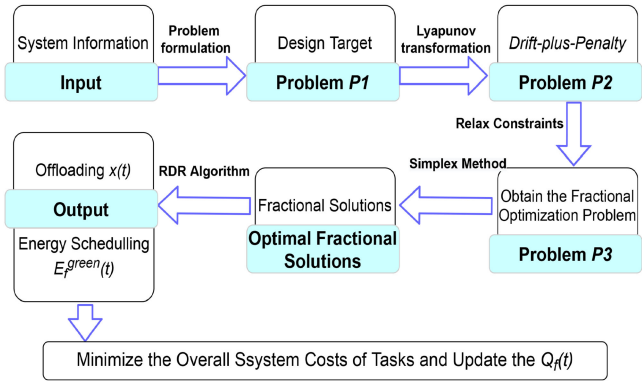


Fig. 3. The summary of the proposed optimization process.

A. Problem Transformation Via Lyapunov Optimization

To proceed, we define the following key concepts in Lyapunov optimization tailored to the formulated problem P1.

1) Queue Stability: In order to stabilize the virtual queue (i.e., $Q_f(T) < \infty$), the quadratic Lyapunov drift function is defined as:

$$L(\Theta(t)) \triangleq \frac{1}{2}\sum_{f=1}^{F} Q_f(t)^2, \quad (10)$$

here $\Theta(t) \triangleq < Q_f(t) >|_{f\in\mathcal{F}}$, the vector of battery queue backlogs on time slot t. We further define the conditional Lyapunov drift, $\Delta(\Theta(t))$ as the expected change in the Lyapunov function from one time slot to the next:

$$\Delta(\Theta(t)) \triangleq \mathbb{E}[L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)]. \quad (11)$$

By the Lyapunov drift theorem [19], if a policy minimizes the Lyapunov drift in each time slot, then all queue backlogs are consistently pushed towards a small size, which potentially maintains the stabilities of all queues.

2) Joint Queue Stability and System Costs Minimization: We next consider the joint queue stability and the objective optimization (i.e., minimizing the time-average costs for task execution of all IoT devices). Following the Lyapunov optimization framework [19], this is equivalent to minimize an upper bound of the following drift-plus-penalty function:

$$\Delta(\Theta(t)) + V\mathbb{E}[C^{Front}(t) + C^{Back}(t) + C^{Cloud}(t)|\Theta(t)],$$

where V is a non-negative control parameter that achieves a desirable tradeoff between the optimality and queue backlogs. The following Lemma provides an upper bound of the drift-plus-penalty function.

Lemma 1: For all possible values of $\Delta(\Theta(t))$ and under any task offloading policy $x(t)$ and green energy scheduling decision $E(t)$ for all time slot t, the drift-plus-penalty function satisfies:

$$\Delta(\Theta(t)) + V\mathbb{E}[C^{Front}(t) + C^{Back}(t) + C^{Cloud}(t)|\Theta(t)] \leq B + y(t)+$$

$$\sum_{i=1}^{N}\sum_{f=1}^{F} x_{if}(t)w_{if}(t) + \sum_{i=1}^{N}\sum_{b=F+1}^{F+B} x_{ib}(t)w_{ib}(t) + \sum_{i=1,c=F+B+1}^{N} x_{ic}(t)w_{ic}(t) + \sum_{f=1}^{F}\sum_{i=1}^{N} x_{if}(t)w_{if'}(t) + \sum_{f=1}^{F} E_f^{green}(t)w_f(t),$$

with
$$w_{if}(t) = V\lambda_i^t T_{if}^{Front}(t),$$
$$w_{ib}(t) = V[\lambda_i^t T_{ib}^{Back}(t) + \lambda_i^e p_b^{Back}(t) E_{ib}^{Back}(t)],$$
$$w_{ic}(t) = V[\lambda_i^t T_i^{Cloud}(t) + \lambda_i^e C_i^{Cloud}(t)],$$
$$w_{if'}(t) = V\beta\lambda_i^e p_f^{Front}(t) E_{if}^{Front}(t),$$
$$w_f(t) = V\lambda_i^e p_f^{Front}(t)[\alpha - \beta] - Q_f(t). \quad (12)$$

Here $B = 1/2\sum_{f=1}^{F}((E_f^{max})^2 + (r_f^{max})^2)$ is a constant value across all time slots, $E_f^{max} = \max_t E_f^{green}(t)$ and $y(t) = \sum_{f=1}^{F} Q_f(t)r_f(t)$ involves no offloading indicators. Therefore, we do not consider them in the joint task offloading and green energy scheduling algorithm design. Due to the page limit, their specific expressions and the detailed proof is given in our online technical report [31].

B. Online Algorithm for P1

According to the transformed problem and Lemma 1, the key technique of approximated Lyapunov optimization is consistently minimizing the drift-plus-penalty upper bound (i.e., the last five terms on the right-hand side) of (12), subject to constraints (1)-(9), which can be approved to achieve a good performance for the problem in P1. Now, we rearrange the formulations in (12) for an unfold form $\mathcal{P}2$, and then propose the online Joint Task Offloading and Energy Scheduling (JTOES) algorithm for solving $\mathcal{P}1$.

$$\mathcal{P}2:$$
$$\min_{x(t),E(t)} \sum_{i=1}^{N}\sum_{f=1}^{F} x_{if}(t)w_{if}(t) + \sum_{i=1}^{N}\sum_{b=F+1}^{F+B} x_{ib}(t)w_{ib}(t) + \sum_{i=1,c=F+B+1}^{N} x_{ic}(t)w_{ic}(t) + \sum_{f=1}^{F}\sum_{i=1}^{N} x_{if}(t)w_{if'}(t) + \sum_{f=1}^{F} E_f^{green}(t)w_f(t)$$
$$s.t. \quad \text{constraints} \quad (1)-(9). \quad (13)$$

The detailed performance analysis of Algorithm 1 would be shown in section V. Unfortunately, even if we get the one slot minimization problem $\mathcal{P}2$, it still is *NP-hard* and intractable to find the optimal solutions. We will prove the *NP-hardness* and then apply a relaxation and randomized dependent rounding technique to obtain a near-optimal solutions for $\mathcal{P}2$.

Algorithm 1: On-line JTOES algorithm for solving $\mathcal{P}1$.

Input: $Q_f(0)$, V, r_f^{max}, $\pi_i(t)$, $P_f^{Front}(t)$, $P_b^{Back}(t)$;
Output: x_{ij}, $E_f^{green}(t)$;
1: **for** each time slot $t \in \{0, 1, 2, \ldots, T-1\}$ **do**
2: Obtain $x_{if}(t)$, $x_{ib}(t)$, x_{ic} and $E_f^{green}(t)$ by solving $\mathcal{P}2$.
3: $Q_f(t+1) = Q_f(t) - E_f^{green}(t) + r_f(t)$.
4: **end for**

C. Approximate Algorithm for $\mathcal{P}2$

Due to the constraint (1)-(4) and (6), $\mathcal{P}2$ is a mixed integer linear programming (MILP) problem which can be proved to be $NP-hard$ in Theorem 1.

Theorem 1: The optimization problem $\mathcal{P}2$ is *NP-hard.*

Proof: We first need to introduce the uncapacitated facility location (UFL) problem [32]:

$$\min \sum_{j\in\mathcal{F}}\sum_{i\in\mathcal{N}} c_{ij}x_{ij} + \sum_{i\in\mathcal{N}} f_i y_i$$
$$\text{s.t.} \sum_{i\in\mathcal{N}} x_{ij} = 1 \text{ for each } j \in \mathcal{F},$$
$$x_{ij} \le y_i,$$
$$x_{ij} \in [0,1], y_i \in \{0,1\} \text{ for each } i \in \mathcal{N}, j \in \mathcal{F}. \quad (14)$$

As we know, in a typical uncapacitated facility-location problem, we are given a set of facilities $i \in \mathcal{N}$ with facility-opening costs f_i and a set of clients with demands $j \in \mathcal{F}$, and we want to open facilities (e.g., the green energy scheduling decisions $E_j^{green}(t)$ in our problem) and assign clients to open facilities (e.g., the task offloading decisions $x_{ij}(t)$ in our problem) so as to minimize the sum of facility-opening costs (e.g., the green energy used costs $\sum_{f\in\mathcal{F}} E_f^{green}(t)w_f(t)$) and client-assignment costs (e.g., the task offloading costs $\sum_{i\in\mathcal{N}}\sum_{f\in\mathcal{F}} x_{if}(t)w_{if}(t)$ in our problem).

The UFL problem is known to be *NP-complete* (see [32]). To prove the $NP-hardness$ of our problem $\mathcal{P}2$, we first prove that the UFL problem can be reduced to the next problem $\tilde{\mathcal{P}}$:

$$\tilde{\mathcal{P}}: \min_{x(t),E(t)} \sum_{i\in\mathcal{N}}\sum_{f\in\mathcal{F}} x_{if}(t)w_{if}(t) + \sum_{f\in\mathcal{F}} E_f^{green}(t)w_f(t)$$
$$s.t. \quad \text{constraints} \quad (1)-(9), \quad (15)$$

where $w_{if}(t) = V\lambda_i^t T_{if}^{Front}(t)$ and $w_f(t) = V\lambda_i^e p_f^{Front}(t)[\alpha-\beta] - Q_f(t)$.

Therefore, the problem $\tilde{\mathcal{P}}$ is $NP-hard$. We can see when the set of F, B, and cloud degrades to F, some constraints are relaxed to unlimited in problem $\tilde{\mathcal{P}}$. So it is obviously to see that the problem $\tilde{\mathcal{P}}$ is a special case of our problem $\mathcal{P}2$. Thus our problem $\mathcal{P}2$ is much harder than the traditional UFL problem because of the constraints (2), (3) and (9), clearly our problem $\mathcal{P}2$ is $NP-hard$.

Therefore we prove the $NP-hardness$ of problem $\mathcal{P}2$. ■

Considering the *NP-hardness* of $\mathcal{P}2$, we can easily know that it is computationally infeasible to find the optimal solution when the system scale is getting large. Fortunately, for a MILP problem, there is plenty of heuristic algorithms that can be used to get sub-optimal solutions. And the core ideas of these solutions are first to get the optimal fractional solutions by relaxing the integral constraint of the MILP problem, then propose heuristic algorithms to round the fractional solutions into integral ones based on the special structure of the problems. Unfortunately, most of these algorithms ([32], [33] and etc) can only get parameterized or constant rounding competitive ratios which also are hard to generalize to common problems.

In [34], the pipage rounding technique is proposed to round the optimal policies, which incurs a constant competitive ratio and an additional constant rounding gap with the optimal policies. And inspired by this, we propose an extended efficient dependent rounding algorithm tailored to our problem by taking into account the edge resource constraints. Solving the integral variables in a MILP problem makes the problem much harder, and in reality, once the integral decisions are made, the remaining task offloading and green energy used problem may be optimally solved by classical linear programming methods. Therefore, getting the integral variables matters a lot.

We first relax the integral constraint (1), obtaining the fractional optimization problem $\mathcal{P}3$ as follows:

$$\mathcal{P}3:$$

$$\min_{x(t),E(t)} \sum_{i=1}^{N}\sum_{f=1}^{F} x_{if}(t)w_{if}(t) + \sum_{i=1}^{N}\sum_{b=F+1}^{F+B} x_{ib}(t)w_{ib}(t)$$

$$+ \sum_{i=1,c=F+B+1}^{N} x_{ic}(t)w_{ic}(t) + \sum_{f=1}^{F}\sum_{i=1}^{N} x_{if}(t)w_{if'}(t)$$

$$+ \sum_{f=1}^{F} E_f^{green}(t)w_f(t)$$

$$s.t. \quad \text{constraints} \quad (2)-(9),$$

$$x_{ij}(t) \in [0,1], \forall i \in \mathcal{N}, \forall j \in \{1,2,\ldots,F+B+1\}. \quad (16)$$

Since the above problem $\mathcal{P}3$ is a standard linear programming problem with linear constraints, it can be optimally solved in polynomial time, by taking existing linear programming optimization technique such as the simplex method [30]. Therefore, we invoke the simplex method to solve the linear programming $\mathcal{P}3$ (relaxed-$\mathcal{P}2$) and obtain the optimal fractional policies $(\dot{\boldsymbol{x}}(t), \dot{\boldsymbol{E}}^{\boldsymbol{green}}(\boldsymbol{t}))$.

Solving the relaxation problem $\mathcal{P}3$ for each time slot t, we can obtain the optimal fractional solution which we denote to be $\dot{\boldsymbol{x}}$. Considering the integrality constraint (9i) of the original problem $\mathcal{P}2$, it is of great importance to choose a properly rounding scheme to round the fractional solutions $\dot{\boldsymbol{x}}$ to get the binary solution $\tilde{\boldsymbol{x}}$. For this purpose, the independent randomized rounding method is a straightforward solution, the core idea of which is to round each fractional $\dot{x}_{ij}(t)$ to binary $\tilde{x}_{ij}(t)$ by treating $x_{ij}(t)$ as the probability of rounding $\dot{x}_{ij}(t)$ up, i.e., $\Pr[\tilde{x}_{ij}(t)=1] = \dot{x}_{ij}(t)$.

Adopting the above independent rounding solution may always generate an infeasible policy. That is, with a certain probability, all variables are rounded up which may violate the capacity constraint of FESs-EH in FET and BESs in BET, or all variables are rounded down and all the tasks needed to be sent to the cloud to execute which would incur high cloud rental fees. To address the above deficiency, we resort to exploiting the inherent dependence of variables $\dot{x}_{ij}(t)$ to improve the optimality and further to propose a resource-constrained randomized dependent rounding (RCRDR) solution. Obviously, the term "dependent" underscores the fact various random decisions we make are highly dependent. Our key idea is that a rounded-down variable will be compensated by another rounded-up variable, ensuring that the physical resource constraint in both FET and BET can be satisfied even after the rounding.

We give a simple expression of RCRDR scheme for $\mathcal{P}2$. For each task i, we introduce two sets: the *rounded* set $\tilde{S}_i^+(t) = \{k|\dot{x}_{ik}(t) \in \{0,1\}\}$ and the *floating* set $\tilde{S}_i^-(t) = \{k|\dot{x}_{ik}(t) \in [0,1]\}$. Intuitively, $\tilde{S}_i^+(t)$ denotes the set of tasks with binary $\tilde{x}_{ij}(t)$ while $\tilde{S}_i^-(t)$ denotes the set of tasks with fractional $\dot{x}_{ij}(t)$ and thus should be rounded. For each fractional $\dot{x}_{ik}(t)$, we introduce a probability coefficient p_{ik} and a weight coefficient w_{ik} associated with it. We initialize $p_{ik} = \dot{x}_{ij}(t)$, and $w_{ik} = C_i(t)$.

Our proposed RCRDR scheme runs a series of rounding iterations by rounding the elements in $\tilde{S}_i^-(t)$ with fractional $x_{ij}(t)$. Specially, at each iteration, we randomly select two elements k_1 and k_2 from the *floating* $\tilde{S}_i^-(t)$, and let the probability of one of these two elements round to 0 or 1, then used the parameter γ_1 and γ_2 to denote, which can also adjust the value of p_{ik_1} and p_{ik_2}. Note that after adjustment, at least one of p_{ik_1} and $p_{ik_2} \in \{0,1\}$, and then set $\tilde{x}_{ik_1}(t) = p_{ik_1}$ or $\tilde{x}_{ik_2}(t) = p_{ik_2}$. By doing so, at each iteration, the number of elements in $\tilde{S}_i^-(t)$ would decrease at least by 1. Finally, when $\tilde{S}_i^-(t)$ has only one element in the last iteration, we set the last element $p_{ik} = 1$ if the total physical resource constraint in FET $\sum_{k\in\mathcal{F}} \tilde{x}_{ik}(t)C_i(t)$ is smaller than $P_k^{Front}(t)$ or if the total physical resource constraint in BET $\sum_{k\in\mathcal{B}} \tilde{x}_{ik}(t)C_i(t)$ is smaller than $P_k^{Back}(t)$ when $k \in \{F+1, F+2, \ldots, F+B\}$, otherwise set $p_{ik} = 0$. In Algorithm 2 we summarize the above detailed RCRDR algorithm.

We should note that our proposed RCRDR algorithm is cost-efficient and computation-efficient, in terms of that it would not offload all tasks to the cloud to increase the cloud rental fees, and it only has the complexity of $O(|N||H|)$ to make the rounding process because there are at most $|N||H|$ fractional variables, where $H = F + B + 1$. Therefore, the complexity of Algorithm 1 is $O(|N||H||T|)$. We will illustrate three important properties of the RCRDR algorithm and their proofs in the following.

- Continuous reduction property. After each iteration, there will be at least one of the two selected variables $\dot{x}_{ik_1}(t)$ and $\dot{x}_{ik_2}(t)$ rounded to be an integer.
- Weighted conservation property. That is, after the main loop of each iteration, the sum of the products of the selected two elements' probabilities and their weights remains unchanged. Moreover, the RCRDR algorithm can

Algorithm 2: The Proposed RCRDR for Solving $\mathcal{P}2$.

Input: $Q_f(0)$, V, r_f^{max}, $\pi_i(t)$, $P_f^{Front}(t)$, $P_b^{Back}(t)$;
Output: the rounded policies $\dot{x}(t)$ and green energy scheduling decisions $\dot{E}^{green}(t)$;

1: Invoke the simplex method to solve the LP problem $\mathcal{P}3$ and obtain the optimal fractional policies $(\dot{\boldsymbol{x}}(t), \dot{\boldsymbol{E}}^{\boldsymbol{green}}(\boldsymbol{t}))$;
2: **for** each task $i \in \mathcal{N}$ **do**
3: let $\tilde{S}_i^+(t) = \{k|\dot{x}_{ik}(t) \in \{0,1\}\}$, $\tilde{S}_i^-(t) = \{k|\dot{x}_{ik}(t) \in [0,1]\}$;
4: **for** each offloadee $k \in \tilde{S}_i^+(t)$ **do**
5: set $\tilde{x}_{ik}(t)$=$\dot{x}_{ik}(t)$;
6: **end for**
7: **for** each offloadee $k \in \tilde{S}_i^-(t)$ **do**
8: set $p_{ik} = \dot{x}_{ik}(t)$ and $w_{ik} = C_i(t)$;
9: **end for**
10: **while** $\tilde{S}_i^-(t) > 1$ **do**
11: Randomly select two elements k_1, k_2 from $\tilde{S}_i^-(t)$;
12: Define $\gamma_1 = \min\{1 - p_{ik_1}, \frac{w_{ik_2}}{w_{ik_1}} p_{ik_2}\}$, $\gamma_2 = \min\{p_{ik_1}, \frac{w_{ik_2}}{w_{ik_1}}(1 - p_{ik_2})\}$;
13: With the probability $\frac{\gamma_2}{\gamma_1 + \gamma_2}$ set, $p_{ik_1} = p_{ik_1} + \gamma_1$, $p_{ik_2} = p_{ik_2} - \frac{w_{ik_1}}{w_{ik_2}}\gamma_1$;
14: With the probability $\frac{\gamma_1}{\gamma_1 + \gamma_2}$ set, $p_{ik_1} = p_{ik_1} - \gamma_2$, $p_{ik_2} = p_{ik_2} + \frac{w_{ik_1}}{w_{ik_2}}\gamma_2$;
15: If $p_{ik_1} \in \{0,1\}$, then set $\tilde{x}_{ik_1}(t) = p_{ik_1}$;
16: $\tilde{S}_i^+(t) = \tilde{S}_i^+(t) \cup k_1$, $\tilde{S}_i^-(t) = \tilde{S}_i^-(t) \setminus k_1$;
17: If $p_{ik_2} \in \{0,1\}$, then set $\tilde{x}_{ik_2}(t) = p_{ik_2}$;
18: $\tilde{S}_i^+(t) = \tilde{S}_i^+(t) \cup k_2$, $\tilde{S}_i^-(t) = \tilde{S}_i^-(t) \setminus k_2$;
19: **end while**
20: **if** $|\tilde{S}_i^-(t)|$=1 **then**
21: Set $\tilde{x}_{ik_1}(t) = \lceil \dot{x}_{ik_1}(t) \rceil$ the only one element k in $\tilde{S}_i^-(t)$;
22: $\tilde{S}_i^+(t) = \tilde{S}_i^+(t) \cup k$, $\tilde{S}_i^-(t) = \tilde{S}_i^-(t) \setminus k$;
23: **end if**
24: **end for**
25: Apply again the simplex method to problem $\mathcal{P}2$ when knowing $\tilde{x}(t)$ and obtain $\tilde{E}^{green}(t)$.

finally achieve that:

$$\sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{F}+\mathcal{B}} \tilde{x}_{ik}(t) C_i(t) \leq \sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{F}+\mathcal{B}} \dot{x}_{ik}(t) C_i(t) + |\mathcal{N}| \max_{i\in\mathcal{N}} C_i(t), \tag{17}$$

which means that the total physical resources constraints in both FET and BET after running RCRDR are bounded by the total physical resources of fractional policies plus $|\mathcal{N}| \max_{i\in\mathcal{N}} C_i(t)$, and $\max_{i\in\mathcal{N}} C_i(t)$ is the maximum computing resources required for the task $i \in \mathcal{N}$. This property plays the key role in ensuring the capacity constraint Eq. (2) and Eq. (3) would not be violated when running RCRDR to round the fractional solution. Also, we can see that our resource-constrained randomized dependent rounding algorithm fully considers resource constraints in edge servers.

- Marginal distribution property. In the main while loop, the probability of each element $\dot{x}_{ik}(t) \in \tilde{S}_i^-(t)$ satisfies $\Pr(\tilde{x}_{ik}(t)=1)=\dot{x}_{ik}(t)$, $\Pr(\tilde{x}_{ik}(t)=0)=1\text{-}\dot{x}_{ik}(t)$. This property indicates that the task would not be offloaded to several servers and will be assigned only once when rounding the fractional policies.

The detailed proof is given in our online technical report [31]. We should note that the inequality (17) values a lot since it establishes the connection between rounding computing resources sizes and optimal fractional computing resources sizes, which also will be used as a bridge to analyze the gap of our RCRDR algorithm in the next section.

V. Performance Analysis

In this section, we will illustrate the performance of our proposed algorithms. We first give the rounding gap of $\mathcal{P}2$ and then presents the performance analysis of $\mathcal{P}1$ using the Lyapunov optimization technique [19].

A. Rounding Gap of RCRDR

We first give our conclusion of the rounding gap of RCRDR in the following theorem:

Theorem 2: After applying the randomized dependent rounding RCRDR algorithm to get the final integral solutions $\tilde{x}(t)$ and to further get the $E^{\tilde{green}}(t)$, the maximum cost of $\mathcal{P}2$ is proportional to the optimal value and has an additional constant deviation, i.e.,

$$\tilde{C}(x_{ij}(t)) + \tilde{C}(x_{if}(t)) + \tilde{C}(E^{green}(t)) \leq \phi C_t^{opt} + \Pi. \tag{18}$$

Here $\tilde{C}(x_{ij}(t)) + \tilde{C}(x_{if}(t)) + \tilde{C}(E^{green}(t))$ is the cost of problem $\mathcal{P}2$ after applying the rounding decisions $\tilde{x}(t)$ and $\tilde{E}^{green}(t)$, and C_t^{opt} is the optimal cost of original problem $\mathcal{P}2$ in time slot t. ϕ and Π are constants which will be specified in our online technical report [31] in which the detailed proof of Theorem 2 will be given.

The basic idea is to leverage the relationship between the optimal fractional solution $\tilde{x}_{ik}(t)$ and the rounded solution $\tilde{E}_j^{green}(t)$, which has been characterized by (17). Then, we further take this connection as a bridge to bound each cost terms of problem $\mathcal{P}2$.

B. Performance Analysis of JTOES

This subsection presents the performance analysis of JTOES using the Lyapunov optimization technique [19] while using the RCRDR to solve the subproblem $\mathcal{P}2$. The following theorem gives our conclusions.

Theorem 3: By applying JTOES to $\mathcal{P}1$ and RCRDR to the subproblem $\mathcal{P}2$, the time-average system cost satisfies:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{C(\boldsymbol{x}(t),\boldsymbol{E^{green}}(t))\}\leq\frac{B}{V}+\phi C_t^{opt}+\Pi, \tag{19}$$

and the time-average harvested green energy satisfies:

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{f\in\mathcal{F}}\mathbb{E}\{Q_f(t)\}$$
$$\leq\frac{B}{\eta}$$
$$+V\frac{\phi[C^{max}(\boldsymbol{x}(t),\boldsymbol{E}^{green}(t))-C^{opt}(\boldsymbol{x}(t),\boldsymbol{E}^{green}(t))]}{\eta}. \tag{20}$$

Here $C^{opt}(\boldsymbol{x}(t),\boldsymbol{E}^{green}(t))=\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{C(\boldsymbol{x}^{opt}(t),(\boldsymbol{E}^{green})^{opt}(t))\}$ is the optimal time-average cost, $C^{max}(\boldsymbol{x}(t),\boldsymbol{E}^{green}(t))$ is the largest cost, and $\eta\geq 0$ is a constant which represents the long-term queue length surplus achieved by some stationary control policies. And as we have specified above that $B=1/2\sum_{f=1}^{F}((E_f^{max})^2+(r_f^{max})^2)$ and $E_f^{max}=\max_t E_f^{green}(t)$. Due to the limitation of pages, the constants ϕ, Π and η will be specified in our online technical report [31].

Besides, the proof of Theorem 3 will be given in our online technical report [31]. By adjusting the parameter V, we can balance between the performance gap from the offline optimum and the expected energy queue length. Performance evaluation in Section VI corroborates that our proposed approximated algorithm is highly efficient, with at most 4% higher system cost than the optimal solution.

VI. Performance Evaluation

In this section, we evaluate the performance of our proposed JTOES algorithm and RCRDR algorithm via extensive trace-driven simulations. The parameters about tasks' and wireless connections set in our simulation are based on the commonly adopted wireless environment settings that have been used in [35], [36]. The harvested green energy amount is assumed to follow a uniformly distribution, and the detailed value range is similar to the setting in [37].

We simulate a hierarchal edge computing system with 5 FESs-EH and 2 BESs. For each FES-EH f, its accessible BES set $\mathcal{B}_f$ is chosen uniformly randomly from the power set of the BESs set with size $|\mathcal{B}_f|=3$. In this paper, similar to many existing studies such as [38] [39], we assume that the backhaul capacities between the front-edge tier and the backend edge-tier as well as the capacities between the backend edge tier and the cloud tier are fixed parameters. We set the time slot length $\tau_0=1$ second. During each time slot, the tasks arrive at the system in the unit of packets, each with a size of [500,2000] KB. All results are averaged over 50000 time slots. The values of the other parameters are listed in Table II .

For performance comparison, we consider three benchmark policies, namely, *Cloud server execution* (Cloud execution), *Dynamic execution with greedy energy scheduling* (Dynamic execution (GD)), Dynamic execution under no energy harvesting (Dynamic execution (NoEH)), which minimize the execution cost at the current time slot. They work as follows:

1) *Cloud execution:* The tasks are offloaded to cloud server to execute.
2) *Dynamic execution (GD):* In this situation, the green energy is scheduled greedily, which means that the $E_f^{green}(t)$ is not a decision variable, the tasks are executed by offloading to the execution modes with the minimum costs. While in this situation, the FES-EH' costs can be denoted in two circumstances, case 1: the green energy in FES-EH f is capable of providing the energy to execute task i, then $\tilde{C}_i^{Front}(t)=\lambda_i^t T_{if}^{Front}(t)+\alpha*\lambda_i^e p_f^{Front}(t)E_{if}^{Front}(t)$, case 2: otherwise we have: $\tilde{C}_i^{Front}(t)=\lambda_i^t T_{if}^{Front}(t)+\beta*\lambda_i^e p_f^{Front}(t)E_{if}^{Front}(t)$.
3) *Dynamic execution (NoEH):* The FESs-EH are not equipped with the energy harvesting ability and the operators select the minimum costs of offloading mode for each task. While in this situation, the FESs' costs, the BESs' costs, and the cloud rental fees caused by executed IoT device i' tasks at time slot t can be denoted as $\tilde{C}_i^{Front}(t)=\lambda_i^t T_{if}^{Front}(t)+\lambda_i^e p_f^{Front}(t)E_{if}^{Front}(t)$,

TABLE II
Simulation Setup and System Parameters

Parameters	Value Range
simulation timespan T	5000
number of FESs-EH in FET F	5
number of BESs in BET B	2
number of tasks N	10
task input size	[5000,10000]KB
task output size ratio	[0,30%]
the amount of computing resources	[0.15,0.20]G cycles
data rates from devices to FESs-EH	[250,300]Mbps
data rates from devices to BESs	[200,250]Mbps
data rates from device to cloud server	[90,110]Mbps
energy consumption in FESs-EH	0.005J/KB
energy consumption in BESs	0.015J/KB
resource rental fee in cloud server	0.5J/KB
computation capacity of FESs-EH	0.55G CPU
computation capacity of BESs	0.95G CPU
energy weighting parameters	[0,1]
time weighting parameters	[0,1]
electricity price of FESs-EH	[1,2]$
electricity price of in BESs	[2,3]$
harvested green energy	[0,80]J
Lyapunov control parameter	[0.1,10]

TABLE III
Average Running Time Among Different Rounding Algorithms

	Opt	RCRDR	IRR	Top-i
$N=25,\ F=4$	0.3008s	0.0190s	0.0201s	0.0215s
$N=50,\ F=5$	3.4326s	0.2081s	0.2158s	0.9219s
$N=75,\ F=6$	13.5246s	0.3536s	0.3716s	3.7492s

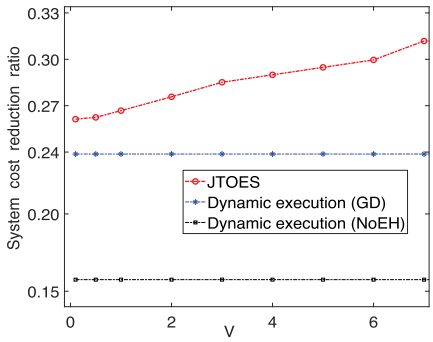


Fig. 4. Reduction ratio of costs vs control parameter V under different methods.

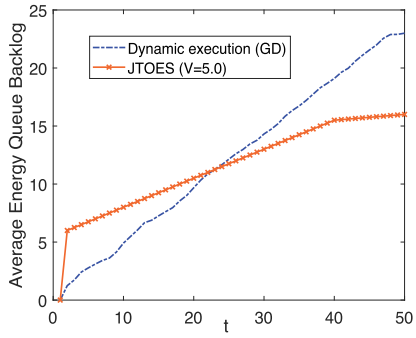


Fig. 5. The dynamic of average energy queue vs time slot t under different methods.

$C_i^{Back}(t) = \lambda_i^t T_{ib}^{Back}(t) + \lambda_i^e p_b^{Back}(t) E_{ib}^{Back}(t)$, $C_i^{Cloud}(t) = \lambda_i^t T_i^{Cloud}(t) + \lambda_i^e C_i^{Cloud}(t)$. Operators select the minimum value of $\tilde{C}_i^{Front}(t)$, $C_i^{Back}(t)$, $C_i^{Cloud}(t)$ and offload the tasks to the appropriate mode to execute.

In our experiment, **Cloud execution** is used as the benchmark of system cost saving rate.

A. Cost Trade-Off

We conduct the simulations under different values of V to evaluate the cost-queue trade-off performance of JTOES.

As shown in Fig. 4, when V increases from 0.1 to 10, the time-average system cost reduction ratio increases until it gets to a minimum which means that the system cost is getting smaller and finally approaches to a minimum under this system circumstances. Specifically, when V changes from 2 to 7, the system cost decreased dramatically. This phenomenon verifies the (19) in Theorem 3, Section V, where the time-average cost is proportional to the values of $1/V$. Therefore, the larger V we take, the smaller system costs we get. Fig. 4 also shows that our JTOES achieves excellent performance in system cost comparing to the dynamic execution (GD) algorithm and dynamic execution (NoEH) algorithm. We can see that the dynamic execution (NoEH) algorithm achieves the smallest cost reduction ratio when comparing to JTOES and dynamic execution (GD), this proves that our scheduling green energy method can effectively reduce the system costs.

To further prove the effective performance in energy queue scheduling of our JTOES, we depict the instant and time-average energy queue backlog in Fig. 5 in which that our JTOES outperforms the dynamic execution (GD) algorithm in the green energy $E_f^{green}(t)$ scheduling.

In Fig. 6, we can see that when V increases, the time-average energy queue backlog grows and almost proportionally to the values of V. This phenomenon verifies the (20) in Theorem 3, Section V, where the queue length is almost proportional to the values of V. And we can see as the value of V increases, the energy queue backlog gradually changes little, this is because in our simulation the tasks arriving ration is not a variable value thus a bigger value of V may not influence too much on the changes of our system costs. To balance the trade-off between the task execution cost and the queue backlog, we choose $V = 7$ for the

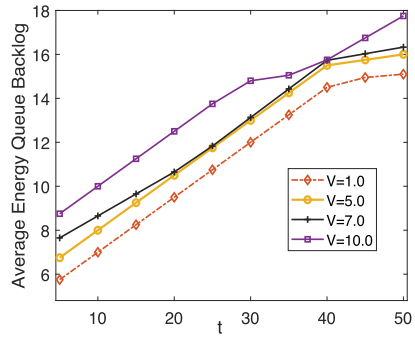


Fig. 6. The dynamic of energy queue vs control parameter V under JTOES.

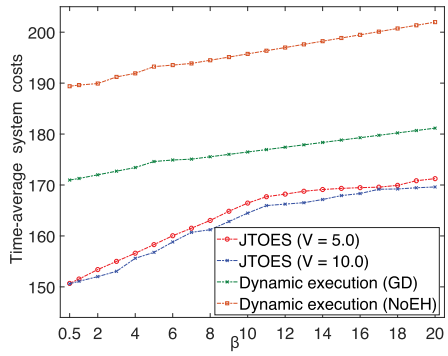


Fig. 7. Average system costs in JTOES vs different β under different methods.

rest simulations in which the value of V is not the variable. From Fig. 5 and Fig. 6 we can see no matter what V is, the change curve of the energy queue will gradually become stable over time, which shows that our JTOES can provide sustainable energy for system task execution when the maximum battery capacity is fixed, for which the dynamic execution (GD) algorithm cannot fulfill.

B. Impact of β, Tasks Number N, Edge Servers Number F on System Performance

Fig. 7 shows that the influence of the cost-efficient of using grid β in FET on system performance. In our simulation, the cost-efficient of using green energy α is set to be 0.5, and we set the value of β from 0.5 to 20. A larger β means a higher price to buy the grid power in FET and it will influence the system costs. Specifically, setting $\beta = \alpha = 0.5$ achieves

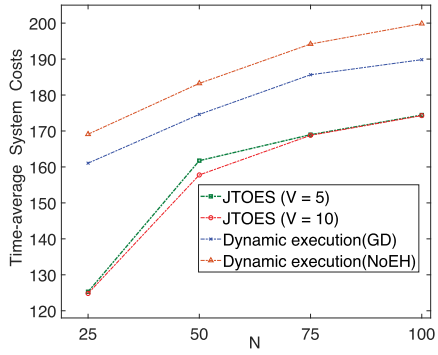


Fig. 8. Average System costs vs different tasks number N under different methods.

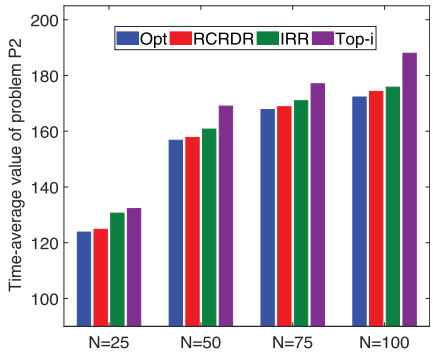


Fig. 10. Time-average value of problem $\mathcal{P}2$ under different tasks number N.

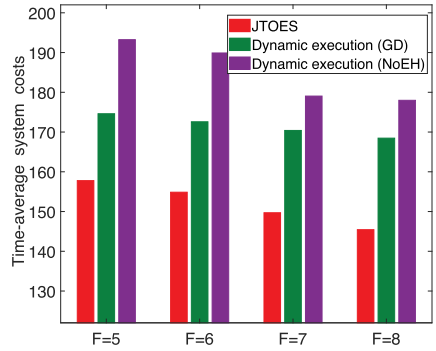


Fig. 9. Time-average system costs under different values of F.

the smallest system cost, and the time-average cost in four methods all increasing as the value of β increases. The JTOES algorithm we proposed always achieves better performance than the dynamic execution (GD) algorithm and dynamic execution (NoEH) algorithm. As we have verified aforementioned, the time-average cost is proportional to the values of $1/V$, in Fig. 7, $JTOES(V = 10.0)$ achieves the bigger time-average system cost than $JTOES(V = 5.0)$. And we choose $\beta = 5$ for the rest simulations.

The effect of the number of devices N on the system performance is shown in Fig. 8. It can be seen that with the increase of N, the performance of our proposed algorithm JTOES is better than the other two algorithms (compared with dynamic execution (GD) algorithm and dynamic execution (NoEH) algorithm, the system task execution cost is decreased by more than 6% and 15% respectively). All these results show that our proposed algorithm JTOES in this paper has a good scalability.

Fig. 9 shows that the time-average system cost under different number of FESs-EH in FET (the different computing capacity in FET). When the number of FESs-EH in FET continues to increase, our JTOES and the benchmarks all have decreasing system costs. This is because the more computing resources the FET has, they can serve more tasks itself and there is no need to offload the tasks to the BET or the public cloud, avoiding a large offloading latency. Notice that our JTOES always achieves the smallest system cost among the different computing capacity in FESs-EH (at least 10% lower than dynamic execution (GD)). Specifically, when the value of F increases from 5 to 8, the system cost of JTOES decreases by 11.5% dramatically, which shows the fluctuation of the computing capacity in FET influence the system cost significantly.

C. Approximation Performance of RCRDR in JTOES

In this subsection, we conduct substantial simulations to evaluate the approximation performance of RCRDR for JTOES. We compare RCRDR with the following benchmarks:

- *Optimal Solution (Opt):* Directly solves $\mathcal{P}2$ by using a commercial-grade MILP solver (dual-simplex solver) to compute the optimal solutions;
- *Independent Randomized Rounding (IRR):* Ignoring the computing capacity in FET and BET, we rounded the fractional optimal solutions with possibility $\Pr[\tilde{x}_{ij}(t) = 1] = \dot{x}_{ij}(t)$;
- *Top-i:* Places the Top-i tasks according to the fractional optimal solutions (sort the $\dot{x}_{ij}(t)$ in a descending order, and then round the biggest fractional element to 1 one by one) while satisfying the computing capacity of the FESs-EH in FET and BESs in BET.

As shown in Fig. 10 , we present the time-average objective value of the problem $\mathcal{P}2$ while using different rounding policies. We can see that the RCRDR algorithm always achieves the best approximation performance to the optimal solutions Opt under different values of N. The rounding policy IRR always achieves a good performance in minimizing the problem $\mathcal{P}2$, only slightly larger than our rounding policy RCRDR, however, it cannot guarantee the computing capacity constraint due to its probabilistic nature (when all the $x_{ij}(t) = 1, j \in \{1, 2, \ldots, F, F+1, \ldots, F+B\}$, the FESs-EH in FET or BESs in BET can not have enough computing resources to compute all the tasks). In contract, our rounding policy RCRDR can finely inherit the probabilistic of IRR and use the dependence of x_{ij} to satisfy the computing constraint in FET and BET.

Besides achieving the near-optimal performance, the RCRDR algorithm also does well in the average running time. In Fig. 11, we show the average running time of the benchmarks. We can see that the average running time of the optimal solver Opt increases dramatically when the tasks number grows larger, that is because of the NP-hardness of $\mathcal{P}2$. However, the running time of our RCRDR only increases proportionally with the tasks number. Besides, our RCRDR always achieves the shortest running time when comparing with IRR algorithm and Top-i algorithm.This

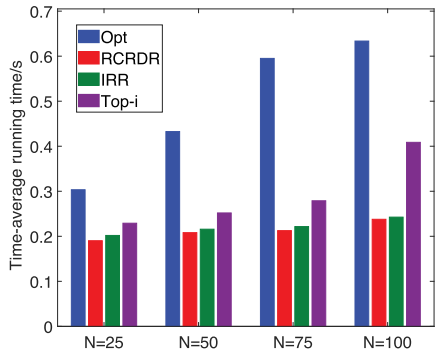


Fig. 11. Average running time vs different tasks number N under different methods.

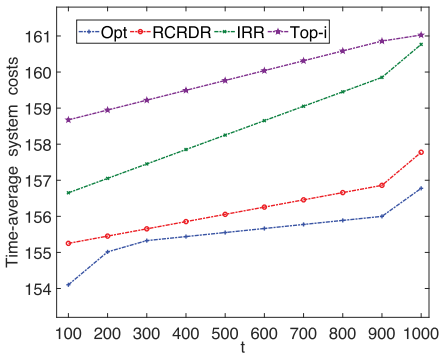


Fig. 12. The dynamic of time-average system costs vs time slot t under different rounding methods.

shows that our RCRDR has good scalability and can be used to handle a large system where contains a big number of tasks.

In Fig. 12, we show the detailed time-average system costs among different approximation policies vs time slot t. We can see as time slot t increases, the time-average system cost achieved by RCRDR and Opt is very close, and the approximate ratios between RCRDR and Opt to maintain in a very small value (only 4% larger than the optimal solutions). This shows that the RCRDR can always achieve a good performance in system cost minimization.

Moreover, we further extend the number of F to evaluate the performance of RCRDR. From TABLE III we can see when the number of tasks and FESs-EH in FET continues to increase, the running time of Opt increases dramatically, especially when the system size is larger than $N = 75, F = 6$, Opt takes nearly 38 times longer than RCRDR. Therefore, the RCRDR can provide a good trade-off between performance and time complexity, especially when the system size is large. Though the IRR achieves a small running time, it violates the computing resource capacity in both FET and BET.

VII. Conclusion

In this paper, we study the joint task offloading and energy scheduling problem in an edge computing system with multiple tiers in which the edge servers in FET can harvest the green energy. We introduce harvested energy (green energy) into the edge servers when managing a multi-tier edge computing system and consider minimizing the system cost that covers latency, energy consumption, and cloud rental fees. We further develop a novel approximated dynamic optimization framework to overcome the uncertainty of system information, only using the current system information to minimize the system costs meanwhile ensuring the stability of the harvested energy buffer queues. The rigorous performance analysis reveals the rounding gap and the extensive performance evaluation corroborate the excellent performance on our framework. For the future work, we are going to explore the possibility of integrating the green energy prediction techniques into the holistic algorithm design.

References

[1] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L.-C. Wang, "Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 44–52, Jun. 2019.

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Comput. Graph. Appl.*, vol. 21, no. 6, pp. 34–47, Nov./Dec. 2001.

[3] Z. Xiong *et al.*, "UAV-assisted wireless energy and data transfer with deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 85–99, Mar. 2021.

[4] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6G networks," *IEEE Netw.*, vol. 34, no. 6, pp. 272–280, Nov./Dec. 2020.

[5] W. Hu *et al.*, "Quantifying the impact of edge computing on mobile applications," in *Proc. 7th ACM SIGOPS Asia-Pacific Workshop Syst.*, New York, NY, USA: Association for Computing Machinery, 2016, pp. 1–8.

[6] X. Chen, Q. Shi, L. Yang, and J. Xu, "ThriftyEdge: Resource-efficient edge computing for intelligent IoT applications," *IEEE Netw.*, vol. 32, no. 1, pp. 61–65, Jan./Feb. 2018.

[7] H. Ma, Z. Zhou, and X. Chen, "Predictive service placement in mobile edge computing," in *Proc. IEEE/CIC Int. Conf. Commun. China*, 2019, pp. 792–797.

[8] T. Koketsu Rodrigues, J. Liu, and N. Kato, "Offloading decision for mobile multi-access edge computing in a multi-tiered 6G network," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2021.3090061.

[9] T. K. Rodrigues, J. Liu, and N. Kato, "Application of cybertwin for offloading in mobile multi-access edge computing for 6G networks," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16231–16242, Nov. 2021.

[10] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. INFOCOM IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[12] N. Bhushan *et al.*, "Network densification: The dominant theme for wireless evolution into 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 82–89, Feb. 2014.

[13] D. Li, K. Ota, Y. Zhong, M. Dong, Y. Tang, and J. Qiu, "Towards high-efficient transaction commitment in a virtualized and sustainable RDBMS," *IEEE Trans. Sustain. Comput.*, vol. 6, no. 3, pp. 507–521, Jul.-Sep. 2021.

[14] S. P. Beeby, M. J. Tudor, and N. M. White, "Energy harvesting vibration sources for microsystems applications," *Meas. Sci. Technol.*, vol. 17, no. 12, pp. R175–R195, Oct. 2006.

[15] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Commun. Surv. Tut.*, vol. 17, no. 2, pp. 757–789, Apr.-Jun. 2015.

[16] Q. Li and M. Zhou, "The survey and future evolution of green computing," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun.*, 2011, pp. 230–233.

[17] M. C. V. S. Mary, G. P. Devaraj, T. A. Theepak, D. J. Pushparaj, and J. M. Esther, "Intelligent energy efficient street light controlling system based on IoT for smart city," in *Proc. Int. Conf. Smart Syst. Inventive Technol.*, 2018, pp. 551–554.

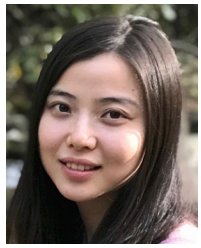
[18] Y.-C. Chang and Y.-H. Lai, "Campus edge computing network based on IoT street lighting nodes," *IEEE Syst. J.*, vol. 14, no. 1, pp. 164–171, Mar. 2020.

[19] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[20] H. Sun, F. Zhou, and R. Q. Hu, "Joint offloading and computation energy efficiency maximization in a mobile edge computing system," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 3052–3056, Mar. 2019.
[21] J. Feng, Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1320–1323, Oct. 2019.
[22] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.
[23] K. Peng, J. Nie, N. Kumar, C. Cai, and Y. Zhang, "Joint optimization of service chain caching and task offloading in mobile edge computing," *Appl. Soft Comput.*, vol. 103, no. 3, 2021, Art. no. 107142.
[24] Z. Li, M. Xu, J. Nie, J. Kang, W. Chen, and S. Xie, "Noma-enabled cooperative computation offloading for blockchain-empowered Internet of Things: A learning approach," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2364–2378, Feb. 2021.
[25] 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.
[26] 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.
[27] 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, Dec. 2016.
[28] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.
[29] L. Park, C. Lee, W. Na, S. Choi, and S. Cho, "Two-stage computation offloading scheduling algorithm for energy-harvesting mobile edge computing," *Energies*, vol. 12, no. 22, 2019, Art. no. 4367.
[30] J. A. Nelder and R. Mead, "A simplex method for function minimization comput," *Comput. J.*, no. 4, p. 4, 1965.
[31] H. Ma, P. Huang, Z. Zhou, and X. Chen, "Greenedge: Joint green energy scheduling and dynamic task offloading in multi-tiered edge computing systems," [Online]. Available:https://bit.ly/3oY2CRE.
[32] F. A. Chudak and D. B. Shmoys, "Improved approximation algorithms for the uncapacitated facility location problem," *SIAM J. Comput.*, vol. 33, no. 1, pp. 1–25, 2003.
[33] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM J. Comput.*, vol. 38, no. 4, pp. 1411–1429, 2008.
[34] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *J. Combinatorial Optim.*, vol. 8, no. 3, pp. 307–328, 2004.
[35] H. Ma, Z. Zhou, and X. Chen, "Leveraging the power of prediction: Predictive service placement for latency-sensitive mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6454–6468, Oct. 2020.
[36] X. Chen, B. Proulx, X. Gong, and J. Zhang, "Exploiting social ties for cooperative D2D communications: A mobile social networking case," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1471–1484, Oct. 2015.
[37] W. Fang, X. Zhao, Y. An, J. Li, Z. An, and Q. Liu, "Optimal scheduling for energy harvesting mobile sensing devices," *Comput. Commun.*, vol. 75, no. 1, pp. 62–70, 2016.
[38] T. Yang, N. Liu, W. Kang, and S. ShamaiShitz, "Converse results for the downlink multicell processing with finite backhaul capacity," *IEEE Trans. Inf. Theory*, vol. 65, no. 1, pp. 368–379, Jan. 2019.
[39] J. Kim, H. Lee, S.-E. Hong, and S.-H. Park, "Flexible connectivity level control via rate splitting in fog RAN downlink with finite-capacity backhaul," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2020, pp. 984–988.

Huirong Ma (Graduate Student Member, IEEE) received the B.S. degree from the School of Computer Science and Technology, Soochow University, Suzhou, China, in 2017. She is currently working toward the Ph.D. degree with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. Her research interests include edge computing, edge intelligence, and edge reliability.

Peng Huang (Graduate Student Member, IEEE) received the bachelor's degree in 2019 from Sun Yat-sen University, Guangzhou, China, where he is currently working toward the master's degree with the School of Computer Science and Engineering. His research interests include mobile edge computing and SLAM (simultaneous localization and mapping).

Zhi Zhou (Member, IEEE) received the B.S., M.E., and Ph.D. degrees from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2012, 2014, and 2017, respectively. He is currently a Research Fellow with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. Since 2016, he has been a Visiting Scholar with the University of Göttingen, Göttingen, Germany. His research interests include edge computing, cloud computing, and distributed systems. He was nominated for the 2019 CCF Outstanding Doctoral Dissertation Award, the sole recipient of 2018 ACM Wuhan & Hubei Computer Society Doctoral Dissertation Award, and the recipient of the Best Paper Award of IEEE UIC 2018.



Xiaoxi Zhang (Member, IEEE) received the B.E. degree in electronics and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2013, and the Ph.D. degree in computer science from The University of Hong Kong, Hong Kong, in 2017. She is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-sen University (SYSU), Guangzhou, China. Before joining SYSU, she was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. Her research interests include optimization and algorithm design for networked systems, including cloud and edge computing networks, NFV systems, and distributed machine learning systems.

Xu Chen (Senior Member, IEEE) received the Ph.D. degree in information engineering from the Chinese University of Hong Kong, Hong Kong, in 2012. He is currently a Full Professor with Sun Yat-sen University, Guangzhou, China, and the Vice Director with National and Local Joint Engineering Laboratory. From 2012 to 2014, he was a Postdoctoral Research Associate with Arizona State University, Tempe, AZ, USA. From 2014 to 2016, he was a Humboldt Scholar Fellow with the Institute of Computer Science, University of Goettingen, Goettingen, Germany. He was the recipient of the prestigious Humboldt Research Fellowship awarded by Alexander von Humboldt Foundation of Germany, 2014 Hong Kong Young Scientist Runner-up Award, 2016 Thousand Talents Plan Award for Young Professionals of China, 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, 2017 IEEE ComSoc Young Professional Best Paper Award, Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics (ISI), Best Paper Runner-up Award of 2014 IEEE International Conference on Computer Communications (INFOCOM), and Best Paper Award of 2017 IEEE Intranational Conference on Communications (ICC). He is currently the Area Editor of the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, an Associate Editor for the IEEE TRANSACTIONS WIRELESS COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC) series on Network Softwarization and Enablers.