energies

MDPI

Article

Two-Stage Computation Offloading Scheduling Algorithm for Energy-Harvesting Mobile Edge Computing

Laihyuk Park [1], Cheol Lee [2], Woongsoo Na [3], Sungyun Choi [4,*] and Sungrae Cho [2,*]

[1] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea; lhpark@seoultech.ac.kr
[2] School of Computer Science and Engineering, Chung-Ang University, 221 Heukseok, Dongjak, Seoul 156-756, Korea; clee@uclab.re.kr
[3] Media Intellectualization Research Section, Electronics and Telecommunications Research Institute, 218 Gajeong-ro, Yuseong-gu, Daejeon 34129, Korea; wsna@etri.re.kr
[4] School of Electrical Engineering, Korea University, 145 Anam-ro, Seongbuk-gu, Seoul 02841, Korea
* Correspondence: sungyun@korea.ac.kr (S.C.); srcho@cau.ac.kr (S.C.)

Received: 6 September 2019; Accepted: 12 November 2019; Published: 15 November 2019

Abstract: Recently, mobile edge computing (MEC) technology was developed to mitigate the overload problem in networks and cloud systems. An MEC system computes the offloading computation tasks from resource-constrained Internet of Things (IoT) devices. In addition, several convergence technologies with renewable energy resources (RERs) such as photovoltaics have been proposed to improve the survivability of IoT systems. This paper proposes an MEC integrated with RER system, which is referred to as energy-harvesting (EH) MEC. Since the energy supply of RERs is unstable due to various reasons, EH MEC needs to consider the state-of-charge (SoC) of the battery to ensure system stability. Therefore, in this paper, we propose an offloading scheduling algorithm considering the battery of EH MEC as well as the service quality of experience (QoE). The proposed scheduling algorithm consists of a two-stage operation, where the first stage consists of admission control of the offloading requests and the second stage consists of computation frequency scheduling of the MEC server. For the first stage, a non-convex optimization problem is designed considering the computation capability, SoC, and request deadline. To solve the non-convex problem, a greedy algorithm is proposed to obtain approximate optimal solutions. In the second stage, based on Lyapunov optimization, a low-complexity algorithm is proposed, which considers both the workload queue and battery stability. In addition, performance evaluations of the proposed algorithm were conducted via simulation. However, this paper has a limitation in terms of verifying in a real-world scenario.

Keywords: computation offloading; mobile edge computing; energy harvesting; lyapunov optimization

1. Introduction

In recent years, along with the development of the Internet of Things (IoT) technology, it has become easier to connect mobile devices to the Internet [1–3]. In particular, IoT-based sensor devices, which have low computing performance, can overcome their computational limitations with the help of cloud systems [4,5]. However, the explosive growth of IoT data has resulted in increased traffic load in networks and cloud systems. This overload reduces the quality of experience (QoE) of the services and can result in network blackout, which shuts down the network system [6,7]. To solve these problems, the mobile edge computing (MEC) technology, which is a type of radio access network (RAN) with cloud computing capabilities, has been developed to assist resource-constrained IoT

devices [6,7]. In an MEC environment, the MEC server computes the offloaded workload from the IoT devices and charges the bill accordingly. This paper proposes an energy harvesting (EH) MEC system, which enhances the survivability of the MEC through energy harvesting if the MEC is installed in a remote area where grid power supply is not available. In the EH MEC system, the MEC server is powered by renewable energy resources (RER) such as photovoltaic or wind turbine resources. Therefore, it is possible to establish a system that deploys IoT sensor nodes and collects information in places where it is difficult to install electricity facilities such as deserts or unmanned islands. However, the EH MEC system has important challenges in terms of stability. On the one hand, since the energy supply from RERs is uncertain with respect to weather or time, the EH MEC system cannot achieve stable energy unlike conventional grid powered MEC. On the other hand, if EH MEC operation only considers maximization of the system performance, system black out will occur since the power supply of EH MEC is unstable, i.e., EH MEC has to consider battery stability [8]. Thus, EH MEC reduces the energy consumption when the amount of harvested energy is not sufficient even if the system performance is decreased.

This paper proposes an EH MEC scheduling algorithm that considers the battery stability. Figure 1 shows the proposed MEC system. As shown in the figure, the IoT devices transmit the offloading requests to the MEC system. Then, the MEC system determines the admission of requests (referred to as offloading scheduling). If the offloading request is accepted, it is executed by the MEC server. Otherwise, it is transferred to the cloud system. In addition, the MEC server determines its computation frequency based on the battery state-of-charge (SoC) (referred to as MEC scheduling). According to circuit theories, the CPU power is dominated by the dynamic power, which originates from the toggling activities of the logic gates inside the CPU [9]. Thus, in this paper, we assume that the power consumption of the computation offloading can be handled by CPU frequency scheduling such as dynamic voltage frequency scaling (DVFS) [9,10]. In the proposed system, if the SoC of the battery is sufficient, the MEC server raises the computation frequency for faster offloading service. Otherwise, the MEC server will lower the frequency to ensure system stability, i.e. to avoid blackout.

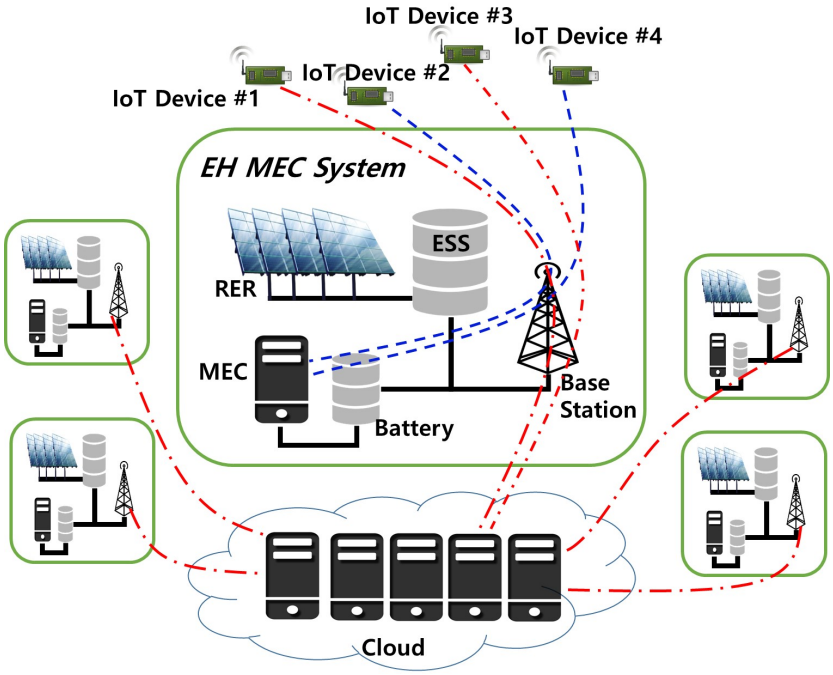

Figure 1. Architectural view of the proposed EH MEC system, consisting of RERs, energy storage system (ESS), a base station, MEC, etc. The IoT devices send their computational workloads to the base station. If the computation can be performed by the EH MEC system, it is offloaded to the MEC. Otherwise, it is offloaded to the cloud system.

The contributions of this paper are summarized as follows:

- EH MEC system: We propose an MEC integrated with RER to improve the survivability of the system through energy harvesting.The mathematical model of the MEC system includes a battery state, energy harvesting, offloading request and request deadline.
- Real-time Scheduling: We propose a two-stage scheduling algorithm, consisting of offloading and MEC scheduling. The optimization problems of the two types of scheduling are modeled as NP-hard problems. Therefore, they cannot be solved in non-deterministic polynomial time. To solve this problem, we propose a greedy algorithm to find approximate solutions for offloading scheduling. For MEC scheduling, a Lyapunov optimization-based scheduling algorithm is proposed to find the optimal computation frequency in real time.
- Battery Stability: As aforementioned, the EH MEC system has to consider the battery stability to ensure a stable operation. If the SoC is sufficient, the proposed offloading scheduler increases the number of permitted requests. Similarly, the MEC scheduler controls the frequency based on the battery SoC.

The rest of this paper is organized as follows. In Section 2, we review related works on EH MEC systems. In Section 3, the scheduling algorithms are proposed, and the corresponding optimization problem is presented. The design of the greedy algorithm, which can obtain approximate optimal solutions for offloading scheduling, and a low complexity MEC frequency scheduling algorithm based on Lyapunov optimization are provided in Section 4. Section 5 presents the performance evaluation. The conclusions of this paper are drawn in Section 6.

2. Related Work

In this section, we review the existing related works to clarify the motivation of the proposed algorithm. Over the past several years, MEC researchers have focused on maximizing computational efficiency and minimizing energy consumption of IoT devices [11–16]. In [11], the authors proposed joint offloading and computation energy efficiency maximization algorithm for MEC system. They proposed novel computation efficiency indicator and solved the problem by using iterative and gradient descent method. However, this algorithm did not consider energy harvesting. In [12], energy efficient task offloading algorithms for non-orthogonal multiple access (NOMA) MEC environment is presented. This algorithm determined the uplink power control solution, and then solved the task offloading partition and time allocation. However, this scheme also did not consider energy harvesting. The authors of [13] proposed the delay constraint offloading algorithm. This algorithm solved the problem of minimizing energy consumption, assuming that MEC server can charge IoT devices. In [14], the authors proposed a bound improving branch and bound approach to minimize energy consumption of IoT device in which orthogonal frequency division multiple access (OFDMA) was considered for uplink transmission. They focused on the energy consumption of the IoT device, under the consideration of computation offloading, subcarrier allocation, and computing resource allocation. In [15], an IoT device offloading scheduling algorithm in wireless power transfer environment is proposed. In this algorithm, the IoT device decides whether to compute the task itself or offload the task to MEC server. A cooperative partial computation offloading algorithm is proposed for MEC and cloud server environment in [16]. The authors proposed the branch and bound approach to solve the single MEC scenario, and then expanded it to multiple MEC and cloud scenario. For the multiple scenarios, an iterative heuristic MEC resource allocation algorithm was proposed. The authors of [9] proposed an algorithm for computation offloading scheduling of MEC server. They minimized the energy consumption of the MEC server as well as guaranteed the stability of the task buffer. However, it is difficult to apply this algorithm in EH environment since they did not consider energy harvesting. These previous studies [11–16] focused on the energy efficiency of IoT devices. However, this paper focuses on the energy efficiency scheduling of MEC server, i.e., MEC server has to provide stable offloading service. Due to the advancement in energy harvesting techniques, research about IoT

systems with energy harvesting attracts significant attention [10,17–19]. In [17], reinforcement learning based offloading algorithm is proposed. In this algorithm, energy harvesting IoT devices decide their offloading rates according to battery levels. However, the target of this algorithm is not suitable to the proposed environment where MEC is harvesting energy. In [18], the authors proposed dynamic computation offloading algorithm with a special focus on computation capacity of MEC server. In this algorithm, the energy harvesting is not primarily considered in the offloading scheduling; it is only mentioned that it can extend network life time. A healthcare IoT system was proposed in energy harvesting environment by the authors of [19]. This algorithm is also not suitable for the EH MEC environments since it aims to protect user privacy. The authors of [10] proposed a Lyapunov optimization based algorithm for energy harvesting IoT devices. They assumed that the IoT device is equipped with an energy harvesting module and harvests electricity from the module. Each IoT devices schedules its computation frequency based on the energy harvesting status. All of the works in [10,17–19] assumed that IoT devices were equipped with energy harvesting modules and tried to solve the offloading scheduling problem under battery constraints. However, these algorithms are not suitable for the proposed EH MEC environment where the MEC server is equipped with energy harvesting module, i.e., our proposed scheme aims at the stable operation of MEC server in energy harvesting environment, but previous studies aimed at survivability of IoT devices. In [20], the authors proposed MEC scheduling algorithm in EH MEC environment. They modeled the MEC battery and harvested energy, and minimized the service delay and operation cost via reinforcement learning. The energy harvesting model is similar to the model of our proposed algorithm. However, they did not consider the deadline constraint in their problem formulation. In this paper, we consider the constraints of service deadline.

3. System Model

In this section, the mathematical modeling of the proposed EH MEC system is presented. To formulate the system model, this paper assumes that the time is divided into equal time slots indexed by t such that

$$t \in \mathcal{T} \text{ where } \mathcal{T} = \{1, 2, 3, \cdots, T\}. \tag{1}$$

This paper also assumes that each IoT device requests multiple computation offloading tasks to the EH MEC server. Therefore, the scheduling is performed for each task, and we define the notations for the set of received tasks at time t as $\mathcal{R}^t$, i.e.,

$$\mathcal{R}^t = \{r_1^t, r_2^t, r_3^t, \cdots, r_R^t\}, \tag{2}$$

where R is the number of tasks requested in the timeslot. $\mathcal{R}^t$ is stored in the request buffer in the scheduler.

As shown in Figure 1, the energy of the EH MEC system is supplied from the RER and is stored in the ESS for the operation of the MEC system. Part of the energy is transferred to the small scale battery and used to operate the MEC server. The amount of energy received by the battery is denoted as e^t, and it has a maximum value of $E_{\text{harv}}^{\max}$, i.e.,

$$0 \leq e^t \leq E_{\text{harv}}^{\max}, \forall t, \tag{3}$$

where the e^ts are i.i.d. in the different timeslots.

The battery level for computation offloading at the beginning of t is denoted as B^t, and it is assumed that $B^0 = 0$ and $B^t < +\infty$, i.e., B^t is consumed only for computation offloading. In addition, if we denote the energy consumed by computation offloading at t as E_{offload}^t, then the following equation holds [10]:

$$B^{t+1} = B^t - E_{\text{offload}}^t + e^t, t \in \mathcal{T}. \tag{4}$$

Based on the defined notations, the optimization problems of the computation offloading scheduling are formulated as presented in the following subsections.

3.1. Computation Offloading Model

When an IoT device requests computation offloading to the EH MEC system, the EH MEC system can either allow offloading or pass the request to the cloud server. When the EH MEC system offloads all computation tasks, the service QoE will increase and the network overhead will be reduced. However, it is impossible to offload all the computation tasks due to the constraints of the EH MEC in terms of energy or computing performance. Therefore, the scheduler of the EH MEC system controls the admission of offloading requests, i.e., it determines whether it needs to be executed by the MEC or passed to the cloud server.

Figure 2 shows the proposed EH MEC scheduler. As shown in the figure, the proposed EH MEC scheduler consists of two sub-schedulers (offloading and MEC scheduler), three buffers (request, cloud, and MEC buffers), and two managers (battery and backhaul managers). At time t, the requests from the IoT devices are stored in the request buffer, and the request buffer is modeled as $\mathcal{R}^t$. Similarly, the MEC buffer can be modeled as

$$\mathcal{M}^t = \{m_1^t, m_2^t, m_3^t, \cdots, m_M^t\}, \tag{5}$$

where M is the number of unexecuted tasks in the MEC buffer. Analogously, the cloud buffer can be modeled as

$$\mathcal{C}^t = \{c_1^t, c_2^t, c_3^t, \cdots, c_C^t\}, \tag{6}$$

where C is the number of tasks in the cloud buffer.

Based on the SoC and MEC state, the offloading scheduler performs task scheduling (i.e., admission control) and updates the MEC buffer and cloud buffer.

For offloading scheduling, this paper introduces the following mathematical models. A request for computation offloading r_i^t is defined as follows:

$$r_i^t \triangleq [L_i^t, d_i^t], \tag{7}$$

where L_i^t is the input bit size of the task and d_i^t is the execution deadline.

In this paper, we assume that d_i^t represents only the execution time in the MEC server. If we denote the number of CPU cycles required to process the bit input in the MEC server as X, then the number of CPU cycles required to execute r_i^t successfully can be obtained as

$$W_{\mathcal{R},i} = L_i^t \cdot X. \tag{8}$$

The CPU frequency of the MEC server at t and the delay for the execution of r_i^t are denoted as f^t and $D_{\mathcal{R},i}^t$, respectively. Therefore, $D_{\mathcal{R},i}^t$ can be obtained as

$$D_{\mathcal{R},i}^t = \frac{W_{\mathcal{R},i}}{f^t}. \tag{9}$$

In addition, the offloading schedule of $\mathcal{R}^t$ can be represented by

$$\mathcal{S}^t \triangleq [s_1^t, s_2^t, s_3^t, \cdots, s_R^t], \tag{10}$$

where

$$s_i^t = \begin{cases} 1, & \text{if } r_i^t \text{ is allocated to the MEC server,} \\ 0, & \text{if } r_i^t \text{ is allocated to the cloud server.} \end{cases}$$

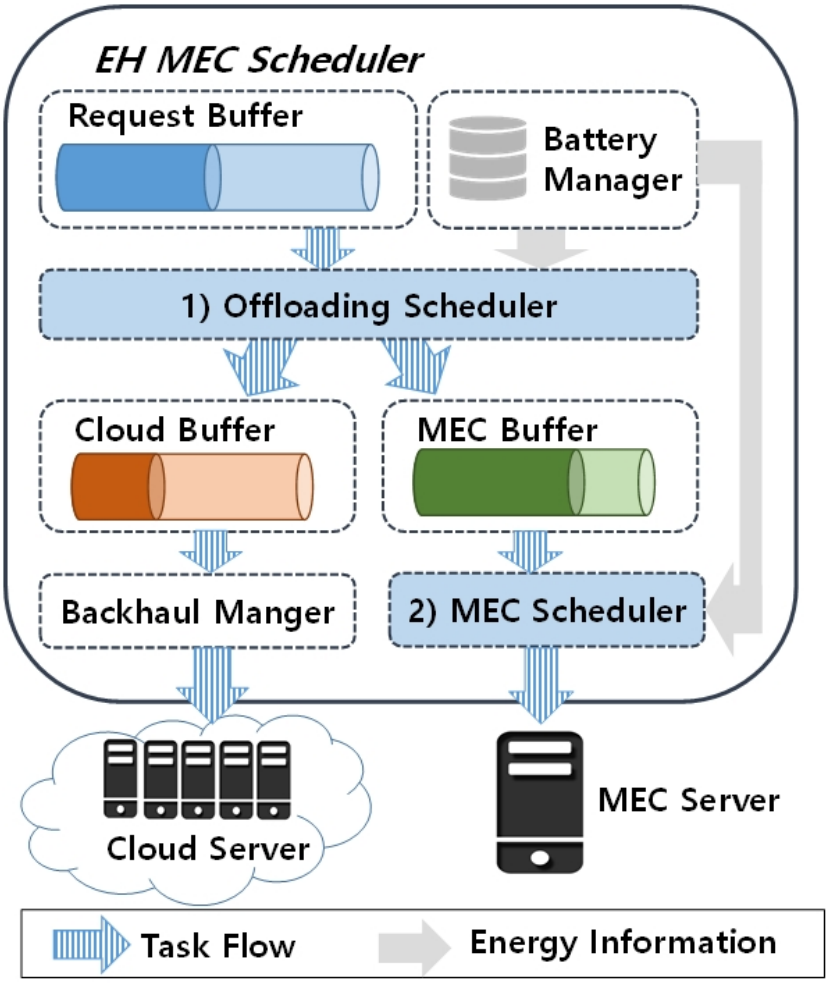

Figure 2. The proposed architecture of the EH MEC scheduler. The EH MEC scheduler includes two sub schedulers: (1) offloading scheduler; and (2) MEC scheduler.

Based on the above definitions and notations, the execution delay can be computed as follows:

$$D^t_{schedule} = \sum_{m=1}^{M} D^t_{\mathcal{M},m} + \sum_{r=1}^{R} s^t_r \cdot D^t_{\mathcal{R},r}, \tag{11}$$

where $D^t_{\mathcal{M},i}$ is the execution delay of task m^t_i in the MEC buffer and can be computed similarly to Equation (9). Therefore, the scheduling constraint for the task deadline can be derived as follows:

$$s^t_i \cdot D^t_{schedule} \leq d^t_i, \forall i. \tag{12}$$

In addition, the energy consumed for executing the task r^t_i can be obtained as follows:

$$E^t_{\mathcal{R},i} = \kappa \times W_{\mathcal{R},i} \times (f^t)^2, \tag{13}$$

where κ is the effective switched capacitance, which depends on the chip architecture. Therefore, the scheduling constraint for energy consumption can be expressed as

$$E^t_{\text{offload}} = \sum_{r=0}^{R} s^t_r \cdot E^t_{\mathcal{R},r} + \sum_{m=1}^{M} E^t_{\mathcal{M},m} \leq B^t, \tag{14}$$

where $E^t_{\mathcal{M},i}$ is the energy consumption for task m^k_i in the MEC buffer and can be computed similarly to Equation (13).

We assume that the MEC system charges the offloading bill according to the number of instruction executions, and the offloading pricing per instruction is denoted as p. Therefore, the computation offloading bill of r^t_i, denoted as $P^t_{\mathcal{R},i}$, can be obtained as

$$P_{\mathcal{R},i} = p \cdot W_{\mathcal{R},i}. \tag{15}$$

This paper also assumes that the EH MEC system wishes to maximize the bill as well as satisfy the execution delay and energy constraints. Therefore, the objective function of the offloading scheduling can be derived as follows:

$$\underset{\mathcal{S}^t}{\text{maximize}} \sum_{r=1}^{R} s^t_r \cdot P^t_{\mathcal{R},i}, \tag{16}$$

subject to

$$s^t_r \cdot D^t_{schedule} \leq d^t_r, \forall r \in \mathcal{R}, \tag{17}$$

$$\sum_{r=0}^{R} s^t_r \cdot E^t_{\mathcal{R},r} + \sum_{m=1}^{M} E^t_{\mathcal{M},m} \leq B^t. \tag{18}$$

3.2. MEC Computation Model

Based on the objective function in Equation (16) and constraints in Equations (17) and (18), the requests of the MEC buffer $\mathcal{M}^t$ can be obtained. This section describes the mathematical models used for the computation of $\mathcal{M}^t$.

To compute more bits, the MEC server increases the CPU frequency f^t according to Equation (9). However, an increase in the amount of computation results in increased energy consumption, which affects the battery life according to Equation (13).

To maximize the profit of the EH MEC system, it needs to offload more bits. However, it is impossible to maintain the maximum CPU frequency due to constraints on the battery in the EH MEC system. Therefore, the frequency scheduling of the MEC scheduler aims to maximize both the profit and battery stability.

To model the battery stability, a virtual queue Z^t is introduced. Z^t represents the amount of uncharged battery. Based on the definition of B^t, Z^t is greater than or equal to 0, i.e., $0 \leq Z^t < +\infty$. Based on Equation (4), the following equation is derived:

$$Z^{t+1} = (Z^t + E^t_{\text{offload}} - e^t)^+, \ t \in \mathcal{T}, \tag{19}$$

where

$$\alpha^+ \triangleq \max(0, \alpha). \tag{20}$$

Note that the battery should always be stable. Therefore, one constraint on the battery stability can be derived as follows:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[Z^\tau] < +\infty \tag{21}$$

The number of CPU cycles needed to execute x^t_i successfully is denoted as $W_{\mathcal{M},i}$, and it can be obtained in a similar manner as that in Equation (8). Accordingly, the required number of execution instructions at the MEC server at time t, which is denoted as I^t, can be obtained as follows:

$$I^t = \sum_{i=1}^{M} W_{\mathcal{M},i}, \tag{22}$$

where $0 \leq I^t \leq f_{\max}$.

Based on the defined notations, the number of execution cycles from the MEC buffer at time t can be obtained as

$$Q^t = \min(I^t, f^t). \tag{23}$$

A mathematical approach for maximizing the time-averaged expected number of execution cycles from the MEC buffer can be obtained as follows:

$$\text{maximize} \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Q^\tau. \tag{24}$$

Assuming that the computation frequency of the MEC server can be changed dynamically, the frequency scheduling vector can be defined as

$$\overrightarrow{f}^t = [f_1^t, f_2^t, \cdots, f_k^t], \tag{25}$$

where k is the number of frequency selections and f_i^t is an indicator function: if its frequency is selected, $f_i^t = 1$; otherwise, $f_i^t = 0$. Since the MEC should select only one frequency at t, the following constraints are derived:

$$\forall t \in \{0, \cdots, \infty\}, \tag{26}$$

$$\sum_{i=1}^{k} f_i^t = 1, \tag{27}$$

$$f_i^t \in \{0, 1\}. \tag{28}$$

Moreover, the selected frequency should guarantee the request deadline. Therefore, the minimum frequency scheduling vector required to guarantee all the request deadlines is denoted as k^{th}, and the following constraints can be derived:

$$\sum_{i=1}^{k^{th}-1} f_i^t = 0, \tag{29}$$

$$\sum_{i=k^{th}}^{k} f_i^t = 1, \tag{30}$$

and Equation (27) can be replaced by Equations (29) and (30).

To change $\overrightarrow{f}^t$ into a scalar value, this paper introduces the frequency selectable vector $\overrightarrow{F}$, which is defined as follows:

$$\overrightarrow{F} = [F_1, F_2, F_3, \cdots, F_k]. \tag{31}$$

Then, f^t can be calculated as

$$f^t = \overrightarrow{f}^t \cdot (\overrightarrow{F})^T. \tag{32}$$

Accordingly, Equations (20) and (24) are, respectively, modified as follows:

$$Z^{t+1} = Z^t + E_{\text{offload}}^t[f^t] - e^t, t \in \mathcal{T}, \tag{33}$$

$$\text{maximize} \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Q^\tau[f^\tau]. \tag{34}$$

Finally, the objective function, considering both the MEC buffer and battery stability, can be obtained as follows:

$$\text{maximize} \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Q^\tau[f^\tau], \tag{35}$$

subject to

$$\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}[Z^{\tau}] < +\infty. \tag{36}$$

4. Proposed Scheduling Algorithm

4.1. Computation Offloading Scheduling

In Section 3.1, the mathematical models and optimization problem are designed. As shown in Equation (11), $D^t_{schedule}$ is a function of s^t_r. The constraint of the task deadline in Equation (12) is non-convex since there exists a multiplication of the function $D^t_{schedule}$ by s^t_r. Therefore, the proposed optimization problem for offloading scheduling is NP-hard and cannot be solved in non-deterministic polynomial time. To solve this problem, this paper presents a novel greedy scheduling algorithm for obtaining approximate solutions.

Algorithm 1 presents the proposed greedy algorithm for offloading scheduling. The greedy algorithm consists of offloading request sorting and offloading request scheduling. First, the sequence of offloading request scheduling based on offloading request sorting is determined. The scheduling sequence is determined based on the deadline of the requests. From Equations (11) and (12), it is obvious that a computation request having a longer deadline in the MEC buffer will be satisfied if a request with a shorter deadline is satisfied. Therefore, the proposed algorithm sorts $\mathcal{R}^t$ in the descending order of d^t_i (Line 1 in Algorithm 1). Second, the algorithm alternatively schedules each request by checking the following constraints. If r^t_i is scheduled to the MEC server, the constraints of the delay and the battery should be satisfied (Line 4 in Algorithm 1). To check the delay, the expected delay can be modified by Equation (11) and expressed as follows:

$$\widehat{D^t_{schedule}} = \sum_{m=1}^{M} D^t_{\mathcal{M},m} + D^t_{\mathcal{R},i}, \tag{37}$$

Of course, the expected delay should satisfy the execution deadline as follows:

$$\widehat{D^t_{schedule}} \leq d^t_i. \tag{38}$$

To check the battery SoC, Equation (18) can be modified as follows:

$$E^t_{\mathcal{R},i} + \sum_{m=1}^{M} E^t_{\mathcal{M},m} \leq B^t, \tag{39}$$

i.e., the sum of energy consumption for the MEC tasks and the request should not be larger than the current battery capacity.

Algorithm 1 The Greedy Algorithm.

1: **Step 1:** Offloading request sorting (descending order of d^t_i)
2: **Step 2:** Offloading request scheduling
3: **for** $i = 1 \rightarrow N$ **do**
4: Determine if r^t_i satisfies both Equations (38) and (39).
5: **if** r^t_i satisfies the both Equations **then**
6: Update $\mathcal{M}^t$
7: **else**
8: Update $\mathcal{C}^t$
9: **end if**
10: **end for**

If r_i^t is suitable for allocation to the MEC, it is transferred to the MEC buffer and $\mathcal{M}^t$ is updated (Lines 5 and 6 in Algorithm 1). Otherwise, r_i^t is transferred to the cloud buffer and $\mathcal{C}^t$ is updated (Lines 7 and 8 in Algorithm 1).

4.2. MEC Scheduling

In Section 3.2, the mathematical models and the corresponding optimization problem are designed. As shown in Equations (35) and (36), the proposed objective function and the constraints considered in this paper form an NP-hard problem. Our objective function is a time domain function; therefore, the Lyapunov drift optimization technique is suitable for solving this problem since we can observe the tradeoff between the performance and battery stability. Let Θ^t denote the vector of the uncharged battery queues at time t, and the quadratic Lyapunov function is defined as

$$L^t = \frac{1}{2}(\Theta^t)^T\Theta^t = \frac{1}{2}(Z^t)^2, \tag{40}$$

where $(\Theta^t)^T$ denotes the transpose of Θ^t; however, it should be noted that Θ^t has only one queue vector, and therefore Equation (40) can be derived. Let Δ^t be a conditional quadratic Lyapunov function, which can be formulated as $\mathbb{E}[L^{t+1} - L^t \mid \Theta^t]$, i.e., the drift on t [21]. The dynamic policy is designed to solve the proposed optimization formulation by observing only the current uncharged battery queue Z^t, which is maximized as follows:

$$Q^t - V\Delta^t, \tag{41}$$

where V is a positive constant value parameter used to control the drift policy, which affects the reward–battery tradeoffs [21]. Next, we select a frequency at each time slot t. By selecting a frequency, we receive a reward. This selection can be represented as follows:

$$\underset{f^t \in F_k}{\operatorname{argmax}}\ Q^t[f^t] - V \cdot Z^t \cdot (E^t_{\text{offload}}[f^t] - e^t), \tag{42}$$

where e^t is the energy harvested at t and has a constant value. Since it does not impact the results, Equation (42) can be updated as follows:

$$\underset{f^t \in F_k}{\operatorname{argmax}}\ Q^t[f^t] - V \cdot Z^t \cdot (E^t_{\text{offload}}[f^t]). \tag{43}$$

Since Equation (43) is in the closed form, the proposed algorithm can dynamically control $\overrightarrow{f}^t$ and find the optimal $\overrightarrow{f}^t$ in polynomial time.

5. Performance Evaluation

The performance of the proposed scheduling algorithms was evaluated. Through intensive MATLAB-based simulations, the following performances were verified: (1) design of the proposed algorithms; and (2) adaptation of the EH MEC environments.

5.1. Design of the Proposed Algorithm

To verify the proposed greedy algorithm, small-scale topologies ($R = 25$) were generated in a random manner. In addition, the following parameters were considered. Each request size (L_i^t) followed a uniform distribution ([5,000, 15,000] bits), and the request deadline (d_i^t) followed a uniform distribution ($[1, 2]$ seconds). The battery size was assumed to be 100 Wh (=360,000 J), and the SoC of the battery followed a uniform distribution ($[30, 95]$%). The value of κ was set to 7.4×10^{-27}, and it was derived using an *Intel i7 Processor*. X was set to 740, i.e., the MEC server required 740 cycles to compute 1 bit. To approximately measure the greedy algorithm, the small-scale topologies were simulated

10,000 times with the above mentioned parameters. In the small-scale topologies, the optimal solution can be computed using the brute-force method since there are only a few offloading requests.

Figure 3 shows the difference in offloading bit size between the optimal solution and the greedy algorithm. As shown in the figure, the proposed algorithm guarantees more than 96% of the optimal value. Moreover, more than 99% of the difference between the greedy algorithm and the optimal solution are less than 700 bytes.

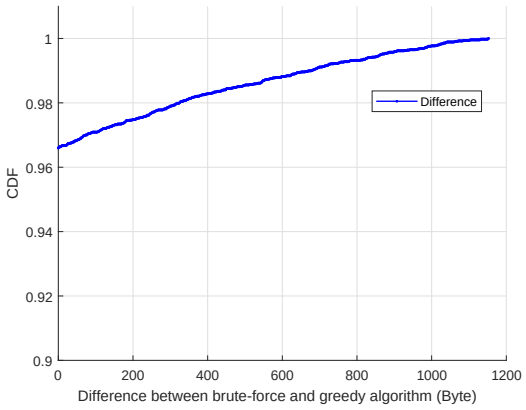

Figure 3. Difference vs. CDF.

This study also verified that the proposed MEC scheduling algorithm is well designed, i.e., the proposed Lyapunov drift optimization techniques can control the tradeoff between the performance and the battery stability according to V. For the verification of the MEC scheduling algorithms, the following simulation parameters were considered. The battery size of the MEC server was assumed to be 100 Wh (=360,000 J), and the SoC of the battery at the beginning of the simulation was assumed to be 50%.The number of IoT devices was 480, and each IoT device randomly generated the offloading request according to the Poisson distribution (the mean of the inter-arrival time was 1 s). The deadline of each request was randomly generated according to the uniform distribution using [1,2]. The number of timeslots was 86,400, and the time unit was 1 s, i.e., we simulated one day.

Figure 4 shows the changes in average battery amount and total bill according to V. As shown in the figure, the average battery amount increases with increasing V. On the other hand, the total bill decreases with increasing V. In the proposed optimization problem of MEC scheduling, there is a tradeoff between battery stability and the bill. Therefore, we can verify that the proposed Lyapunov-based MEC scheduling algorithm works well. In addition, the algorithm efficiently controls the priority between the battery stability and the bill.

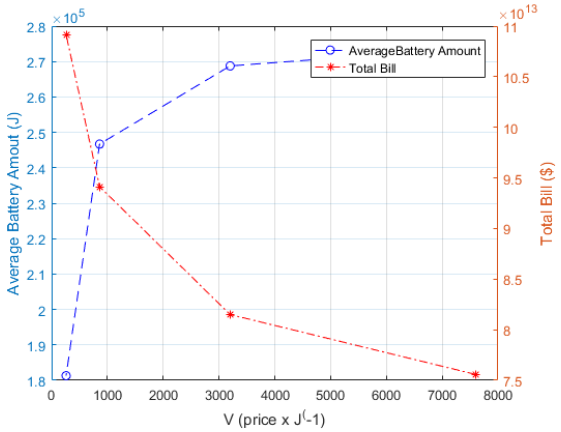

Figure 4. Average battery amount and total bill vs. V.

5.2. Adaptation of the Proposed Scheduling

This study verified the performance of the proposed scheduling algorithm in EH MEC environments. For evaluating the energy harvesting, actual measurement parameters from renewable energy resources were used. These measurement parameters were the actual field data recorded at an industrial complex that has factories making mineral waters and relevant research facilities on Jeju Island of South Korea for spring, summer, and autumn of 2017. Through the performance evaluation, we can confirm how the proposed algorithm works according to the amount of energy harvested.

Figure 5 shows the accept ratio of offloading requests according to the timeslot. In the proposed offloading scheduling algorithm, which is presented in Section 4.1, the scheduler determines the admission of requests by considering the request deadline as well as the MEC battery stability. Therefore, the proposed scheduler increases the accept ratio if the battery is sufficiently stable. As shown in the figure, the accept ratio changes according to the amount of energy harvested. At the beginning of the simulation, the requests are constantly accepted. Then, the accept ratio rapidly increases at t = 40,000 where energy harvesting begins, and then it fluctuates according to the amount of energy harvested. In addition, a higher V indicates an increasing CPU frequency; hence, the scheduler can accept more requests. Therefore, the total bill of the MEC server increases with decreasing V, as shown in Figure 4.

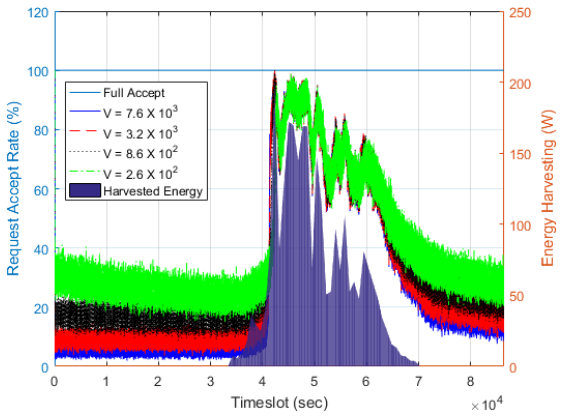

Figure 5. Request accept rate vs. timeslot.

Figure 6 shows the CPU cycles according to timeslot. As shown in the figure, the proposed scheduler efficiently controls the CPU frequency according to the amount of energy harvested. This is because the increase in energy harvesting stabilizes the battery, which results in greater battery stability. Consequently, it increases the CPU frequency to accept more requests. In addition, the figure shows the difference in number of CPU cycles with V, and a higher V relatively decreases the CPU frequency to stabilize the battery more. From t = 0 to t = 40,000, where the energy is not sufficiently harvested, the differences among different values of V are greater than the differences from t = 40,000 to t = 60,000, where the energy is sufficiently harvested. This is because, when the battery is not sufficiently stable, the offloading scheduler increases the number of rejected requests, and the MEC scheduler schedules the frequencies differently according to V. When the battery is sufficiently stable, however, the offloading scheduler increases the number of accepted requests and the MEC scheduler increases the CPU frequency to the maximum depending on the consumption of the battery, i.e., the CPU frequency is maximized depending on the boundaries that the battery can support.

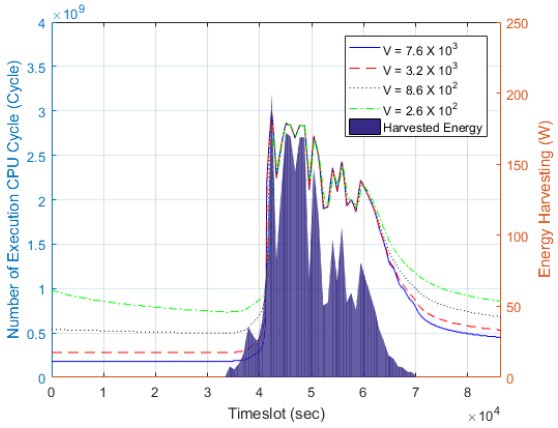

Figure 6. CPU cycles vs. timeslot.

Figure 6 shows the battery amount according to the timeslot. Similar to the previous plots, the battery amount changes according to the amount of energy harvested as shown in Figure 7. We can see that the battery amount with $V = 2.6 \times 10^2$ more rapidly compared to decreases in the battery amount with other values of V. This is because the energy consumption is quadratic with respect to the increase in CPU frequency. As shown in Figure 6, the CPU frequencies among different Vs are similar at 40,000 < t < 60,000. However, their battery amounts are different since the number of accepted requests are different.

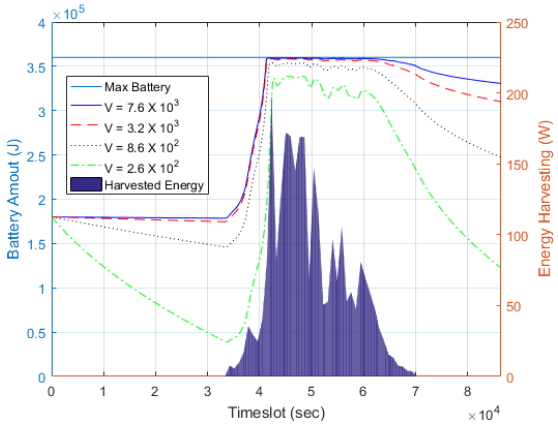

Figure 7. Battery amount vs. timeslot.

The evaluation measured the computation time of the proposed scheduling algorithm to verify its applicability. Figure 8 shows the computation time of the proposed algorithm against the number of IoT devices. As shown in the figure, the computation time is less than 0.4 ms when the number of IoT devices is 1000. In addition, the computation time increases linearly with the number of IoT devices. Thus, the complexity of the proposed scheduling algorithm is linear with the number of IoT devices. This is because the proposed scheduling algorithm is designed based on Lyapunov drift optimization, which can solve the problem in polynomial time. Therefore, the proposed algorithm can schedule the computation offloading sufficiently in real-time.

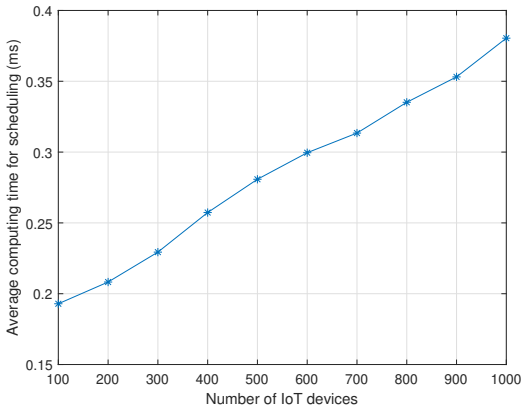

Figure 8. Computation time (ms) vs. number of IoT devices.

6. Conclusions

This paper proposes optimization formulations and their corresponding algorithms for computation offloading scheduling in an EH MEC system. For the admission control of the offloading requests, an offloading scheduling algorithm was designed to consider both the QoE of the offloading request and the system stability. The optimization problem of the offloading scheduling was formulated as a non-convex problem, and a greedy algorithm was proposed to solve the problem in polynomial time. The performance evaluation showed that the proposed greedy algorithm achieves a near-optimal performance, providing solutions that are close to the optimal solutions. To improve the survivability of the EH MEC, the proposed optimization problem of MEC scheduling considers the battery stability as well as the workload queue. Since the proposed problem of MEC scheduling is non-convex, a Lyapunov optimization-based algorithm was proposed for low-complexity scheduling. Through intensive simulations, we verified that the proposed two-stage algorithm efficiently controls the offloading requests and the CPU frequency in EH environments. However, this paper has a limitation in terms of verifying in a real-world scenario. Thus, it is essential to note that the proposed algorithm needs to be enhanced by conducting real-world testing in energy harvesting mobile edge computing. In addition, designing a scheduling algorithm by considering the offloading communication overhead is an interesting future research direction.

Author Contributions: Conceptualization, L.P.; methodology, L.P. and W.N.; software, L.P.; validation, L.P.; formal analysis, L.P.; investigation, L.P. and C.L.; resources, L.P. and S.C. (Sungyun Choi); data curation, C.L.; writing–original draft preparation, L.P.; writing–review and editing, S.C. (Sungyun Choi) and S.C. (Sungrae Cho); visualization, L.P.; supervision, S.C. (Sungrae Cho); project administration, W.N.; funding acquisition, S.C. (Sungrae Cho).

Funding: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2019R1F1A1064164).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kamilaris, A.; Pitsillides, A. Mobile Phone Computing and the Internet of Things: A Survey. *IEEE Internet Things J.* **2016**, *3*, 885–898. [CrossRef]
2. Xu, L.D.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243. [CrossRef]
3. Perera, C.; Liu, C.H.; Jayawardena, S.; Chen, M. A Survey on Internet of Things From Industrial Market Perspective. *IEEE Access* **2015**, *2*, 1660–1679. [CrossRef]
4. Deng, S.; Huang, L.; Taheri, J.; Zomaya, A.Y. Computation Offloading for Service Workflow in Mobile Cloud Computing. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 3317–3329. [CrossRef]
5. You, C.; Huang, K.; Chae, H. Energy Efficient Mobile Cloud Computing Powered by Wireless Energy Transfer. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 1757–1771. [CrossRef]

6. Guo, H.; Liu, J.; Zhang, J.; Sun, W.; Kato, N. Mobile-Edge Computation Offloading for Ultra-Dense IoT Networks. *IEEE Internet Things J.* **2018**, *5*, 4977–4988. [CrossRef]
7. Fan, Q.; Ansari, N. Application Aware Workload Allocation for Edge Computing-Based IoT. *IEEE Internet Things J.* **2018**, *5*, 2146–2153. [CrossRef]
8. Zhang, D.; Chen, Z.; Ren, J.; Zhang, N.; Awad, M.K.; Zhou, H.; Shen, X.S. Energy-Harvesting-Aided Spectrum Sensing and Data Transmission in Heterogeneous Cognitive Radio Sensor Network. *IEEE Trans. Veh. Technol.* **2017**, *66*, 831–843. [CrossRef]
9. Mao, Y.; Zhang, J.; Song, S.H.; Letaief, K.B. Stochastic Joint Radio and Computational Resource Management for Multi-User Mobile-Edge Computing Systems. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 5994–6009. [CrossRef]
10. Mao, Y.; Zhang, J.; Letaief, K.B. Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3590–3605. [CrossRef]
11. Sun, H.; Zhou, F.; Hu, R.Q. Joint Offloading and Computation Energy Efficiency Maximization in a Mobile Edge Computing System. *IEEE Trans. Veh. Technol.* **2019**, *68*, 3052–3055. [CrossRef]
12. Pan, Y.; Chen, M.; Yang, Z.; Huang, N.; Shikh-Bahaei, M. Energy-Efficient NOMA-Based Mobile Edge Computing Offloading. *IEEE Commun. Lett.* **2019**, *23*, 310–313. [CrossRef]
13. Feng, J.; Pei, Q.; Yu, F.R.; Chu, X.; Shang, B. Computation Offloading and Resource Allocation for Wireless Powered Mobile Edge Computing with Latency Constraint. *IEEE Wirel. Commun. Lett.* **2019**, *8*, 1320–1323. [CrossRef]
14. Yang, X.; Yu, X.; Huang, H.; Zhu, H. Energy Efficiency Based Joint Computation Offloading and Resource Allocation in Multi-Access MEC Systems. *IEEE Access* **2019**, *7*, 117054–117062. [CrossRef]
15. Wang, F.; Xu, J.; Wang, X.; Cui, S. Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 1784–1797. [CrossRef]
16. Ning, Z.; Dong, P.; Kong, X.; Xia, F. A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 4804–4814. [CrossRef]
17. Min, M.; Xiao, L.; Chen, Y.; Cheng, P.; Wu, D.; Zhuang, W. Learning-Based Computation Offloading for IoT Devices With Energy Harvesting. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1930–1941. [CrossRef]
18. Wei, Z.; Zhao, B.; Su, J.; Lu, X. Dynamic Edge Computation Offloading for Internet of Things With Energy Harvesting: A Learning Method. *IEEE Internet Things J.* **2019**, *6*, 4436–4447. [CrossRef]
19. Min, M.; Wan, X.; Xiao, L.; Chen, Y.; Xia, M.; Wu, D.; Dai, H. Learning-Based Privacy-Aware Offloading for Healthcare IoT With Energy Harvesting. *IEEE Internet Things J.* **2019**, *6*, 4307–4316. [CrossRef]
20. Xu, J.; Chen, L.; Ren, S. Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 361–373. [CrossRef]
21. Neely, M.J. *Stochastic Network Optimization with Application to Communication and Queueing Systems*; Morgan & Claypool: San Rafael, CA, USA, 2010.