# Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices

Yuyi Mao, Jun Zhang, and Khaled B. Letaief, *Fellow, IEEE*

**Abstract**

Mobile-edge computing (MEC) is an emerging paradigm to meet the ever-increasing computation demands from mobile applications. By offloading the computationally intensive workloads to the MEC server, the quality of computation experience, e.g., the execution latency, could be greatly improved. Nevertheless, as the on-device battery capacities are limited, computation would be interrupted when the battery energy runs out. To provide satisfactory computation performance as well as achieving green computing, it is of significant importance to seek renewable energy sources to power mobile devices via energy harvesting (EH) technologies. In this paper, we will investigate a green MEC system with EH devices and develop an effective computation offloading strategy. The *execution cost*, which addresses both the execution latency and task failure, is adopted as the performance metric. A low-complexity online algorithm, namely, the *Lyapunov optimization-based dynamic computation offloading* (LODCO) algorithm is proposed, which jointly decides the offloading decision, the CPU-cycle frequencies for mobile execution, and the transmit power for computation offloading. A unique advantage of this algorithm is that the decisions depend only on the instantaneous side information without requiring distribution information of the computation task request, the wireless channel, and EH processes. The implementation of the algorithm only requires to solve a deterministic problem in each time slot, for which the optimal solution can be obtained either in closed form or by bisection search. Moreover, the proposed algorithm is shown to be asymptotically optimal via rigorous analysis. Sample simulation results shall be presented to verify the theoretical analysis as well as validate the effectiveness of the proposed algorithm.

**Index Terms**

Mobile-edge computing, energy harvesting, dynamic voltage and frequency scaling, power control, QoE, Lyapunov optimization.

The authors are with the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: {ymaoac, eejzhang, eekhaled}@ust.hk). Khaled B. Letaief is also with Hamad bin Khalifa University, Doha, Qatar (e-mail: kletaief@hkbu.edu.qa).

This work is supported by the Hong Kong Research Grants Council under Grant No. 16200214.

## I. Introduction

The growing popularity of mobile devices, such as smart phones, tablet computers and wearable devices, is accelerating the advent of the Internet of Things (IoT) and triggering a revolution of mobile applications [1]. With the support of on-device cameras and embedded sensors, new applications with advanced features, e.g., navigation, face recognition and interactive online gaming, have been created. Nevertheless, the tension between resource-limited devices and computation-intensive applications becomes the bottleneck for providing satisfactory quality of experience (QoE) and hence may defer the advent of a mature mobile application market [2].

Mobile-edge computing (MEC), which provides cloud computing capabilities within the radio access networks (RAN), offers a new paradigm to liberate the mobile devices from heavy computation workloads [3]. In conventional cloud computing systems, remote public clouds, e.g., Amazon Web Services, Google Cloud Platform and Microsoft Azure, are leveraged, and thus long latency may be incurred due to data exchange in wide area networks (WANs). In contrast, MEC has the potential to significantly reduce latency, avoid congestion and prolong the battery lifetime of mobile devices by offloading the computation tasks from the mobile devices to a physically proximal MEC server [4], [5]. Thus, lots of recent efforts have been attracted from both industry [3] and academia [6].

Unfortunately, although computation offloading is effective in exploiting the powerful computation resources at cloud servers, for conventional battery-powered devices, the computation performance may be compromised due to insufficient battery energy for task offloading, i.e., mobile applications will be terminated and mobile devices will be out of service when the battery energy is exhausted. This can possibly be overcome by using larger batteries or recharging the batteries regularly. However, using larger batteries at the mobile devices implies increased hardware cost, which is not desirable. On the other hand, recharging batteries frequently is reported as the most unfavorable characteristic of mobile phones[1], and it may even be impossible in certain application scenarios, e.g., in the wireless sensor networks (WSNs) and the IoT for surveillance where the nodes are typically hard-to-reach. Meanwhile, the rapidly increasing energy consumption of the information and communication technology (ICT) sector also brings a strong need for green computing [7]. Energy harvesting (EH) is a promising technology to resolve these issues, which can capture ambient recyclable energy, including solar radiation, wind, as

[1]CNN.com, “Battery life concerns mobile users,” available on http://edition.cnn.com/2005/TECH/ptech/09/22/phone.study/.

well as human motion energy [8], and thus it facilitates self-sustainability and perpetual operation [9].

By integrating EH techniques into MEC, satisfactory and sustained computation performance can be achieved. While MEC with EH devices open new possibilities for cloud computing, it also brings new design challenges. In particular, the computation offloading strategies dedicated for MEC systems with battery-powered devices cannot take full benefits of the renewable energy sources. In this paper, we will develop new design methodologies for MEC systems with EH devices.

### *A. Related Works*

Computation offloading for mobile cloud computing systems has attracted significant attention in recent years. To increase the batteries' lifetime and improve the computation performance, various code offloading frameworks, e.g., MAUI [10] and ThinkAir [11], were proposed. However, the efficiency of computation offloading highly depends on the wireless channel condition, as the implementation of computation offloading requires data transmission. This calls for computation offloading policies that incorporate the characteristics of wireless channels [12]–[14]. In [12], a stochastic control algorithm adapted to the time-varying wireless channel was proposed, which determines the offloaded software components. For the femto-cloud computing systems, where the cloud server is formed by a set of femto access points, the transmit power, precoder and computation load distribution were jointly optimized in [13]. In addition, a game-theoretic decentralized computation offloading algorithm was proposed for multi-user mobile cloud computing systems [14]. Nevertheless, these works assume non-adjustable processing capabilities of the central processing units (CPUs) at the mobile devices, which is not energy-efficient since the CPU energy consumption increases super-linearly with the CPU-cycle frequency [15]. With dynamic voltage and frequency scaling (DVFS) techniques, the local execution energy consumption for applications with strict deadline constraints is minimized by controlling the CPU-cycle frequencies [16]. Besides, a joint allocation of communication and computation resources for multi-cell MIMO cloud computing systems was proposed in [17]. Most recently, the energy-delay tradeoff of mobile cloud systems with heterogeneous types of computation tasks were investigated by a Lyapunov optimization algorithm, which decides the offloading policy, task allocation, CPU clock speeds and selected network interfaces [18].

Energy harvesting was introduced to communication systems for its potential to realize self-sustainable and green communications [19], [20]. With non-causal side information (SI)[2], including the channel side information (CSI) and energy side information (ESI), the maximum throughput of point-to-point EH fading channels can be achieved by the directional water-filling algorithm [21]. The study was later extended to EH networks with causal SI [22]. Cellular networks with renewable energy supplies have also been widely investigated. Resources allocation policies that maximize the energy efficiency in OFDMA systems with hybrid energy supplies (HES), i.e., both grid and harvested energy are accessible to base stations, were proposed in [23]. To save the grid energy consumption, a sleep control scheme for cellular networks with HES was developed in [24], and a low-complexity online base station assignment and power control algorithm based on Lyapunov optimization was proposed in [25].

The design principles for MEC systems with EH devices are different from those for EH communication systems or MEC systems with battery-powered devices. On one hand, compared to EH communication systems, computation offloading policies require a joint design of the offloading decision, i.e., whether to offload a task, the CPU-cycle frequencies for mobile execution[3], and the transmission policy for task offloading, which makes it much more challenging. On the other hand, compared to MEC systems with battery-powered devices, the design objective is shifted from minimizing the battery energy consumption to optimizing the computation performance as the harvested energy comes for free. In addition, taking care of the ESI is a new design consideration, and the time-correlated battery energy dynamics poses another challenge.

## *B. Contributions*

In this paper, we will investigate MEC systems with EH devices and develop an effective dynamic computation offloading algorithm. Our major contributions are summarized as follows:

- We consider an EH device served by an MEC server, where the computation tasks can be executed locally at the device or be offloaded to the MEC server for cloud execution[4].

[2]'Causal SI' refers to the case that, at any time instant, only the past and current SI is known, while non-causal SI means that the future SI is also available.

[3]We use "local execution" and "mobile execution" interchangeably in this paper.

[4]It is worthwhile to point out that powering mobile devices in MEC systems with wireless energy harvesting was proposed in [26], where the harvested energy is radiated from a hybrid access point and fully controllable. This is different from the system considered in this paper where the EH process is random and uncontrollable.

An execution cost that incorporates the execution delay and task failure is adopted as the performance metric, while DVFS and power control are adopted to optimize the mobile execution process and data transmission for computation offloading, respectively.

- The *execution cost minimization* (ECM) problem, which is an intractable high-dimensional Markov decision problem, is formulated assuming causal SI, and a low-complexity online **L**yapunov **o**ptimization-based **d**ynamic **c**omputation **o**ffloading (LODCO) algorithm is proposed. In each time slot, the system operation, including the offloading decision, the CPU-cycle frequencies for mobile execution, and the transmit power for computation offloading, only depends on the optimal solution of a deterministic optimization problem, which can be obtained either in closed form or by bisection search.
- We identify a non-decreasing property of the scheduled CPU-cycle frequencies (the transmit power) with respect to the battery energy level, which shows that a larger amount of available energy leads to a shorter execution delay for mobile execution (MEC server execution). Performance analysis for the LODCO algorithm is also conducted. It is shown that the proposed algorithm can achieve asymptotically optimal performance of the ECM problem by tuning a two-tuple control parameters. Moreover, it does not require statistical information of the involved stochastic processes, including the computation task request, the wireless channel, and EH processes, which makes it applicable even in unpredictable environments.
- Simulation results are provided to verify the theoretical analysis, especially the asymptotic optimality of the LODCO algorithm. Moreover, the effectiveness of the proposed policy is demonstrated by comparisons with three benchmark polices with greedy harvested energy allocation. It is shown that the LODCO algorithm not only achieves significant performance improvement in terms of execution cost, but also effectively reduces task failure.

The organization of this paper is as follows. In Section II, we introduce the system model. The ECM problem is formulated in Section III. The LODCO algorithm for the ECM problem is proposed in Section IV and its performance analysis is conducted in Section V. We show the simulation results in Section VI and conclude this paper in Section VII.

## II. System Model

In this section, we will introduce the system model studied in this paper, i.e., a mobile-edge computing (MEC) system with an EH device. Both the computation model and energy harvesting model will be discussed.

## A. Mobile-edge Computing Systems with EH Devices

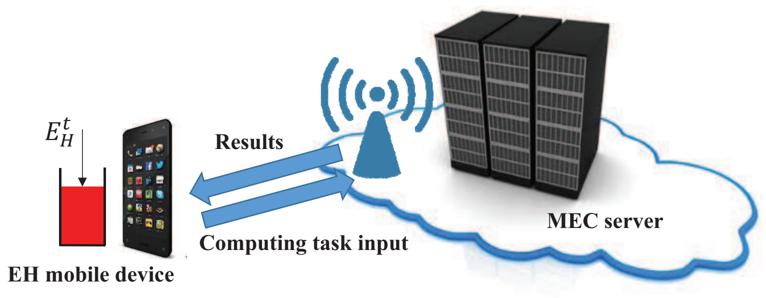


Fig. 1. A mobile-edge computing system with an EH mobile device.

We consider an MEC system consisting of a mobile device and an MEC server as shown in Fig. 1. In particular, the mobile device is equipped with an EH component and powered purely by the harvested renewable energy. The MEC server, which could be a small data center managed by the telecom operator, is located at a distance of $d$ meters away and can be accessed by the mobile device through the wireless channel. The mobile device is associated with a system-level clone at the MEC server, namely, the cloud clone, which runs a virtual machine and can execute the computation tasks on behalf of the mobile device [16]. By offloading the computation tasks for MEC, the computation experience can be improved significantly [4]–[6].

We assume that time is slotted, and denote the time slot length and the time slot index set by $\tau$ and $\mathcal{T} \triangleq \{0, 1, \cdots\}$, respectively. The wireless channel is assumed to be independent and identically distributed (i.i.d.) block fading, i.e., the channel remains static within each time slot, but varies among different time slots. Denote the channel power gain at the $t$th time slot as $h^t$, and $h^t \sim F_H(x), t \in \mathcal{T}$, where $F_H(x)$ is the cumulative distribution function (CDF) of $h^t$. For ease of reference, we list the key notations of our system model in Table I.

## B. Computation Model

We use $A(L, \tau_d)$ to represent a computation task, where $L$ (in bits) is the input size of the task, and $\tau_d$ is the execution deadline, i.e., if it is decided that task $A(L, \tau_d)$ is to be executed, it should be completed within time $\tau_d$. The computation tasks requested by the applications running at the mobile device are modeled as an i.i.d. Bernoulli process. Specifically, at the beginning of each time slot, a computation task $A(L, \tau_d)$ is requested with probability $\rho$, and with probability

TABLE I

SUMMARY OF KEY NOTATIONS

| **Notation** | **Description** |
|---|---|
| $d$ | Distance between the mobile device and the MEC server |
| $\mathcal{T}$ | Index set of the time slots |
| $h^t$ | Channel power gain from the mobile device to the MEC server in time slot $t$ |
| $A(L, \tau_d)$ | Computation task with $L$ bits input and deadline $\tau_d$ |
| $\{I_j^t\}$ | Computation mode indicators at time slot $t$ |
| $\zeta^t$ | Task arrival indicator at time slot $t$ |
| $X$ $(W)$ | Number of CPU cycles required to process one bit task input ($A(L, \tau_d)$) |
| $\{f_w^t\}$ | Scheduled CPU-cycle frequencies for local execution at time slot $t$ |
| $p^t$ | Transmit power for computation offloading at time slot $t$ |
| $f_{\mathrm{CPU}}^{\max}$ $(p_{\mathrm{tx}}^{\max})$ | Maximum allowable CPU-cycle frequency (transmit power) |
| $D_{\mathrm{mobile}}^t$ $(D_{\mathrm{server}}^t)$ | Execution delay of local execution (MEC server execution) at time slot $t$ |
| $E_{\mathrm{mobile}}^t$ $(E_{\mathrm{server}}^t)$ | Energy consumption of local execution (MEC server execution) at time slot $t$ |
| $e^t$ $(E_H^t)$ | Harvested (harvestable) energy at time slot $t$ |
| $E_H^{\max}$ | Maximum value of $E_H^t$ |
| $B^t$ | Battery energy level at the beginning of time slot $t$ |
| $\phi$ | The weight of the task dropping cost |

$1-\rho$, there is no request. Denote $\zeta^t = 1$ if a computation task is requested at the $t$th time slot and $\zeta^t = 0$ if otherwise, i.e., $\mathbb{P}(\zeta^t = 1) = 1 - \mathbb{P}(\zeta^t = 0) = \rho, t \in \mathcal{T}$. We focus on delay-sensitive applications with execution deadline less than the time slot length, i.e., $\tau_d \leq \tau$ [12]–[14], [17], [27], and assume no buffer is available for queueing the computation requests.

Each computation task can either be executed locally at the mobile device, or be offloaded to and executed by the MEC server. It may also happen that neither of these two computation modes is feasible, e.g., when energy is insufficient at the mobile device, and hence the computation task will be dropped. Denote $I_j^t \in \{0, 1\}$ with $j = \{\mathrm{m}, \mathrm{s}, \mathrm{d}\}$ as the computation mode indicators, where $I_{\mathrm{m}}^t = 1$ and $I_{\mathrm{s}}^t = 1$ indicate that the computation task requested in the $t$th time slot is executed at the mobile device and offloaded to the MEC server, respectively, while $I_{\mathrm{d}}^t = 1$ means the computation task is dropped. Thus, the computation mode indicators should satisfy the following operation constraint:

$$I_{\mathrm{m}}^t + I_{\mathrm{s}}^t + I_{\mathrm{d}}^t = 1, t \in \mathcal{T}. \tag{1}$$

**Local Executing Model:** The number of CPU cycles required to process one bit input is denoted as $X$, which varies from different applications and can be obtained through off-line measurement [28]. In other words, $W = LX$ CPU cycles are needed in order to successfully execute task $A(L, \tau_d)$. The frequencies scheduled for the $W$ CPU cycles in the $t$th time slot are denoted as $f_w^t, w = 1, \cdots, W$, which can be implemented by adjusting the chip voltage with DVFS techniques [29]. As a result, the delay for executing the computation task requested in the $t$th time slot locally at the mobile device can be expressed as

$$D_{\text{mobile}}^t = \sum_{w=1}^{W} \left(f_w^t\right)^{-1}. \tag{2}$$

Accordingly, the energy consumption for local execution by the mobile device is given by

$$E_{\text{mobile}}^t = \kappa \sum_{w=1}^{W} \left(f_w^t\right)^2, \tag{3}$$

where $\kappa$ is the effective switched capacitance that depends on the chip architecture [15]. Moreover, we assume the CPU-cycle frequencies are constrained by $f_{\text{CPU}}^{\max}$, i.e., $f_w^t \leq f_{\text{CPU}}^{\max}, \forall w$.

**Mobile-edge Executing Model:** In order to offload the computation task for MEC, the input bits of $A(L, \tau_d)$ should be transmitted to the MEC server. We assume sufficient computation resource, e.g., a high-speed multi-core CPU, is available at the MEC server, and thus ignore its execution delay [16], [18], [26]. It is further assumed that the output of the computation is of small size so the transmission delay for feedback is negligible. Denote the transmit power as $p^t$, which should be less than the maximum transmit power $p_{\text{tx}}^{\max}$. According to the Shannon-Hartley formula, the achievable rate in the $t$th time slot is given by $r\left(h^t, p^t\right) = \omega \log_2\left(1 + \frac{h^t p^t}{\sigma}\right)$, where $\omega$ is the system bandwidth and $\sigma$ is the noise power at the receiver. Consequently, if the computation task is executed by the MEC server, the execution delay equals the transmission delay for the input bits, i.e.,

$$D_{\text{server}}^t = \frac{L}{r\left(h^t, p^t\right)}, \tag{4}$$

and[5] the energy consumed by the mobile device is given by

$$E_{\text{server}}^t = p^t \cdot D_{\text{server}}^t = p^t \cdot \frac{L}{r\left(h^t, p^t\right)}. \tag{5}$$

[5]When the execution delay in the MEC server is non-negligible, the proposed algorithm can still be applied by modifying the expression of $D_{\text{server}}^t$ in (4) as $D_{\text{server}}^t = L / r\left(h^t, p^t\right) + \tau_{\text{server}}$, where $\tau_{\text{server}}$ denotes the execution delay in the MEC server.

## C. Energy Harvesting Model

The EH process is modeled as successive energy packet arrivals, i.e., $E_H^t$ units of energy arrive at the mobile device at the beginning of the $t$th time slot. We assume $E_H^t$'s are i.i.d. among different time slots with the maximum value of $E_H^{\max}$. Although the i.i.d. model is simple, it captures the stochastic and intermittent nature of the renewable energy processes [22], [25], [30]. In each time slot, part of the arrived energy, denoted as $e^t$, satisfying

$$0 \le e^t \le E_H^t, t \in \mathcal{T}, \tag{6}$$

will be harvested and stored in a battery, and it will be available for either local execution or computation offloading starting from the next time slot. We start by assuming that the battery capacity is sufficiently large. Later we will show that by picking the values of $e^t$'s, the battery energy level is deterministically upper-bounded under the proposed computation offloading policy, and thus we only need a finite-capacity battery in actual implementation. More importantly, including $e^t$'s as optimization variables facilitates the derivation and performance analysis of the proposed algorithm. Similar techniques were adopted in previous studies, such as [22], [25] and [30]. Denote the battery energy level at the beginning of time slot $t$ as $B^t$. Without loss of generality, we assume $B^0 = 0$ and $B^t < +\infty, t \in \mathcal{T}$. In this paper, energy consumed for purposes other than local computation and transmission is ignored for simplicity, while more general energy models can be handled by the proposed algorithm with minor modifications.[6] Denote the energy consumed by the mobile device in time slot $t$ as $\mathcal{E}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right)$, which depends on the selected computation mode, scheduled CPU-cycle frequencies and transmit power, and can be expressed as

$$\mathcal{E}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) = I_{\mathrm{m}}^t E_{\mathrm{mobile}}^t + I_{\mathrm{s}}^t E_{\mathrm{server}}^t, \tag{7}$$

subject to the following energy causality constraint:

$$\mathcal{E}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) \le B^t < +\infty, t \in \mathcal{T}. \tag{8}$$

Thus, the battery energy level evolves according to the following equation:

$$B^{t+1} = B^t - \mathcal{E}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) + e^t, t \in \mathcal{T}. \tag{9}$$

[6]We will demonstrate how to adapt the proposed algorithm to more general energy models of mobile devices, e.g., by taking the power consumption of screens and operating systems into account, in Section IV-A.

With EH mobile devices, the computation offloading policy design for MEC systems becomes much more complicated compared to that of conventional mobile cloud computing systems with battery-powered devices. Specifically, both the ESI and CSI need to be handled, and the temporally correlated battery energy level makes the system decision coupled in different time slots. Consequently, an optimal computation offloading strategy should strike a good balance between the computation performance of the current and future computation tasks.

## III. Problem Formulation

In this section, we will first introduce the performance metric, namely, the execution cost. The execution cost minimization (ECM) problem will then be formulated and its unique technical challenges will be identified.

### *A. Execution Cost Minimization Problem*

Execution delay is one of the key measures for users' QoE [12]–[14], [16]–[18], which will be adopted to optimize the computation offloading policy for the considered MEC system. Nevertheless, due to the intermittent and sporadic nature of the harvested energy, some of the requested computation tasks may not be able to be executed and have to be dropped, e.g., due to lacking of energy for local computation, while the wireless channel from the mobile device to the MEC server is in deep fading, i.e., the input of the tasks cannot be delivered. To take this aspect into consideration, we penalize each dropped task by a unit of cost. Thus, we define the execution cost as the weighted sum of the execution delay and the task dropping cost, which can be expressed by the following formula:

$$\mathrm{cost}^t = \mathcal{D}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) + \phi \cdot \mathbf{1}\left(\zeta^t = 1, I_{\mathrm{d}}^t = 1\right), \tag{10}$$

where $\phi$ (in second) is the weight of the task dropping cost, $\mathbf{1}\left(\cdot\right)$ is the indicator function, and $\mathcal{D}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right)$ is given by

$$\mathcal{D}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) = \mathbf{1}\left(\zeta^t = 1\right) \cdot \left(I_{\mathrm{m}}^t D_{\mathrm{mobile}}^t + I_{\mathrm{s}}^t D_{\mathrm{server}}^t\right). \tag{11}$$

Without loss of generality, we assume that executing a task successfully is preferred to dropping a task, i.e., $\tau_d \leq \phi$.

If it is decided that a task is to be executed, i.e., $I_{\mathrm{m}}^t = 1$ or $I_{\mathrm{s}}^t = 1$, it should be completed before the deadline $\tau_d$. In other words, the following deadline constraint should be met:

$$\mathcal{D}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) \leq \tau_d, t \in \mathcal{T}. \tag{12}$$

Consequently, the ECM problem can be formulated as:

$$\begin{aligned}
\mathcal{P}_1: \quad & \min_{\boldsymbol{I}^t, \boldsymbol{f}^t, p^t, e^t} \lim_{T\to\infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=0}^{T-1} \mathrm{cost}^t\right] \\
\text{s.t.} \quad & (1), (6), (8), (12) \\
& I_{\mathrm{m}}^t + I_{\mathrm{s}}^t \le \zeta^t, t \in \mathcal{T} && (13) \\
& \mathcal{E}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) \le E_{\max}, t \in \mathcal{T} && (14) \\
& 0 \le p^t \le p_{\mathrm{tx}}^{\max} \cdot \mathbf{1}\left(I_{\mathrm{s}}^t = 1\right), t \in \mathcal{T} && (15) \\
& 0 \le f_w^t \le f_{\max}^{\mathrm{CPU}} \cdot \mathbf{1}\left(I_{\mathrm{m}}^t = 1\right), w = 1, \cdots, W, t \in \mathcal{T}, && (16) \\
& I_{\mathrm{m}}^t, I_{\mathrm{s}}^t, I_{\mathrm{d}}^t \in \{0, 1\}, t \in \mathcal{T}, && (17)
\end{aligned}$$

where (13) indicates that if there is no computation task requested, neither mobile execution nor MEC server execution is feasible. (14) is the battery discharging constraint, i.e., the amount of battery output energy cannot exceed $E_{\max}$ in each time slot, which is essential for preventing the battery from over discharging [30], [31]. The maximum allowable transmit power and the maximum CPU-cycle frequency constraints are imposed by (15) and (16), respectively, while the zero-one indicator constraint for the computation mode indicators is represented by (17).

## B. *Problem Analysis*

In the considered MEC system, the system state is composed of the task request, the harvestable energy, the battery energy level, as well as the channel state, and the action is the energy harvesting and the computation offloading decision, including the scheduled CPU-cycle frequencies and the allocated transmit power. It can be checked that the allowable action set depends only on the current system state, and is irrelevant with the state and action history. Besides, the objective is the long-term average execution cost. Thus, $\mathcal{P}_1$ is a Markov decision process (MDP) problem. In principle, $\mathcal{P}_1$ can be solved optimally by standard MDP algorithms, e.g., the *relative value iteration algorithm* and the *linear programming reformulation approach* [32]. Nevertheless, for both algorithms, we need to use finite states to characterize the system, and discretize the feasible action set. For example, if we use $K = 20$ states to quantize the wireless channel, $M = 20$ states to characterize the battery energy level, $E = 5$ states to describe the harvestable energy, and admits $L = 10$ transmit power levels and $F = 10$ CPU-cycle frequencies, there are $2KME = 4000$ possible system states in total. For the relative value iteration algorithm, this will take a long time to converge as there will be as many as $L + 1 + F^W$ feasible actions in some states. For the linear programming (LP) reformulation approach, we need to solve an LP problem with $2KME \times \left(L + 1 + F^W\right)$ variables, which will be practically infeasible even for a small value of $W$, e.g., $1000$. In addition, it will be difficult to obtain solution insights with the MDP algorithms as they are based on numerical iteration. Moreover, quantizing the state and action may lead to severe performance degradation, and the memory requirement for storing the optimal policy will yet be another big challenge.

In the next section, we will propose a **L**yapunov **o**ptimization-based **d**ynamic **c**omputation **o**ffloading (LODCO) algorithm to solve $\mathcal{P}_1$, which enjoys the following favorable properties:

- There is no need to quantize the system state and feasible action set, and the decision of the LODCO algorithm within each time slot is of low complexity. In addition, there is no memory requirement for storing the optimal policy.
- The LODCO algorithm has no prior information requirement on the channel statistics, the distribution of the renewable energy process or the computation task request process.
- The performance of the LODCO algorithm is controlled by a two-tuple control parameters. Theoretically, by adjusting these parameters, the proposed algorithm can behave arbitrarily close to the optimal performance of $\mathcal{P}_1$.
- An upper bound of the required battery capacity is obtained, which shall provide guidelines for practical installation of the EH components and storage units.

## IV. Dynamic Computation Offloading: The LODCO Algorithm

In this section, we will develop the LODCO algorithm to solve $\mathcal{P}_1$. We will first show an important property of the optimal CPU-cycle frequencies, which helps to simplify $\mathcal{P}_1$. In order to take advantages of Lyapunov optimization, we will introduce a modified ECM problem to assist the algorithm design. The LODCO algorithm will be then proposed for the modified problem, which also provides a feasible solution to $\mathcal{P}_1$. In Section V, we will show that this solution is asymptotically optimal for $\mathcal{P}_1$.

### *A. The LODCO Algorithm*

We first show that the optimal CPU-cycle frequencies of the $W$ CPU cycles scheduled for a single computation task should be the same, as stated in the following lemma.

***Lemma** 1:* If a task requested at the $t$th time slot is being executed locally, the optimal frequencies of the $W$ CPU cycles should be the same, i.e., $f_w^t = f^t, w = 1, \cdots, W$.

*Proof:* The proof can be obtained by contradiction, which is omitted for brevity. ■

The property of the optimal CPU-cycle frequencies in Lemma 1 indicates that we can optimize a scalar $f^t$ instead of a $W$-dimensional vector $\boldsymbol{f}^t$ for each computation task, which helps to reduce the number of optimization variables. However, due to the energy causality constraint (8), the system's decisions are coupled among different time slots, which makes the design challenging. This is a common difficulty for the design of EH systems. We find that by introducing a non-zero lower bound, $E_{\min}$, on the battery output energy at each time slot, such coupling effect can be eliminated and the system operations can be optimized by ignoring (8) at each time slot. Thus, we first introduce a modified version of $\mathcal{P}_1$ as

$$\begin{aligned} \mathcal{P}_2 : \quad & \min_{\boldsymbol{I}^t, f^t, p^t, e^t} \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T-1} \mathrm{cost}^t\right] \\ & \text{s.t.} \quad (1), (6), (8), (12) - (17) \\ & \qquad \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right) \in \{0\} \bigcup \left[E_{\min}, E_{\max}\right], t \in \mathcal{T}, \end{aligned} \tag{18}$$

where $0 < E_{\min} \leq E_{\max}$. Compared to $\mathcal{P}_1$, only a scalar $f^t$ needs to be determined for mobile execution, which preserves optimality according to Lemma 1, and thus $D_{\mathrm{mobile}}^t = W\left(f^t\right)^{-1}$ and $E_{\mathrm{mobile}}^t = W\kappa\left(f^t\right)^2$. Besides, all constraints in $\mathcal{P}_1$ are retained in $\mathcal{P}_2$, and an additional constraint on the battery output energy is imposed by (18). Hence, $\mathcal{P}_2$ is a tightened version of $\mathcal{P}_1$. Denote the optimal values of $\mathcal{P}_1$ and $\mathcal{P}_2$ as $\mathrm{EC}_{\mathcal{P}_1}^*$ and $\mathrm{EC}_{\mathcal{P}_2}^*$, respectively. The following proposition reveals the relationship between $\mathrm{EC}_{\mathcal{P}_1}^*$ and $\mathrm{EC}_{\mathcal{P}_2}^*$, which will later help show the asymptotic optimality of the proposed algorithm.

***Proposition** 1:* The optimal value of $\mathcal{P}_2$ is greater than that of $\mathcal{P}_1$, but smaller than the optimal value of $\mathcal{P}_1$ plus a positive constant $\nu\left(E_{\min}\right)$, i.e., $\mathrm{EC}_{\mathcal{P}_1}^* \leq \mathrm{EC}_{\mathcal{P}_2}^* \leq \mathrm{EC}_{\mathcal{P}_1}^* + \nu\left(E_{\min}\right)$, where $\nu\left(E_{\min}\right) = \rho\left[\phi\left(1 - F_H(\eta)\right) + \mathbf{1}_{E_{\min} \geq E_{\min}^{\tau_d}} \cdot \left(\phi - \tau_{E_{\min}}\right)\right]$. Here, $\eta = \left(2^{\frac{L}{\tau_d \omega}} - 1\right)\sigma\tau_d E_{\min}^{-1}$, $E_{\min}^{\tau_d} = \kappa W^3 \tau_d^{-2}$ and $\tau_{E_{\min}} = \kappa^{\frac{1}{2}} W^{\frac{3}{2}} E_{\min}^{-\frac{1}{2}}$.

*Proof:* Please refer to Appendix A. ■

In general, the upper bound in Proposition 1 is not tight. However, as $E_{\min}$ goes to zero, $\nu\left(E_{\min}\right)$ diminishes as shown in the following corollary.

***Corollary** 1:* By letting $E_{\min}$ approach zero, $\mathrm{EC}_{\mathcal{P}_2}^*$ can be made arbitrarily close to $\mathrm{EC}_{\mathcal{P}_1}^*$, i.e., $\lim\limits_{E_{\min}\to 0} \nu\left(E_{\min}\right) = 0$.

*Proof:* The proof is omitted due to space limitation. ■

Proposition 1 bounds the optimal performance of $\mathcal{P}_2$ by that of $\mathcal{P}_1$, while Corollary 1 shows that the performance of both problems can be made arbitrarily close. Actually, Corollary 1 fits our intuition, since when $E_{\min} \to 0$, $\mathcal{P}_2$ reduces to $\mathcal{P}_1$. However, due to the temporally correlated battery energy levels, the system's decisions are time-dependent, and thus the vanilla version of Lyapunov optimization techniques, where the allowable action sets are i.i.d., cannot be applied directly. Fortunately, the weighted perturbation method offers an effective solution to circumvent this issue [33]. In order to present the algorithm, we first define the perturbation parameter and the virtual energy queue at the mobile device, which are two critical elements.

***Definition** 1:* The perturbation parameter $\theta$ for the EH mobile device is a bounded constant satisfying

$$\theta \geq \tilde{E}_{\max} + V\phi \cdot E_{\min}^{-1}, \tag{19}$$

where $\tilde{E}_{\max} = \min\{\max\{\kappa W \left(f_{\text{CPU}}^{\max}\right)^2, p_{\text{tx}}^{\max}\tau\}, E_{\max}\}$, and $0 < V < +\infty$ is a control parameter in the LODCO algorithm with unit as $\text{J}^2 \cdot \text{second}^{-1}$.[7]

***Definition** 2:* The virtual energy queue $\tilde{B}^t$ is defined as $\tilde{B}^t = B^t - \theta$, which is a shifted version of the actual battery energy level at the mobile device.

As will be elaborated later, the proposed algorithm minimizes the weighted sum of the net harvested energy and the execution cost in each time slot, with weights of the virtual energy queue length $\tilde{B}^t$, and the control parameter $V$, respectively, which tends to stabilize $B^t$ around $\theta$ and meanwhile minimize the execution cost. The LODCO algorithm is summarized in Algorithm 1. In each time slot, the system operation is determined by solving a deterministic per-time slot problem, which is parameterized by the current system state and with all constraints in $\mathcal{P}_2$ except the energy causality constraint (8).

***Remark** 1:* When the power consumption for maintaining the basic operations at the mobile device, denoted as $P_{\text{basic}}$, is considered, there will be four computation modes for the time slots with $\zeta^t = 1$, i.e., mobile execution ($I_{\text{m}}^t = 1$), MEC server execution ($I_{\text{s}}^t = 1$), dropping the task while maintaining the basic operations ($I_{\text{d}}^t = 1$), as well as dropping the task and disabling the basic operations ($I_{\text{f}}^t = 1$); while for the time slots with $\zeta^t = 0$, two modes exist, i.e., the basic operations are maintained ($I_{\text{d}}^t = 1$) or disabled ($I_{\text{f}}^t = 1$). As a result, the energy

[7]Since the right-hand side of (19) increases with $\phi$ ($\phi \in [\tau_d, +\infty)$), a larger value of $\phi$ will result in a large value of $\theta$, i.e., a higher perturbed energy level in the proposed algorithm.

consumed by the mobile device at the $t$th time slot can be written as $\mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right) = I_{\mathrm{m}}^t E_{\mathrm{mobile}}^t + I_{\mathrm{s}}^t E_{\mathrm{server}}^t + \left(I_{\mathrm{m}}^t + I_{\mathrm{s}}^t + I_{\mathrm{d}}^t\right) P_{\mathrm{basic}}\tau$. We introduce a unit of cost to penalize the interruption of basic operations, and thus the execution cost can be expressed as $\mathrm{cost}^t = \mathcal{D}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) + \phi \cdot \mathbf{1}\left(\zeta^t = 1, I_{\mathrm{d}}^t \text{ or } I_{\mathrm{f}}^t = 1\right) + \psi \cdot \mathbf{1}\left(I_{\mathrm{f}}^t = 1\right)$, where $\psi > 0$ is the weight of the basic operations interruption cost. It is worthwhile to note that the framework of the proposed LODCO algorithm can be modified for this case, where the major changes lie on the selection of the perturbation parameter $\theta$ and the solution for the per-time slot problem, and will not be detailed in this paper.

**Algorithm 1** The LODCO Algorithm

```
1: At the beginning of time slot $t$, obtain the task request indicator $\zeta^t$, virtual energy queue length $\tilde{B}^t$, harvestable energy $E_H^t$, and channel gain $h^t$.
2: Decide $e^t, \boldsymbol{I}^t$, $f^t$ and $p^t$ by solving the following deterministic problem:
   $\min_{\boldsymbol{I}^t, p^t, f^t, e^t} \tilde{B}^t\left[e^t - \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right)\right] + V\left[\mathcal{D}\left(\boldsymbol{I}^t, f^t, p^t\right) + \phi \cdot \mathbf{1}\left(\zeta^t = 1, I_{\mathrm{d}}^t = 1\right)\right]$
   $\text{s.t.} \quad (1), (6), (12) - (18).$
3: Update the virtual energy queue according to (9) and Definition 2.
4: Set $t = t + 1$.
```

### B. *Optimal Computation Offloading in Each Time Slot*

In this subsection, we will develop the optimal solution for the per-time slot problem, which consists of two components: the optimal energy harvesting, i.e., to determine $e^t$, as well as the optimal computation offloading decision, i.e., to determine $\boldsymbol{I}^t$, $f^t$ and $p^t$. The results obtained in this subsection are essential for feasibility verification and performance analysis of the LODCO algorithm in Section V.

**Optimal Energy Harvesting:** It is straightforward to show that the optimal amount of harvested energy $e^{t*}$ can be obtained by solving the following LP problem:

$$\min_{0 \leq e^t \leq E_H^t} \tilde{B}^t e^t, \tag{20}$$

and its optimal solution is given by

$$e^{t*} = E_H^t \cdot \mathbf{1}\{\tilde{B}^t \leq 0\}. \tag{21}$$

**Optimal Computation Offloading:** After decoupling $e^t$ from the objective function, we can then simplify the per-time slot problem into the following optimization problem $\mathcal{P}_{\text{CO}}$:

$$\begin{aligned}\mathcal{P}_{\text{CO}}: \min_{\boldsymbol{I}^t, f^t, p^t} & -\tilde{B}^t \cdot \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right) + V\left[\left(\mathcal{D}\left(\boldsymbol{I}^t, f^t, p^t\right) + \phi \cdot \mathbf{1}\left(\zeta^t = 1, I_{\text{d}}^t = 1\right)\right]\right. \\ \text{s.t.} \quad & (1), (12) - (18).\end{aligned} \tag{22}$$

Denote the feasible action set and the objective function of $\mathcal{P}_{\text{CO}}$ as $\mathcal{F}_{\text{CO}}^t$ and $J_{\text{CO}}^t\left(\boldsymbol{I}^t, f^t, p^t\right)$, respectively. For the time slots without computation task request, i.e., $\zeta^t = 0$, there is a single feasible solution for $\mathcal{P}_{\text{CO}}$ due to (13), which is given by $I_{\text{m}}^t = I_{\text{s}}^t = 0$, $I_{\text{d}}^t = 1$, $f^t = 0$, and $p^t = 0$. Thus, we will focus on the time slots with computation task requests in the following. First, we obtain the optimal CPU-cycle frequency for a task being executed locally at the mobile device by solving the following optimization problem $\mathcal{P}_{\text{ME}}$:

$$\mathcal{P}_{\text{ME}}: \min_{f^t} -\tilde{B}^t \cdot \kappa W \left(f^t\right)^2 + V \cdot \frac{W}{f^t}$$

$$\text{s.t.} \quad 0 < f^t \leq f_{\text{CPU}}^{\max} \tag{23}$$

$$\frac{W}{f^t} \leq \tau_d \tag{24}$$

$$\kappa W \left(f^t\right)^2 \in \left[E_{\min}, E_{\max}\right], \tag{25}$$

which is obtained by plugging $I_{\text{m}}^t = 1$, $I_{\text{s}}^t = I_{\text{d}}^t = 0$ and $p^t = 0$ into $\mathcal{P}_{\text{CO}}$, and using the fact that $f^t > 0$ for local execution. (24) is the execution delay constraint for mobile execution, and (25) is the CPU energy consumption constraint obtained by combining (14) and (18). We denote the objective function of $\mathcal{P}_{\text{ME}}$ as $J_{\text{m}}^t\left(f^t\right)$. Note that mobile execution is not necessarily feasible due to limited computation capability of the processing unit at the mobile device as indicated by (23). In the following proposition, we develop the feasibility condition and the optimal solution for $\mathcal{P}_{\text{ME}}$ given it is feasible.

***Proposition*** *2:* $\mathcal{P}_{\text{ME}}$ is feasible if and only if $f_L \leq f_U$, where $f_L = \max\{\sqrt{\frac{E_{\min}}{\kappa W}}, \frac{W}{\tau_d}\}$ and $f_U = \min\{\sqrt{\frac{E_{\max}}{\kappa W}}, f_{\max}\}$. If $\mathcal{P}_{\text{ME}}$ is feasible, its optimal solution is given by:

$$f^{t*} = \begin{cases} f_U, & \tilde{B}^t \geq 0 \text{ or } \tilde{B}^t < 0, f_0^t > f_U \\ f_0^t, & \tilde{B}^t < 0, f_L \leq f_0^t \leq f_U \\ f_L, & \tilde{B}^t < 0, f_0^t < f_L, \end{cases} \tag{26}$$

where $f_0^t = \left(\frac{V}{-2\tilde{B}^t\kappa}\right)^{\frac{1}{3}}$.

*Proof:* We first show the feasibility condition. Due to (24), $f^t$ should be no less than $W/\tau_d$ in order to meet the delay constraint. Besides, since the CPU energy consumption increases with $f^t$, the battery output energy constraint can be equivalently expressed as $\sqrt{\frac{E_{\min}}{\kappa W}} \leq f^t \leq \sqrt{\frac{E_{\max}}{\kappa W}}$. By incorporating (23), we rewrite the feasible CPU-cycle frequency set as $f_L = \max\{\sqrt{\frac{E_{\min}}{\kappa W}}, W/\tau_d\} \leq f^t \leq f_U = \min\{\sqrt{\frac{E_{\max}}{\kappa W}}, f_{\max}\}$, i.e., $\mathcal{P}_{\mathrm{ME}}$ is feasible if and only if $f_L \leq f_U$.

Next, we proceed to show the optimality of (26) when $\mathcal{P}_{\mathrm{ME}}$ is feasible. When $\tilde{B}^t \geq 0$, $J_{\mathrm{m}}^t\left(f^t\right)$ decreases with $f^t$, i.e., the minimum value is achieved by $f^t = f_U$. When $\tilde{B}^t < 0$, $J_{\mathrm{m}}^t\left(f^t\right)$ is convex with respect to $f^t$ as both $-\tilde{B}^t \kappa W \left(f^t\right)^2$ and $VW/f^t$ are convex functions of $f^t$. By taking the first-order derivative of $J_{\mathrm{m}}^t\left(f^t\right)$ and setting it to zero, we obtain a unique solution $f_0^t = \left(\frac{V}{-2\tilde{B}^t\kappa}\right)^{\frac{1}{3}} > 0$. If $f_0^t < f_L$, $J_{\mathrm{m}}^t\left(f^t\right)$ is increasing in $[f_L, f_U]$, and thus $f^{t*} = f_L$; if $f_0^t > f_U$, $J_{\mathrm{m}}^t\left(f^t\right)$ is decreasing in $[f_L, f_U]$, and thus $f^{t*} = f_U$; otherwise, if $f_L \leq f_0^t \leq f_U$, $J_{\mathrm{m}}^t\left(f^t\right)$ is decreasing in $[f_L, f_0^t]$ and increasing in $(f_0^t, f_U]$, and we have $f^{t*} = f_0^t$. ■

It can be seen from Proposition 2 that the optimal CPU-cycle frequency is chosen by balancing the cost of the harvested energy and the execution cost. Interestingly, we find that a higher CPU-cycle frequency, i.e., lower execution delay, can be supported with a greater amount of available harvested energy, which is because that the cost of renewable energy is reduced and more energy can be used to enhance the user's QoE, as demonstrated in Corollary 2.

***Corollary** 2:* The optimal CPU-cycle frequency for local execution $f^{t*}$ is independent with the channel gain $h^t$, and non-decreasing with the virtual energy queue length $\tilde{B}^t$.

*Proof:* Since $\mathcal{P}_{\mathrm{ME}}$ does not depend on $h^t$, the optimal CPU-cycle frequency is independent with the channel state. As $f_L$ and $f_U$ are constants independent with $\tilde{B}^t$, and $f_0^t$ increases with $\tilde{B}^t$ for $\tilde{B}^t < 0$, we can conclude that $f^{t*}$ is non-decreasing with $\tilde{B}^t$ based on (26). ■

Next, we will consider the case that the task is executed by the MEC server, where the optimal transmit power for computation offloading can be obtained by solving the following optimization problem $\mathcal{P}_{\mathrm{SE}}$:

$$\mathcal{P}_{\mathrm{SE}}: \min_{p^t} -\tilde{B}^t \cdot \frac{p^t L}{r\left(h^t, p^t\right)} + V \cdot \frac{L}{r\left(h^t, p^t\right)}$$

$$\text{s.t.} \quad 0 < p^t \leq p_{\mathrm{tx}}^{\max} \tag{27}$$

$$\frac{L}{r\left(h^t, p^t\right)} \leq \tau_d \tag{28}$$

$$\frac{p^t L}{r\left(h^t, p^t\right)} \in \left[E_{\min}, E_{\max}\right], \tag{29}$$

which is obtained by plugging $I_{\rm s}^t = 1$, $I_{\rm m}^t = I_{\rm d}^t = 0$ and $f^t = 0$ into $\mathcal{P}_{\rm CO}$, and using the fact that $p^t > 0$ for computation offloading. (28) and (29) stand for the execution delay constraint and the battery output energy constraint for MEC, respectively. We denote the objective function of $\mathcal{P}_{\rm SE}$ as $J_{\rm s}^t\left(p^t\right)$. Due to the wireless fading, it may happen that computation offloading is infeasible. In order to derive the feasibility condition and the optimal solution for $\mathcal{P}_{\rm SE}$ given it is feasible, we first provide the following lemma to facilitate the analysis.

***Lemma** 2:* For $h > 0$, $g_1\left(h,p\right) \triangleq \frac{p}{r(h,p)}$ is an increasing function of $p$ ($p > 0$) that takes value from $\left(\sigma \ln 2 \left(\omega h\right)^{-1}, +\infty\right)$.

*Proof:* The proof is omitted due to space limitation. ■

Based on Lemma 2, we combine constraints (27)-(29) into an inequality and obtain the feasibility condition for $\mathcal{P}_{\rm SE}$, as demonstrated in the following lemma.

***Lemma** 3:* $\mathcal{P}_{\rm SE}$ is feasible if and only if $p_L^t \leq p_U^t$, where $p_L^t$ and $p_U^t$ are defined as

$$p_L^t \triangleq \begin{cases} p_{L,\tau_d}^t, & \frac{\sigma L \ln 2}{\omega h^t} \geq E_{\min} \\ \max\{p_{L,\tau_d}^t, p_{E_{\min}}^t\}, & \frac{\sigma L \ln 2}{\omega h^t} < E_{\min} \end{cases} \text{ and } p_U^t \triangleq \begin{cases} \min\{p_{\rm tx}^{\max}, p_{E_{\max}}^t\}, & \frac{\sigma L \ln 2}{\omega h^t} < E_{\max} \\ 0, & \frac{\sigma L \ln 2}{\omega h^t} \geq E_{\max}, \end{cases} \tag{30}$$

respectively. In (30), $p_{L,\tau_d}^t \triangleq \left(2^{\frac{L}{\omega \tau_d}} - 1\right)\sigma / h^t$, $p_{E_{\min}}^t$ is the unique solution for $pL = r\left(h^t, p\right) E_{\min}$ given $\sigma L \ln 2 \left(\omega h^t\right)^{-1} < E_{\min}$, and $p_{E_{\max}}^t$ is the unique solution for $pL = r\left(h^t, p\right) E_{\max}$ given $\sigma L \ln 2 \left(\omega h^t\right)^{-1} < E_{\max}$.

*Proof:* The proof can be obtained based on Lemma 2, which is omitted for brevity. ■

We now develop the optimal solution for $\mathcal{P}_{\rm SE}$ as specified in the following proposition.

***Proposition** 3:* If $\mathcal{P}_{\rm SE}$ is feasible, i.e., $p_L^t \leq p_U^t$, its optimal solution is given by

$$p^{t*} = \begin{cases} p_U^t, & \tilde{B}^t \geq 0 \text{ or } \tilde{B}^t < 0, p_U^t < p_0^t \\ p_L^t, & \tilde{B}^t < 0, p_L^t > p_0^t \\ p_0^t, & \tilde{B}^t < 0, p_L^t \leq p_0^t \leq p_U^t, \end{cases} \tag{31}$$

where $p_0^t$ is the unique solution for equation $\Xi\left(h^t, p, \tilde{B}^t\right) = 0$ and $\Xi\left(h, p, \tilde{B}\right) \triangleq -\tilde{B}\log_2\left(1 + \frac{hp}{\sigma}\right) - \frac{h}{(\sigma + hp)\ln 2}\left(V - \tilde{B}p\right)$.

*Proof:* When $\tilde{B}^t \geq 0$, since both terms in $J_{\rm s}^t\left(p^t\right)$ are non-increasing with $p^t$, we have $p^{t*} = p_U^t$. When $\tilde{B}^t < 0$, we define $g_2\left(h, p, \tilde{B}\right) \triangleq -\frac{\tilde{B}p}{r(h,p)} + \frac{V}{r(h,p)}$, and thus

$$\frac{dg_2\left(h^t, p, \tilde{B}^t\right)}{dp} = \frac{-\tilde{B}^t \log_2\left(1 + \frac{h^t p}{\sigma}\right) - \frac{h^t}{(h^t p + \sigma)\ln 2}\left(-\tilde{B}^t p + V\right)}{\omega \log_2^2\left(1 + \frac{h^t p}{\sigma}\right)} \triangleq \frac{\Xi\left(h^t, p, \tilde{B}^t\right)}{\omega \log_2^2\left(1 + \frac{h^t p}{\sigma}\right)}. \tag{32}$$

Since $\frac{d\Xi\left(h^t,p,\tilde{B}^t\right)}{dp} > 0$, $\Xi\left(h^t,p,\tilde{B}^t\right)$ increases with $p$. In addition, as $\Xi\left(h^t,0,\tilde{B}^t\right) = -\frac{h^t V}{\sigma \ln 2} < 0$ and $\lim\limits_{p\rightarrow+\infty} \Xi\left(h^t,p,\tilde{B}^t\right) = +\infty$, there exists a unique $p_0^t \in (0,+\infty)$ satisfying $\Xi\left(h^t,p_0^t,\tilde{B}^t\right) = 0, \forall h^t > 0$. Since the denominator of (32) is positive for $h^t > 0$ and $p > 0$, $\frac{dg_2\left(h^t,p,\tilde{B}^t\right)}{dp} < 0$ for $p \in (0,p_0^t)$, i.e., $g_2\left(h^t,p,\tilde{B}^t\right)$ is decreasing, and $\frac{dg_2\left(h^t,p,\tilde{B}^t\right)}{dp} \geq 0$ for $p \in [p_0^t,+\infty)$, i.e., $g_2\left(h^t,p,\tilde{B}^t\right)$ is increasing. Consequently, when $\tilde{B}^t < 0$ and $p_L^t \leq p_0^t \leq p_U^t$, $J_{\mathrm{s}}^t\left(p^t\right)$ is non-increasing in $[p_L^t,p_0^t)$ while non-decreasing in $(p_0^t,p_U^t]$, and thus $p^{t*} = p_0^t$; when $\tilde{B}^t < 0$ and $p_L^t > p_0^t$, $J_{\mathrm{s}}^t\left(p^t\right)$ is non-decreasing in the feasible domain, and thus $p^{t*} = p_L^t$; otherwise when $\tilde{B}^t < 0$ and $p_U^t < p_0^t$, $J_{\mathrm{s}}^t\left(p^t\right)$ is non-increasing in the feasible domain, we have $p^{t*} = p_U^t$. ■

Similar to mobile execution, we find a monotonic behavior of the optimal transmit power for computation offloading, as shown in the following corollary.

***Corollary*** *3:* For a given $h^t$ such that $\mathcal{P}_{\mathrm{SE}}$ is feasible, the optimal transmit power for computation offloading $p^{t*}$ is non-decreasing with $\tilde{B}^t$.

*Proof:* Please refer to Appendix B. ■

***Remark*** *2:* We can see from (31) that the optimal transmit power for computation offloading depends on both the battery energy level and the channel state. In Corollary 3, we show a higher battery energy level awakes a higher transmit power, and thus incurs smaller execution latency. However, the monotonicity of $p^{t*}$ with respect to $h^t$ does not hold. This is due to the battery output energy constraint, which makes the feasible set of $p^t$ change with $h^t$.

Based on Proposition 2 and 3, the optimal computation offloading decision can be obtained by evaluating the optimal values of $\mathcal{P}_{\mathrm{CO}}$ for the three computation modes, i.e., dropping the task, mobile execution and MEC server execution, which can be explicitly expressed as

$$\langle \boldsymbol{I}^{t*}, f^{t*}, p^{t*} \rangle = \arg \min_{\langle \boldsymbol{I}^t, f^t, p^t \rangle \in \mathcal{F}_{\mathrm{CO}}^t} J_{\mathrm{CO}}\left(\boldsymbol{I}^t, f^t, p^t\right), \tag{33}$$

where $J_{\mathrm{CO}}\left(\boldsymbol{I}^t, f^t, p^t\right) = \mathbf{1}_{I_{\mathrm{m}}^t=1} J_{\mathrm{m}}^t\left(f^t\right) + \mathbf{1}_{I_{\mathrm{s}}^t=1} J_{\mathrm{s}}^t\left(p^t\right) + \mathbf{1}_{I_{\mathrm{d}}^t=1,\zeta^t=1} \cdot V\phi$, and $V\phi$ is the value of $J_{\mathrm{CO}}\left(\boldsymbol{I}^t, f^t, p^t\right)$ when a computation task is dropped. Note that when $\zeta^t = 1$ and $\mathcal{F}_{\mathrm{CO}}^t = \left\{\langle \left[I_{\mathrm{m}}^t = 0, I_{\mathrm{s}}^t = 0, I_{\mathrm{d}}^t = 1\right], 0, 0 \rangle\right\}$, the computation task has to be dropped, as $\mathcal{P}_{\mathrm{CO}}$ has only one feasible solution. It is also worth mentioning that bisection search can be applied to obtain $p_L^t$, $p_U^t$ and $p_0^t$, i.e., solving $\mathcal{P}_{\mathrm{CO}}$ is of low complexity.

## V. Performance Analysis

In this section, we will first prove the feasibility of the LODCO algorithm for $\mathcal{P}_2$, and the achievable performance of the proposed algorithm will then be analyzed.

## *A. Feasibility*

We verify the feasibility of the LODCO algorithm by showing that under the optimal solution for the per-time slot problem, the energy causality constraint in (8) is always satisfied, as demonstrated in the following proposition.

***Proposition** 4:* Under the optimal solution for the per-time slot problem, when $B^t < \tilde{E}_{\max}$, $I_{\mathrm{d}}^t = 1$, $I_{\mathrm{m}}^t = I_{\mathrm{s}}^t = 0$, $f^t = 0$, and $p^t = 0$, and the energy causality constraint in (8) will not be violated, i.e., the LODCO algorithm is feasible for $\mathcal{P}_2$ ($\mathcal{P}_1$).

*Proof:* When $B^t < \tilde{E}_{\max}$, we will show by contradiction that with the optimal computation offloading decision, $\mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right) = 0$. Suppose there exists an optimal computation offloading decision $\langle \boldsymbol{I}^{t*}, f^{t*}, p^{t*}\rangle$ with either $I_{\mathrm{m}}^{t*} = 1$ or $I_{\mathrm{s}}^{t*} = 1$. With this solution, due to the non-zero lower bound of the battery output energy, i.e., (18), the value of $J_{\mathrm{CO}}\left(\boldsymbol{I}^{t*}, f^{t*}, p^{t*}\right)$ will be no less than $-\tilde{B}^t E_{\min}$, which is greater than $V\phi$ as achieved by the solution with $I_{\mathrm{d}}^t = 1$, i.e., $\langle \boldsymbol{I}^{t*}, f^{t*}, p^{t*}\rangle$ is not optimal for the per-time slot problem. When $B^t \geq \tilde{E}_{\max}$, as $\max\limits_{\langle \boldsymbol{I}^t, f^t, p^t\rangle \in \mathcal{F}_{\mathrm{CO}}^t} \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right) = \tilde{E}_{\max}$, $\mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right) \leq B^t, \forall \langle \boldsymbol{I}^t, f^t, p^t\rangle \in \mathcal{F}_{\mathrm{CO}}^t$. Thus, (8) holds under the LODCO algorithm. ■

Based on the optimal energy harvesting decision and Proposition 4, we show the battery energy level is confined within an interval as shown in the following corollary.

***Corollary** 4:* Under the LODCO algorithm, the battery energy level at the mobile device $B^t$ is confined within $\left[0, \theta + E_H^{\max}\right], \forall t \in \mathcal{T}$.

*Proof:* The lower bound of $B^t$ is straightforward as the energy causality constraint is not violated according to Proposition 4. The upper bound of $B^t$ can be obtained based on the optimal energy harvesting in (21): Suppose $\theta < B^t \leq \theta + E_H^{\max}$, since $e^{t*} = 0$, we have $B^{t+1} \leq B^t \leq \theta + E_H^{\max}$; otherwise, if $B^t \leq \theta$, since $e^{t*} = E_H^t$, we have $B^{t+1} \leq B^t + e^{t*} \leq \theta + e^{t*} \leq \theta + E_H^{\max}$. Consequently, we have $B^t \in \left[0, \theta + E_H^{\max}\right], \forall t \in \mathcal{T}$. ■

As will be seen in the next subsection, the bounds of the battery energy level are useful for deriving the main result on the performance of the proposed algorithm. In addition, Corollary 4 indicates that, given the size of the available energy storage $C_B$, we can determine the control parameter $V$ as $\phi^{-1} \cdot \left(C_B - E_H^{\max} - \tilde{E}_{\max}\right) E_{\min}$, where $C_B$ should be greater than $\tilde{E}_{\max} + E_H^{\max}$ in order to guarantee $V > 0$. This is instructive for installation of EH and storage units at the mobile devices.

## B. Asymptotic Optimality

In this subsection, we will analyze the performance of the LODOC algorithm, where an auxiliary optimization problem $\mathcal{P}_3$ will be introduced to bridge the optimal performance of $\mathcal{P}_2$ and the performance achieved by the proposed algorithm. This will demonstrate the asymptotic optimality of the LODCO algorithm for $\mathcal{P}_1$ conjointly with Proposition 1.

Firstly, we define the Lyapunov function as

$$L\left(\tilde{B}^t\right) = \frac{1}{2}\left(\tilde{B}^t\right)^2 = \frac{1}{2}\left(B^t - \theta\right)^2. \tag{34}$$

Accordingly, the Lyapunov drift function and the Lyapunov drift-plus-penalty function can be expressed as

$$\Delta\left(\tilde{B}^t\right) = \mathbb{E}\left[L\left(\tilde{B}^{t+1}\right) - L\left(\tilde{B}^t\right)|\tilde{B}^t\right] \tag{35}$$

and

$$\Delta_V\left(\tilde{B}^t\right) = \Delta\left(\tilde{B}^t\right) + V\mathbb{E}\left[\mathcal{D}\left(\boldsymbol{I}^t, f^t, p^t\right) + \phi\cdot\mathbf{1}\left(\zeta^t = 1, I_{\mathrm{d}}^t = 1\right)|\tilde{B}^t\right], \tag{36}$$

respectively.

In the following lemma, we derive an upper bound for $\Delta_V\left(\tilde{B}^t\right)$, which will play an important part throughout the analysis of the LODCO algorithm.

***Lemma** 4:* For arbitrary feasible decision variables $e^t$, $\boldsymbol{I}^t$, $f^t$ and $p^t$ for $\mathcal{P}_2$, $\Delta_V\left(\tilde{B}^t\right)$ is upper bounded by

$$\Delta_V\left(\tilde{B}^t\right) \leq \mathbb{E}\left[B^t\left[e^t - \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right)\right] + V\left[\mathcal{D}\left(\boldsymbol{I}^t, f^t, p^t\right) + \phi\cdot\mathbf{1}\left(\zeta^t = 1, I_{\mathrm{d}}^t = 1\right)\right]|\tilde{B}^t\right] + C, \tag{37}$$

where $C = \frac{\left(E_H^{\max}\right)^2 + \left(\tilde{E}_{\max}\right)^2}{2}$.

*Proof:* Please refer to Appendix C. ■

Note that the terms inside the conditional expectation of the upper bound derived in Lemma 4 coincides with the objective function of the per-time slot problem in the LODCO algorithm. To facilitate the performance analysis, we define the following auxiliary problem $\mathcal{P}_3$:

$$\begin{aligned}\mathcal{P}_3:\ &\min_{\boldsymbol{I}^t, f^t, p^t, e^t}\lim_{T\to\infty}\frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T-1}\mathrm{cost}^t\right]\\ &\text{s.t.}\quad (1), (6), (12)-(18)\\ &\qquad\lim_{T\to+\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right) - e^t\right] = 0.\end{aligned} \tag{38}$$

In $\mathcal{P}_3$, the average harvested energy consumption equals the average harvested energy, i.e., the energy causality constraint in $\mathcal{P}_2$ is replaced by (38). Denote the optimal value of $\mathcal{P}_3$ as $\mathrm{EC}^*_{\mathcal{P}_3}$. In the following lemma, we will show that $\mathcal{P}_3$ is a relaxation of $\mathcal{P}_2$.

***Lemma** 5:* $\mathcal{P}_3$ is a relaxation of $\mathcal{P}_2$, i.e., $\mathrm{EC}^*_{\mathcal{P}_3} \leq \mathrm{EC}^*_{\mathcal{P}_2}$.

*Proof:* The proof can be obtained by showing any feasible solution for $\mathcal{P}_2$ is also feasible for $\mathcal{P}_3$, which is omitted for brevity. ■

Besides, in the following lemma, we show the existence of a stationary and randomized policy [34], where the decisions are i.i.d. among different time slots and depend only on $E_H^t$, $\zeta^t$ and $h^t$, that behaves arbitrarily close to the optimal solution of $\mathcal{P}_3$, meanwhile, the difference between $\mathbb{E}\left[e^t\right]$ and $\mathbb{E}\left[\mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right)\right]$ is arbitrarily small.

***Lemma** 6:* For an arbitrary $\delta > 0$, there exists a stationary and randomized policy $\Pi$ for $\mathcal{P}_3$, which decides $e^{t\Pi}$, $\boldsymbol{I}^{t\Pi}$, $f^{t\Pi}$ and $p^{t\Pi}$, such that (1), (6), (12)-(18) are met, and the following inequalities are satisfied:

$$\mathbb{E}\left[\mathcal{D}\left(\boldsymbol{I}^{t\Pi}, f^{t\Pi}, p^{t\Pi}\right) + \phi \cdot \mathbf{1}\left(\zeta^t = 1, I_{\mathrm{d}}^{t\Pi}\right)\right] \leq \mathrm{EC}^*_{\mathcal{P}_3} + \delta, t \in \mathcal{T}, \tag{39}$$

$$\left|\mathbb{E}\left[\mathcal{E}\left(\boldsymbol{I}^{t\Pi}, f^{t\Pi}, p^{t\Pi}\right) - e^{t\Pi}\right]\right| \leq \varrho\delta, t \in \mathcal{T}, \tag{40}$$

where $\varrho$ is a scaling constant.

*Proof:* The proof can be obtained by Theorem 4.5 in [34], which is omitted for brevity. ■

In Section IV, we bounded the optimal performance of the modified ECM problem $\mathcal{P}_2$ with that of the original ECM problem $\mathcal{P}_1$, while in Lemma 5, we showed the auxiliary problem $\mathcal{P}_3$ is a relaxation of $\mathcal{P}_2$. With the assistance of these results, next, we will provide the main result in this subsection, which characterizes the worst-case performance of the LODCO algorithm.

***Theorem** 1:* The execution cost achieved by the proposed LODCO algorithm, denoted as $\mathrm{EC}_{\mathrm{LODCO}}$, is upper bounded by

$$\mathrm{EC}_{\mathrm{LODCO}} \leq \mathrm{EC}^*_{\mathcal{P}_1} + \nu\left(E_{\min}\right) + C \cdot V^{-1}. \tag{41}$$

*Proof:* Please refer to Appendix D. ■

***Remark** 3:* Theorem 1 indicates that the execution cost upper bound can be made arbitrarily tight by letting $V \to +\infty$, $E_{\min} \to 0$, that is, the proposed algorithm asymptotically achieves the optimal performance of the original design problem $\mathcal{P}_1$. However, the optimal performance of $\mathcal{P}_1$ is achieved at the price of a higher battery capacity requirement and longer convergence time to the optimal performance. This is because that, the battery energy level will be stabilized around $\theta$ under the LODCO algorithm. As $E_{\min}$ decreases or $V$ increases, $\theta$ increases accordingly, and it will need a longer time to accumulate the harvested energy, which postpones the arrival of the system stability and hence delays the convergence. Thus, by adjusting the control parameters, we can balance the system performance and the battery capacity/convergence time. Similar phenomenon was observed in our previous work [25].

## VI. Simulation Results

In this section, we will verify the theoretical results derived in Section V and evaluate the performance of the proposed LODCO algorithm through simulations. In simulations, $E_H^t$ is uniformly distributed between 0 and $E_H^{\max}$ with the average EH power given by $P_H = E_H^{\max}(2\tau)^{-1}$, and the channel power gains are exponentially distributed with mean $g_0 d^{-4}$, where $g_0 = -40$ dB is the path-loss constant. In addition, $\kappa = 10^{-28}$, $\tau = \phi = 2$ ms, $w = 1$ MHz, $\sigma = 10^{-13}$ W, $p_{\text{tx}}^{\max} = 1$ W, $f_{\text{CPU}}^{\max} = 1.5$ GHz, $E_{\max} = 2$ mJ, and $L = 1000$ bits. Besides, $X = 5900$ cycles per byte, which corresponds to the workload of processing the English main page of Wikipedia [28]. Moreover, $P_H = 12$ mW, $d = 50$ m and $\tau_d = 2$ ms unless otherwise specified. For comparison, we introduce three benchmark policies, namely, *mobile execution with greedy energy allocation* (Mobile Execution (GD)), *MEC server execution with greedy energy allocation* (MEC Server Execution (GD)) and *dynamic offloading with greedy energy allocation* (Dynamic offloading (GD)), which minimize the execution cost at the current time slot. They work as follows:

- **Mobile Execution (GD):** Compute the maximum feasible CPU-cycle frequency as $f_U^t = \min\{f_{\text{CPU}}^{\max}, \sqrt{\frac{\min\{B^t, E_{\max}\}}{\kappa W}}\}$ when $\zeta^t = 1$. If $W/f_U^t \leq \tau_d$, the computation task will be executed locally with CPU-cycle frequency $f_U^t$; otherwise, mobile execution is infeasible and the task will be dropped. Note that computation offloading is disabled in this policy.
- **MEC Server Execution (GD):** When $\zeta^t = 1$, compute the maximum feasible transmit power as $p_U^t = \min\{p_{\text{tx}}^{\max}, p_{\min\{B^t, E_{\max}\}}^t\}$ if $\sigma L \ln 2\,(\omega h^t)^{-1} < \min\{B^t, E_{\max}\}$, where $p_{\min\{B^t, E_{\max}\}}^t$ is the unique solution of $pL = r\,(h^t, p)\min\{B^t, E_{\max}\}$. If $L/r\,(h^t, p_U^t) \leq \tau_d$, the computation task will be offloaded to the MEC server with transmit power $p_U^t$; otherwise, MEC server execution is infeasible and the computation task will be dropped. Note that the computation tasks are always offloaded to the MEC server in this policy.
- **Dynamic Offloading (GD):** When $\zeta^t = 1$, compute $f_U^t$ and $p_U^t$ as in the Mobile Execution (GD) and MEC Server Execution (GD) policies, respectively, and check if they can meet

the delay requirement. Then the feasible computation mode that incurs smaller execution delay will be chosen. If neither computation modes is feasible, the computation task will be dropped.

## A. Theoretical Results Verification

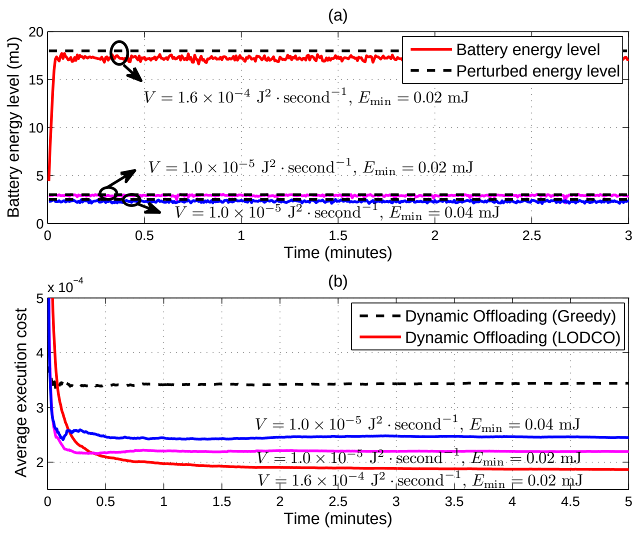


Fig. 2. Battery energy level and average execution cost vs. time, $\rho = 0.6$.

In this subsection, we will verify the feasibility and asymptotic optimality of the LODCO algorithm developed in Proposition 4, Corollary 4, and Theorem 1, respectively. The value of $\theta$ is chosen as the value of the right-hand side of (19). In Fig. 2(a), the battery energy level is depicted to demonstrate the feasibility of the LODOC algorithm for $\mathcal{P}_2$ ($\mathcal{P}_1$). First, we observe that the harvested energy keeps accumulating at the beginning, and finally stabilizes around the perturbed energy level. This is due to the fact that in the proposed algorithm the Lyapunov drift-plus-penalty function is minimized at each time slot. From the curves, with a larger value of $V$ or a smaller value of $E_{\min}$, the stabilized energy level becomes higher, which agrees with the definition of the perturbation parameter in (19). Also, we see that the energy level is confined within $[0, \theta + E_H^{\max}]$, which verifies Corollary 4 and confirms that the energy causality constraint is not violated, i.e., Proposition 4 holds. The evolution of the average execution cost with respect to time is shown in Fig. 2(b). We see that, a larger value of $V$ or a smaller value of $E_{\min}$ results in a smaller long-term average execution cost. Nevertheless, the algorithm converges more slowly to the stable performance. Besides, if $\langle E_{\min}, V\rangle$ are properly selected, the proposed algorithm will achieve significant performance gain compared to the benchmark policies.

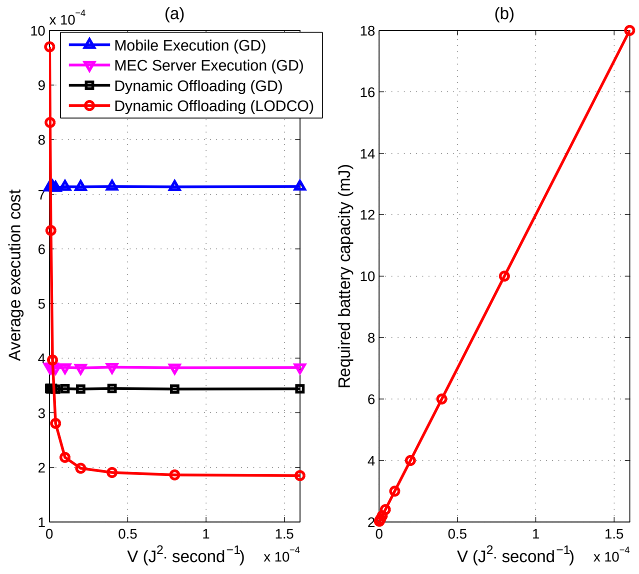
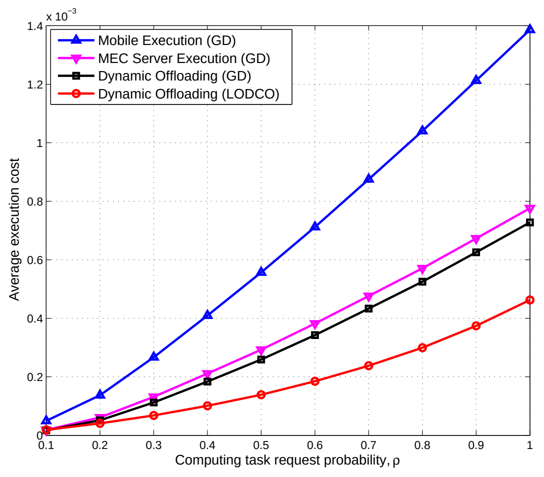


Fig. 3. Average execution cost and required battery capacity vs. $V$, $\rho = 0.6$ and $E_{\min} = 0.02$ mJ.

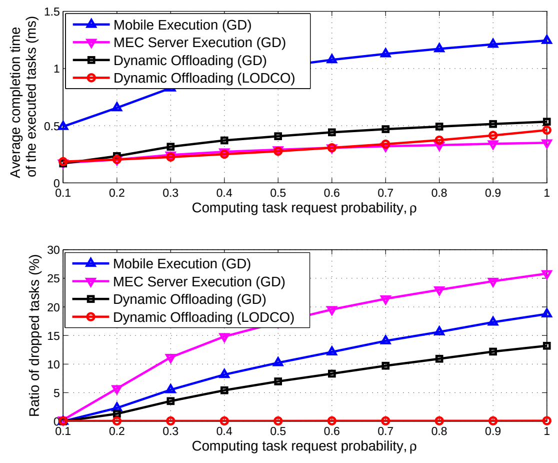
The relationship between the average execution cost/required battery capacity and $V$ is shown in Fig. 3. We see from Fig. 3(a) that the execution cost achieved by the proposed algorithm decreases inversely proportional to $V$, and eventually it converges to the optimal value of $\mathcal{P}_2$, which verifies the asymptotic optimality developed in Theorem 1. However, as shown from Fig. 3(b), the required battery capacity grows linearly with $V$ since the value of $\theta$ is linearly increasing with $V$. Thus, $V$ should be chosen to balance the achievable performance, convergence time and required battery capacity. For instance, if a battery with 18 mW capacity is available, we can choose $V = 1.6 \times 10^{-4}$ $\mathrm{J}^2 \cdot \mathrm{second}^{-1}$ for the LODCO algorithm, and then 74.4%, 51.8% and 46.3% performance gain compared to the Mobile Execution (GD), MEC Server Execution (GD) and Dynamic Offloading (GD) policies, respectively, will be obtained.

## B. *Performance Evaluation*

We will show the effectiveness of the proposed algorithm and demonstrate the impacts of various system parameters in this subsection. First, the impacts of the task request probability $\rho$ on the system performance, including the execution cost, the average completion time of the executed tasks and the task drop ratio, are illustrated in Fig. 4. We see in Fig. 4(a) that the



(a) Execution cost vs. $\rho$

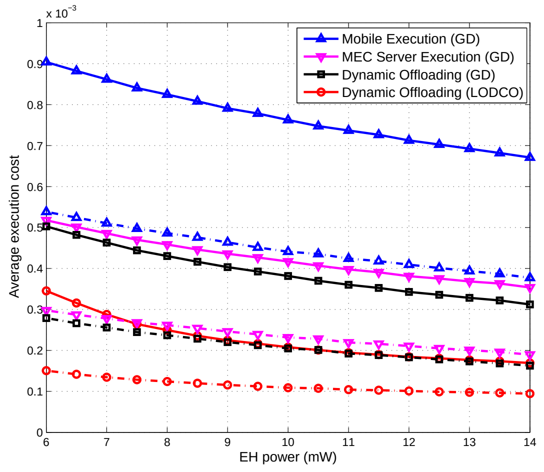


(b) Average completion time/task drop ratio vs. $\rho$

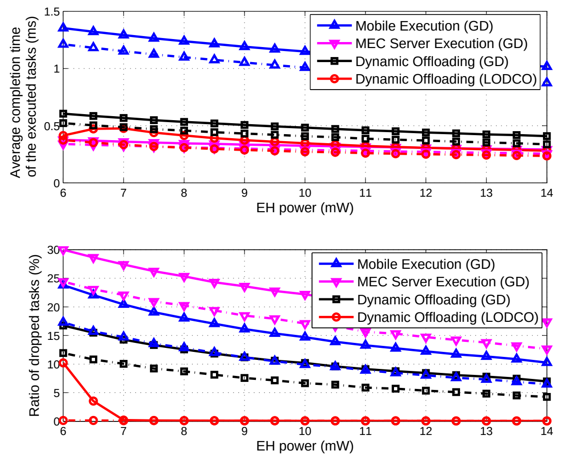
Fig. 4. System performance vs. task arrival probability.

execution cost increases with $\rho$, which is in accordance with our intuition. Besides, the LODCO algorithm achieves significant execution cost reduction compared to the benchmark policies. In Fig. 4(b), the average completion time of the executed tasks and the task drop ratio are shown, We see that the LODCO algorithm achieves a near-zero task drop ratio, while those achieved by the benchmark policies increase rapidly with $\rho$. In terms of the average completion time, the LODCO algorithm outperforms the benchmark policies when $\rho$ is small. However, when $\rho$ is large, the average completion time achieved by the LODCO algorithm is slightly longer than that achieved by the MEC Server Execution (GD) policy. The reason is, in order to minimize the execution cost, the LODCO algorithm suppresses the task drop ratio at the expense of a minor execution delay performance degradation.

The system performance versus the EH rate, i.e., $P_H$, is shown in Fig. 5, where the effectiveness of the LODCO algorithm is again validated. In addition, we see the execution cost decreases as the EH rate increases since consuming the renewable energy incurs no cost. Similar to the execution cost, the task drop ratios achieved by different policies decrease with the EH rate. Interestingly, under the LODCO algorithm, an increase of the EH rate does not necessarily reduce the average completion time, e.g., when $\rho = 0.6$ and $P_H$ increases from $6$ to $7$ mW, the LODCO algorithm has introduced a $0.1$ ms extra average completion time, but secured a 10% task drop reduction. Since the optimization objective is the execution cost, eliminating task drops brings more benefits in terms of system cost when the system resource is scarce, i.e., the



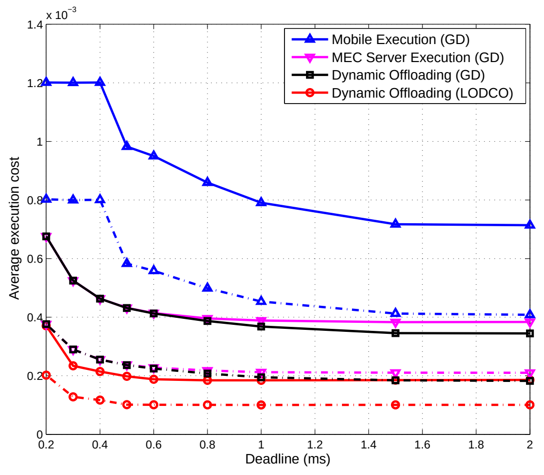
(a) Execution cost vs. $P_H$



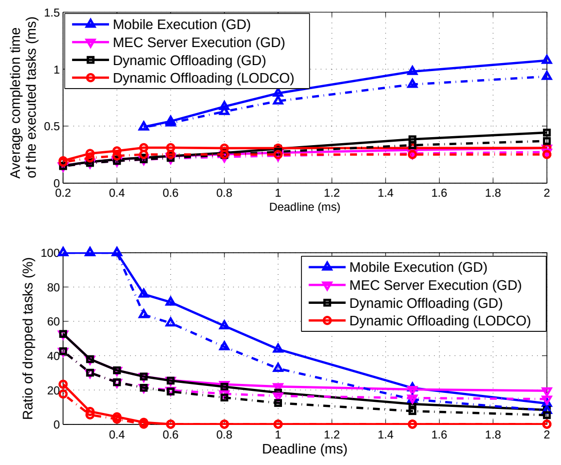
(b) Average completion time/task drop ratio vs. $P_H$

Fig. 5. System performance vs. EH rate, the solid lines corresponds to $\rho = 0.6$ and the dash-solid lines corresponds to $\rho = 0.4$.

harvested energy is insufficient compared to the relatively intense computation workload.



(a) Execution cost vs. $\tau_d$

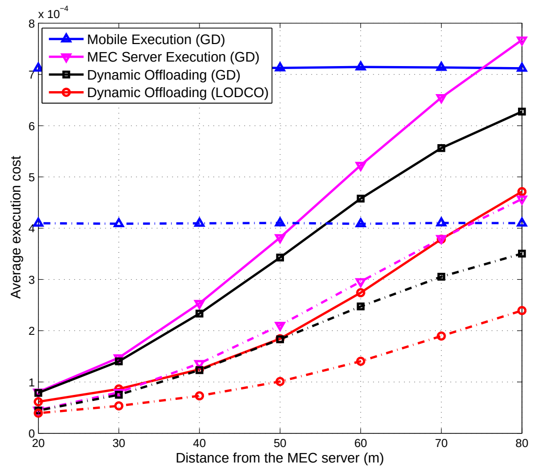


(b) Average completion time/task drop ratio vs. $\tau_d$

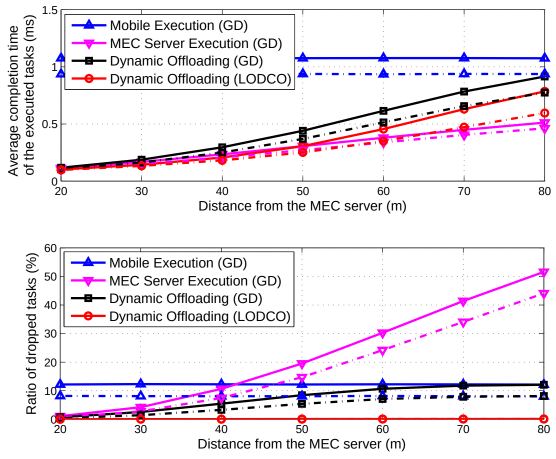
Fig. 6. System performance vs. execution deadline, the solid lines corresponds to $\rho = 0.6$ and the dash-solid lines corresponds to $\rho = 0.4$.

In Fig. 6, we reveal the relationship of between the execution deadline $\tau_d$ and the system performance. As $\tau_d$ decreases, i.e., the computation requirement becomes more stringent, the execution cost, average completion time and task drop ratio achieved by all four policies increase. It can be seen that when $\tau_d \leq 0.4$ ms, the execution cost achieved by the Mobile Execution (GD) policy becomes a constant $\rho\phi$, and the task drop ratio is 100%. Meanwhile, the MEC

Server Execution (GD) and the Dynamic Offloading (GD) policies converge. In these scenarios, the mobile device is not able to conduct any computation because of hardware limitation, i.e., $f^t \leq f_{\mathrm{CPU}}^{\max} = 1.5$ GHz, and all the computation tasks have to be offloaded to the MEC server for MEC. The results in Fig. 6(b) confirms the benefits of MEC as around 50% tasks are successfully executed for $\tau_d = 0.2$ ms even under the greedy offloading policy. Note that for a small value of $\tau_d$, e.g., $\tau_d \leq 0.8$ ms, the average completion time achieved by the LODCO algorithm is slightly longer than those of the other two policies with computation offloading, but the task drop ratio is reduced noticeably by more than 20%. This phenomenon is similar to what was observed in Fig. 4(b), where the LODCO algorithm tends to avoid dropping tasks by prolonging the average completion time in order to achieve a minimum execution cost.



(a) Execution cost vs. $d$



(b) Average completion time/task drop ratio vs. $d$

Fig. 7. System performance vs. distance, the solid lines corresponds to $\rho = 0.6$ and the dash-solid lines corresponds to $\rho = 0.4$.

Finally, we show the relationship between the system performance and $d$, i.e., the distance from the mobile device to the MEC server, in Fig. 7. The performance of the computation offloading policies, including the MEC Server Execution (GD), Dynamic offloading (GD) and the LODCO algorithms, deteriorate as $d$ becomes large. As can be seen from Fig. 7(a), when the mobile device is close to the MEC server, the three computation offloading policies converge and greatly outperform the Mobile Execution (GD) policy. In such scenarios, the mobile device is able to offload the computation tasks to the MEC server with a small amount of harvested energy due to small path loss. With a large value of $d$, e.g., $d = 80$ m, offloading the tasks greedily cannot bring any execution cost reduction compared the Mobile Execution (GD) policy, while the LODCO algorithm offers more than 40% performance gain. From Fig. 7(b), we see that although the MEC Server Execution (GD) policy incurs the least completion time for the executed tasks, its task failure performance sharply degrades. In contrast, the proposed LODCO algorithm achieves a near-zero task drop ratio with an improved completion time performance compared to the Mobile Execution (GD) and Dynamic Offloading (GD) policies.

## VII. Conclusions

In this paper, we investigated mobile-edge computing (MEC) systems with EH mobile devices. The execution cost, which addresses the execution delay and task failure, was adopted as the performance metric. A dynamic computation offloading policy, namely, the Lyapunov optimization-based dynamic computation offloading (LODCO) algorithm, was then developed. It is a low-complexity online algorithm and requires little prior knowledge. We found the monotonic properties of the CPU-cycle frequencies (transmit power) for mobile execution (computation offloading) with respect to the battery energy level, which uncovers the impact of EH to the system operations. Performance analysis was conducted which revealed the asymptotic optimality of the proposed algorithm. Simulation results showed that the proposed LODCO algorithm not only significantly outperforms the benchmark greedy policies in terms of execution cost, but also reduces computation failures noticeably at an expense of minor execution delay performance degradation. Our study provides a viable approach to design future MEC systems with renewable energy-powered devices. It would be interesting to extend the proposed algorithm to more general MEC systems with multiple mobile devices, as well as consider resource-limited MEC servers. Another extension is to combine the concepts of wireless energy transfer and energy harvesting by deploying a power beacon co-located with the MEC server so that the energy deficit incurred by the renewable energy sources can be compensated by the controllable radio frequency energy.

## Appendix

### A. Proof for Proposition 1

Since $\mathcal{P}_2$ is a tightened version of $\mathcal{P}_1$, we have $\mathrm{EC}^*_{\mathcal{P}_1} \leq \mathrm{EC}^*_{\mathcal{P}_2}$. The other side of the inequality can be obtained by constructing a feasible solution for $\mathcal{P}_2$ (denoted as $\langle e^t_{\mathcal{P}_2}, \boldsymbol{I}^t_{\mathcal{P}_2}, f^t_{\mathcal{P}_2}, p^t_{\mathcal{P}_2}\rangle$) based on the optimal solution for $\mathcal{P}_1$ (denoted as $\langle e^t_{\mathcal{P}_1}, \boldsymbol{I}^t_{\mathcal{P}_1}, f^t_{\mathcal{P}_1}, p^t_{\mathcal{P}_1}\rangle$[8]): **i)** If $\mathcal{E}\left(\boldsymbol{I}^t_{\mathcal{P}_1}, f^t_{\mathcal{P}_1}, p^t_{\mathcal{P}_1}\right) \in$

[8]For simplicity, we assume the optimal solution for $\mathcal{P}_1$ satisfies the property of the optimal CPU-cycle frequencies in Lemma 1.

$(0, E_{\min})$, then the computation task will be dropped in the constructed solution and no harvested energy will be consumed, i.e., $\mathrm{cost}^t_{\mathcal{P}_2} = \phi$; **ii)** If $\mathcal{E}\left(\boldsymbol{I}^t_{\mathcal{P}_1}, f^t_{\mathcal{P}_1}, p^t_{\mathcal{P}_1}\right) \in [E_{\min}, E_{\max}]$, the constructed solution for the $t$th time slot will be the same as the optimal solution for $\mathcal{P}_1$; **iii)** The EH decision $e^t_{\mathcal{P}_2}$ is determined by $e^t_{\mathcal{P}_2} = \max\{B^t_{\mathcal{P}_1} - \mathcal{E}\left(\boldsymbol{I}^t_{\mathcal{P}_1}, f^t_{\mathcal{P}_1}, p^t_{\mathcal{P}_1}\right) + e^t_{\mathcal{P}_1} - B^t_{\mathcal{P}_2} + \mathcal{E}\left(\boldsymbol{I}^t_{\mathcal{P}_2}, f^t_{\mathcal{P}_2}, p^t_{\mathcal{P}_2}\right), 0\}$.

It is not difficult to show $B^t_{\mathcal{P}_1} \leq B^t_{\mathcal{P}_2} < +\infty$, and thus the constructed solution is feasible to $\mathcal{P}_2$. If $E_{\min} \geq E^{\tau_d}_{\min}$, where $E^{\tau}_{\min} = \kappa W^3 \tau_d^{-2}$ is the minimum amount of energy required to meet the deadline constraint for mobile execution, for a time slot with $I^t_{\mathrm{m},\mathcal{P}_1} = 1$ and $\mathcal{E}\left(\boldsymbol{I}^t_{\mathcal{P}_1}, f^t_{\mathcal{P}_1}, p^t_{\mathcal{P}_1}\right) \in (0, E_{\min})$, the constructed solution incurs $(\phi - \tau_{E_{\min}})$ units of extra execution cost in the worst case. Here, $\tau_{E_{\min}} = \kappa^{\frac{1}{2}} W^{\frac{3}{2}} E_{\min}^{-\frac{1}{2}}$ is the execution delay corresponds to $E_{\min}$ amount of energy consumption for mobile execution; otherwise, if $E_{\min} < E^{\tau_d}_{\min}$, $I^t_{\mathrm{m},\mathcal{P}_1} = 1$ and $\mathcal{E}\left(\boldsymbol{I}^t_{\mathcal{P}_1}, f^t_{\mathcal{P}_1}, p^t_{\mathcal{P}_1}\right) \in (0, E_{\min})$ is infeasible as the deadline constraint cannot be met. Besides, the probability of offloading a task to the MEC server successfully with energy consumption less than $E_{\min}$ is no greater than $\mathbb{P}\{\omega\tau_d \log_2\left(1 + \frac{h^t p^t}{\sigma}\right) \geq L\} = 1 - F_H(\eta)$, where $\eta \triangleq \left(2^{\frac{L}{\omega\tau_d}} - 1\right)\tau_d \sigma E^{-1}_{\min}$, and the constructed solution will incur at most $\phi$ units of extra execution cost as $\mathrm{cost}^t_{\mathcal{P}_1} > 0$. By further incorporating the task request probability $\rho$, we can obtain the desired result.

## B. *Proof for Corollary 3*

For $\tilde{B}^t < 0$, since $\Xi\left(h^t, p^t_0, \tilde{B}^t\right) = 0$, with some manipulations, we have $\tilde{B}^t \cdot k\left(h^t, p^t_0\right) = \frac{h^t V}{\ln 2}$, where $k(h,p) = \frac{hp}{\ln 2} - (hp + \sigma)\log_2\left(1 + \frac{hp}{\sigma}\right)$, and $\frac{\partial k(h,p)}{\partial p} = -h\log_2\left(1 + \frac{hp}{\sigma}\right) < 0$, i.e., $k(h,p)$ decreases with $p$ for $p > 0$. Denote $\tilde{B}^t_- < \tilde{B}^t_+ < 0$ and the corresponding solutions for $\Xi\left(h^t, p, \tilde{B}^t\right) = 0$ as $p^t_{0,-}$ and $p^t_{0,+}$, respectively. Since $\tilde{B}^t_+ k\left(h^t, p^t_{0,+}\right) = \tilde{B}^t_- k\left(h^t, p^t_{0,-}\right) > 0$, we have $k\left(h^t, p^t_{0,+}\right) < k\left(h^t, p^t_{0,-}\right) < 0$, i.e., $p^t_{0,+} > p^t_{0,-}$. Since $p^t_L$ and $p^t_U$ are invariant with $\tilde{B}^t$, according to (31), $p^{t*}$ is non-decreasing with $\tilde{B}^t$ for $\tilde{B}^t < 0$. Besides, as $p^{t*} = p^t_U$ when $\tilde{B}^t > 0$, we can conclude that $p^{t*}$ is non-decreasing with $\tilde{B}^t$.

## C. *Proof for Lemma 4*

By subtracting $\theta$ at both sides of (9), we have $\tilde{B}^{t+1} = \tilde{B}^t + e^t - \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right)$. Squaring both sides of this equality, we have

$$\begin{aligned}\left(\tilde{B}^{t+1}\right)^2 &= \left(\tilde{B}^t + e^t - \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right)\right)^2 \\ &\leq \left(\tilde{B}^t\right)^2 + 2\tilde{B}^t\left(e^t - \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right)\right) + \left(e^t\right)^2 + \mathcal{E}^2\left(\boldsymbol{I}^t, f^t, p^t\right) \\ &\leq \left(\tilde{B}^t\right)^2 + 2\tilde{B}^t\left(e^t - \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right)\right) + \left(E^{\max}_H\right)^2 + \tilde{E}^2_{\max}.\end{aligned} \tag{42}$$

Dividing both sides of (42) by $2$, adding $V\left[\mathcal{D}\left(\boldsymbol{I}^t, f^t, p^t\right) + \phi \cdot \mathbf{1}\left(\zeta^t = 1, I_{\mathrm{d}}^t = 1\right)\right]$, as well as taking the expectation conditioned on $\tilde{B}^t$, we can obtain the desired result.

## D. Proof for Theorem 1

Since the LODCO algorithm obtains the optimal solution of the per-time slot problem, (43) holds, where $\mathrm{cost}^{t*}$ and $\mathrm{cost}^{t\Pi}$ are the execution cost at the $t$th time slot under $\langle \boldsymbol{I}^{t*}, f^{t*}, p^{t*} \rangle$ and $\langle \boldsymbol{I}^{t\Pi}, f^{t\Pi}, p^{t\Pi} \rangle$, respectively. (†) is because that policy $\Pi$ is independent of the battery energy level $B^t$, and (‡) is due to Corollary 4 and Lemma 6.

$$\begin{aligned} \Delta_V\left(\tilde{B}^t\right) &\leq \mathbb{E}\left[\tilde{B}^t\left[e^{t*} - \mathcal{E}\left(\boldsymbol{I}^{t*}, f^{t*}, p^{t*}\right)\right] + V \cdot \mathrm{cost}^{t*} | \tilde{B}^t\right] + C \\ &\leq \mathbb{E}\left[\tilde{B}^t\left[e^{t\Pi} - \mathcal{E}\left(\boldsymbol{I}^{t\Pi}, f^{t\Pi}, p^{t\Pi}\right)\right] + V \cdot \mathrm{cost}^{t\Pi} | \tilde{B}^t\right] + C \\ &\overset{(\dagger)}{=} \tilde{B}^t \mathbb{E}\left[e^{t\Pi} - \mathcal{E}\left(\boldsymbol{I}^{t\Pi}, f^{t\Pi}, p^{t\Pi}\right)\right] + V \cdot \mathbb{E}\left[\mathrm{cost}^{t\Pi}\right] + C \\ &\overset{(\ddagger)}{\leq} \max\{\theta, E_H^{\max}\} \cdot \varrho\delta + V\left(\mathrm{EC}_{\mathcal{P}_3} + \delta\right) + C. \end{aligned} \tag{43}$$

By letting $\delta$ go to zero, we obtain

$$\Delta_V\left(\tilde{B}^t\right) \leq V \mathrm{EC}_{\mathcal{P}_3}^* + C. \tag{44}$$

Taking the expectation on both sides of (44), summing up the inequalities for $t = 0, \cdots T-1$, dividing by $T$ and letting $T$ go to infinity, we have $\mathrm{EC}_{\mathrm{LODCO}} \leq \mathrm{EC}_{\mathcal{P}_3}^* + \frac{C}{V}$. By further utilizing Proposition 1 and Lemma 5, the theorem is proved.

## References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswmi, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *ELSEVIER Future Gener. Compt. Syst.*, vol. 29, no. 7, pp. 1645-1660, Sep. 2013.

[2] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, “A survey of computation offloading for mobile systems,” *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129-140, Feb. 2013.

[3] European Telecommunications Standards Institute (ETSI), “Mobile-edge computing-Introductory technical white paper,” Sep. 2014.

[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14-23, Oct. 2009.

[5] K. Kumar and Y. H. Lu, “Cloud computing for mobile users: Can offloading computation save energy?,” *Computer*, vol. 43, no. 4, pp. 51-56, Apr. 2010.

[6] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, “Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks,” *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45-55, Nov. 2014.

[7] S. Lambert *et al.*, "World-wide electricity consumption of communication networks," *Optical Express*, vol. 20, no. 26, pp. B513-524, Mar. 2012.

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

[9] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, and K. Huang, "Energy harvesting wireless communications: A review of recent advances," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 3, pp. 360-381, Mar. 2015.

[10] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. ACM Int. Conf. Mobile Syst., Appl., Serv. (MobiSys)*, San Francisco, CA, Jun. 2010, pp. 49-62.

[11] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Orlando, FL, Mar. 2012, pp. 945-953.

[12] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, vol. 1991-1995, Jun. 2012.

[13] O. Munoz, A. Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738-4755, Oct. 2015.

[14] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974-983, Apr. 2015.

[15] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *Kluwer J. VLSI Signal Process. Syst.*, vol. 13, no. 2/3, pp. 203-221, Aug. 1996.

[16] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 4569-4581, Sep. 2013.

[17] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89-103, Jun. 2015.

[18] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510-2523, Dec. 2015.

[19] G. Piro *et al.*, "Hetnets powered by renewable energy sources: Substainable next generation cellular networks," *IEEE Internet Comput.*, vol. 17, no. 1, pp. 32-39, Jan. 2013.

[20] Y. Mao, Y. Luo, J. Zhang, and K. B. Letaief, "Energy harvesting small cell networks: Feasibility, deployment and operation," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 94-101, Jun. 2015.

[21] O. Ozel, K. Tutuncuoglu, J. Yang, S. Ulukus and A. Yener, "Transmission with energy harvesting nodes in fading wireless channels: Optimal policies," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 8, pp. 1732-1743, Sep. 2011.

[22] L. Huang and M. J. Neely, "Utility optimal scheduling in energy-harvesting networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1117-1130, Aug. 2013.

[23] D. W. K. Ng, E. S. Lo, and R. Schober, "Energy-efficient resource allocation in OFDMA systems with hybrid energy harvesting base station," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3412-3427, Jul. 2013.

[24] J. Gong, J. S. Thompson, S. Zhou, and Z. Niu, "Base station sleeping and resource allocation in renewable energy powered cellular networks," *IEEE Trans. Commun.*, vol. 62, no. 11, pp. 3801-3813, Nov. 2014.

[25] Y. Mao, J. Zhang, and K. B. Letaief, "A Lyapunov optimization approach for green cellular networks with hybrid energy supplies," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2463-2477, Dec. 2015.

[26] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, to appear.

[27] Z. Jiang and S. Mao, “Energy delay trade-off in cloud offloading for mutli-core mobile devices,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, Dec. 2015, pp. 1-6.

[28] A. P. Miettinen and J. K. Nurminen, “Energy efficiency of mobile clients in cloud computing,” in *Proc. 2010 USENIX Conference on Hot Topics in Cloud Computing. (HotCloud)*, Jun. 2010, pp. 1-7.

[29] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Upper Saddle River, New Jersy, USA: Prentice Hall, 2003.

[30] S. Lakshminarayana, T. Q. S. Quek and H. V. Poor, “Cooperation and storage tradeoffs in power grids with renewable energy resources,” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 7, pp. 1386-1397, Jul. 2014.

[31] S. Sun, M. Dong, and B. Liang, “Distributed real-time power balancing in renewable-integrated power grids with storage and flexible loads,” *IEEE Trans. Smart Grid*, to appear.

[32] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmonth, MA, USA: Athens Scientific, 2005.

[33] M. J. Neely and L. Huang, “Dynamic product assembly and inventory control for maximum profit,” *IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, Dec. 2010, pp. 2805-2812.

[34] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Calypool., 2010.