Enabling Mobile Edge Computing for Battery-less Intermittent IoT Devices

Yu-Tai Lin, Yu-Cheng Hsiao, Chih-Yu Wang
Research Center for Information Technology Innovation, Academia Sinica, Taiwan

***Abstract*—Intermittent computing enables battery-less systems to support complex tasks such as face recognition through energy harvesting, but without an installed battery. Nevertheless, the latency may not be satisfied due to the limited computing power. Integrating mobile edge computing (MEC) with intermittent computing would be the desired solution to reduce latency and increase computation efficiency. In this work, we investigate the joint optimization problem of bandwidth allocation and the computation offloading with multiple battery-less intermittent devices in a wireless MEC network. We provide a comprehensive analysis of the expected offloading efficiency, and then propose Greedy Adaptive Balanced Allocation and Offloading (GABAO) algorithm considering the energy arrival distributions, remaining task load, and available computing/communication resources. Simulation results show that the proposed system can significantly reduce the latency in a multi-user MEC network with battery-less devices.**

I. Introduction

Battery-less systems receive great attention for deploying Internet of Things (IoTs) systems in harsh or inaccessible environments, where the battery cannot be easily installed, recharged, or replaced [1]. These devices utilize energy harvesting (EH) technology to acquire energy from the ambient environment [2] on the run. The applications are from simple tasks like data collection [3] to complex applications such as face detection [4], thanks to the development of *intermittent computing* [5].

Intermittent computing grants devices the ability to perform all operations intermittently without losing the computing states [6]. However, the low-power EH interface and unstable energy supply limit the maximum computing power of such devices, so the task computation latency is expected long. This limits its application as many IoT applications are latency-critical, such as traffic monitoring, emergency detection, vehicle automation assistance, etc.

Mobile edge computing (MEC) is a promising technology to reduce the task execution latency [7] by offloading the tasks to nearby edge servers for high-speed computation with low latency. Theoretically, battery-less devices can offload their task's data to the edge devices within the local network to accelerate the task computation and thus reduce the latency. Nevertheless, we face two challenges in integrating intermittent devices into MEC system: 1) the wireless communication protocol must resist occasional power failure while still maintaining acceptable latency, and 2) the limited transmission and computing resource should be allocated in the awareness of intermittent computing and unpredictable power failure, especially the risk of communication interruption when offloading tasks to MEC.

In light of these challenges, we investigate the joint bandwidth allocation and computation offloading optimization problem with multiple battery-less intermittent devices in the wireless MEC network. We first provide a comprehensive system model formulation and analysis. Then, we propose **G**reedy **A**daptive **B**alanced **A**llocation and **O**ffloading (GABAO) algorithm, which takes the energy arrival distributions, remaining task load, and available computing/communication resources into account. Simulation results show that the proposed system can significantly reduce the latency in a multi-user MEC network with battery-less devices. Furthermore, the proposed GABAO algorithm outperforms all other baselines in terms of the system latency in all simulations. The main contributions of this paper can be summarized as follows:

- To the best of our knowledge, this paper is the first work to propose the integration of MEC with intermittent systems to reduce the task latency and study the corresponding joint resource allocation problem.
- We propose practical estimation and optimization methods for both local computation and offloading for the proposed system. For estimating the offloading efficiency, specifically, we derive the expected data amount that devices can offload to the MEC server according to the energy arriving model.
- We propose an approximate algorithm to find the bandwidth allocation solution to minimize the system latency. The proposed GABAO algorithm can achieve $(1 - 1/e)$-optimal in terms of average latency.

II. Related Works

Much research on integrating the EH to the battery-powered devices has been done [8], [9]. For the integration of MEC with such devices, a rich set of literature studies the joint allocation of the radio and computation resources [9]. Nevertheless, their solution assumes that EH-IoTs are equipped with batteries, which might not be practical for battery-less devices.

Attentions have been paid to design a reliable operating system for battery-less devices [10] or explore potential applications such as deep neural network inference [11]. Current research on battery-less devices in the MEC network focused on the system with the undemanding offloading requirement [3], [4]. Ma *et al.* [3] design a solar-based gesture recognition system for solar-power IoTs with a machine learning (ML) model

This work was supported by the Ministry of Science and Technology under Grant MOST 108-2628-E-001-003-MY3 and the Academia Sinica under Thematic Research Grant AS-TP-110-M07-2.

implemented at the MEC server. Moreover, Giordano *et al.* [4] present a battery-less smart camera. They focus on efficiently process data locally and develop the tiny ML algorithm on board. Then, the information of the face recognized is sent via the LoRaWAN to edge devices. Nevertheless, none of these works considers the dynamic tasks offloading from the devices to the MEC server.

III. System Model

We consider a MEC server along with a set of battery-less IoT devices $\mathcal{K} \triangleq \{1, 2, \ldots, K\}$. The battery-less devices harvest ambient energy and operate intermittently. We consider a slotted time system with time slot length and time index set as τ and $\mathcal{T} \triangleq \{1, \ldots, T\}$ respectively. Also, we denote t_j as the j-th time slot. Each battery-less device is equipped with capacitors. The capacitors can hold up their energy at most one slot due to leakage, and thus the device can only utilize the energy harvested from the previous time slot.

For the task computation, we consider a data-partition task model that the task-input data can be separated arbitrarily into several groups, but should follow the sequential dependency [12]. We define L_u (in bits) as the total size of the task, L_u^{re} as the remaining bits to finish, X_u (in cycles/bits) as the workload of task processing, and $f_{u,j}$ as the CPU-cycle frequency for device u in t_j, respectively.

When the task is computed at local, we define local energy consumption for device u in j-th time slot as:

$$E_{u,j} = \kappa f_{u,j}^3 \tau + \rho\tau, \ \forall u \in \mathcal{K}, \tag{1}$$

where the first term indicates the power consumption of a device computing task [13] and the second term indicates the power consumption for maintaining the operating system.

Moreover, most modern devices can adjust CPU frequency for energy efficiency but cannot exceed the maximum frequency f_u^{max}. Thus, the optimal CPU frequency for device u in t_j can be obtained as:

$$f_{u,j}^* = \min(f_u^{max}, \sqrt[3]{(E_{u,j-1}^h - E_u^b - \rho\tau)/(\kappa\tau)}). \tag{2}$$

Therefore, the maximum amount of data (in bits) computed by device u in each time slot is $\frac{\tau f_{u,j}^*}{X_u}$.

On the other hand, when the task is offloaded to the MEC server, we assume that the MEC server is equipped with $|\mathcal{K}|$-core high-speed CPU to compute $|\mathcal{K}|$ devices' tasks concurrently and provides $|\mathcal{K}|$ independent queues to store offloading data of each device [14]. We denote the remaining amount of data in the queue u at the beginning of t_j as $q_{u,j}$ (i.e. $q_{u,1} = 0$) and the CPU frequency as f_{MEC}. Then, the maximum amount of data for the MEC server to computing the task from device u in each time slot is $\frac{\tau f_{\text{MEC}}}{X_u}$.

We assume that the amount of ambient energy arrival in each time slot follows a Poisson distribution [15]. Also, we denote the unit of harvested energy as E_0 J, and the device u harvest $E_{u,j}^h$ J in t_j. Based on the autonomous I/O for intermittent systems [10], we define the backup energy threshold as E_u^b and the restore energy threshold as E_u^r. We assume that the power consumption of any device u in wireless communication is greater than local computing E_u^{loc}, and maintaining wireless connection E_u^{co} is less than data offloading E_u^{of}. That is, $E_u^{of} > E_u^{co} > E_u^{loc}$.

According to the required energy for local computing, we consider two energy states for device compute locally: *sleep* ($E_{u,j-1}^h < E_u^b + \rho\tau$) and *computing* ($E_{u,j-1}^h \geq E_u^b + \rho\tau$), and their probabilities are p_u^{loc} and $(1 - p_u^{loc})$. Besides, we consider three energy states for device to conduct wireless communication: *power-failure* ($E_{u,j-1}^h < E_u^{co} + E_u^b + \rho\tau$), *connection* ($E_u^{co} \leq E_{u,j-1}^h - E_u^b - \rho\tau < E_u^{of}$) and *offloading* ($E_{u,j-1}^h \geq E_u^{of} + E_u^b + \rho\tau$). Likewise, the probabilities of three states are p_u^{co}, $(p_u^{of} - p_u^{co})$ and $(1 - p_u^{of})$.

We denote N_u as the total number of time slots for a device to establish a connection to the MEC server and denote N_u^r as the remaining number of time slots for device u to complete the connection. N_u^r is updated in each time slot and equal to 0 if the connection is established. When its energy state is in *power-failure* at any time slot, the connection is broken and, N_u^r is reset to N_u.

We define $\nu_{u,j} \in 0, 1$ indicating whether device u decides to offload their task at t_j, where $\nu_{u,j} = 0$ means compute locally and vise versa. Also, we define $V_u = \{\nu_{u,j} | \forall j \in \mathcal{T}\}$ as the set of offloading decisions each time slot for device u and $\mathcal{V} = \{V_u | \forall u \in \mathcal{K}\}$ as the set of offloading decision of all devices. Furthermore, we define $\mathcal{BW}$ as the total system bandwidth, b_u as the bandwidth allocated to device u and $\mathcal{B} = \{b_u | \forall u \in \mathcal{K}\}$ as the set of bandwidth of all devices. Given the allocated bandwidth, the achievable transmission rate (in bit/sec) for device u to MEC server as:

$$r_u(b_u) = b_u \log_2 \left(1 + \frac{p_u^t h_u}{b_u \sigma^2}\right), \tag{3}$$

where h_u is the channel gain, p_u^t is transmission power, and σ^2 is the power of thermal noise [16]. We set $h_u = d_u^{-\theta}$ where d_u is the distance between the device u and the MEC server and θ is the path loss factor [16]. Plus, we ignore the negligible return of computing results [8].

IV. Problem Formulation

The goal of the MEC server is to minimize the overall system latency, which is the maximum task latency among devices. It will determine the offloading decisions as well as bandwidth allocation given the current system state. Generally, the latency of a task is incremented every τ second through time slots until the last bit of the task is completed. Here we define the auxiliary function $l_u(t_j)$ to calculate the amount of time device u spends on the task in the j-th time slot.

$$l_u(t_j) = \begin{cases} \tau, & L_{u,j}^{re} \neq 0, \ \nu_{u,j} = 1, \\ \min(\tau, \frac{L_{u,j}^{re} X_u}{f_{u,j}^*}), & L_{u,j}^{re} \neq 0, \ \nu_{u,j} = 0, \\ \min(\tau, \frac{q_{u,j} X_u}{f_{MEC}}), & L_{u,j}^{re} = 0, \ q_{u,j} \neq 0, \\ 0, & L_{u,j}^{re} = 0, \ q_{u,j} = 0. \end{cases} \tag{4}$$

TABLE I: The constraints for five cases of offloading events.

(C1)	$m_0 = \max(N_u^r + 1, \phi - N_u - 1) \leq m \leq \phi$
(C2)	$N_u^r + 1 \leq m \leq m_0 - 1, \quad 1 \leq w \leq \phi - N_u - m - 1$
(C3)	$1 \leq w \leq \phi - N_u - N_u^r - 1$
(C4)	$N_u^r + 1 \leq m \leq m_0 - 1, \quad 1 \leq w \leq \phi - N_u - m - 1$
(C5)	$1 \leq s \leq \min(N_u^r, \phi - N_u - 1) = m_1.$

Note that the computation of the task of device u is completed if $L_{u,j}^{re} = 0,\ q_{u,j} = 0$ so that $l_u(t_j) = 0$. The system latency minimization problem can then formulate as follows:

$$\min_{\mathcal{V},\mathcal{B}} \max_{u \in \mathcal{K}} \sum_{j=1}^{T} l_u(t_j \mid E_{u,j-1}^h, L_{u,j}^{re}, \nu_{u,j}, q_{u,j}) \tag{5a}$$

$$\text{subject to} \ \ \nu_{u,j} \in \{0,1\},\ \forall u \in \mathcal{K},\ \forall j \in \mathcal{T}, \tag{5b}$$

$$0 \leq b_u \leq \mathcal{B}\mathcal{W},\ \forall u \in \mathcal{K}, \tag{5c}$$

$$\sum_{u \in \mathcal{K}} b_u \leq \mathcal{B}\mathcal{W}. \tag{5d}$$

However, it is challenging to find a closed-form solution to this optimization problem due to its complexity and unknown energy arrival realizations in future states. Therefore, we propose the GABAO algorithm to solve this problem.

V. Proposed Solution

The key idea of the proposed solution is to offload the tasks to the MEC server when it is more efficient than local computing. Thus, the system should estimate the offloading and local computing efficiency and allocate bandwidth periodically. We define the estimation period as ϕ time slots.

A. Offloading and Local Computing Efficiency

We define offloading efficiency as the amount of data that can be offloaded to the MEC server in the estimation period. The general case of offloading is shown in Fig. 1. As shown in Fig. 1, t_{j+m} is *the last offloading* time slot in the first round, and t_{j+m+w} is *the last power failure* time slot. Varying m and w, we categorize five cases of offloading events as follows:

- (C1) : The device establishes the connection and starts offloading data to the MEC server. However, *power failure* happens, and the remaining time slots are not enough for another connection establishment.
- (C2) : Same as (C1) except the number of remaining time slots is enough to establish another connection and offloading i.e. $\phi - m - w > N_u + 1$.
- (C3) : Extended from (C2), we suppose that no energy state before power failure is *offloading* so the last offloading time slot does not exist in (C3).
- (C4) : Extended from (C2), the device successfully establishes the second connection but none of successive slots with enough energy to offload.
- (C5) : We assume *The first power failure* occurs in t_{j+s}, which makes connection establishment fail. After t_{j+s}, the device considers (C1)-(C4) in shorter estimation length $\phi - s$, respectively.

In Table I, we show the conditions to satisfy the assumption in each case. For (C1), the lower bound of m is $m_0 = \max(N_u^r + 1, \phi - N_u - 1)$. The first term ensures the device has established the connection first within t_{j+1} to $t_{j+N_u^r}$, and the second term guarantees the remaining time slots for *Second-Round* are not sufficient to establish another connection and offload. For (C2), (C3), and (C4), the upper bound of w ensures at least one offloading time slot in *Second-Round*. For (C5), the upper bound of s is $m_1 = \min(N_u^r, \phi - N_u - 1)$, where the first term means the first connection failed and the second term confirms the number of remaining time slots is sufficient to establish a connection and offload. We utilize (C1)-(C5) to reduce the calculation of estimation on the expected number of offloading time slots. We elaborate details of the calculation as below.

(C1): We assume that the connection is establishing from t_{j+1} to $t_{j+N_u^r}$ so that the energy states among these slots are either *connection* or *offloading*. The corresponding probability is $(1 - p_u^{co})^{N_u^r}$. Also, t_{j+m} is the last offloading slot and its probability is $(1 - p_u^{of})$. Therefore, the energy states from $t_{j+N_u^r+1}$ to t_{j+m-1} are either *connection* or *offloading*, and the device offloads the task to the server only when the energy state is *offloading*. Through the binomial permutation, the probability of device u having k slots offloading to the MEC server from $t_{j+N_u^r+1}$ to t_{j+m-1} can be expressed as:

$$p_{k,off} = \tbinom{m-N_u^r-1}{k}(1 - p_u^{of})^k (p_u^{of} - p_u^{co})^{m-N_u^r-1-k}. \tag{6}$$

Multiplying the probability of t_{j+m}, the probability of device u having $(k+1)$ slots offloading from $t_{j+N_u^r+1}$ to t_{j+m} is:

$$\psi(m,k) = p_{k,off}(1 - p_u^{of}). \tag{7}$$

As we assume the t_{j+m} is the last offloading slot, the device cannot offload within t_{j+m+1} to $t_{j+\phi}$. There are two possible scenarios for this outcome: 1) the *power-failure* occurs and the remaining time slots are not enough to establish another connection, 2) the connection is maintained, but none of the slots is with enough energy to offload. In the former situation, we assume that the first power failure happens at the beginning of t_{j+m+v}, the energy states of t_{j+m+1} to $t_{j+m+v-1}$ only could be $connection$. Since another connection can't be completed after the $power\text{-}failure$, the energy states of subsequent time slots after the t_{j+m+v} are irrelevant. Thus, we can form a geometric sequence with the common ratio $(p_u^{of} - p_u^{co})$ to express the probability of each possible event as follows:

$$p_{gs} = \frac{p_u^{co}(1 - (p_u^{of} - p_u^{co})^{\phi-m})}{1 - (p_u^{of} - p_u^{co})}. \tag{8}$$

In the latter situation, on the other hand, the energy states of t_{j+m+1} to $t_{j+\phi}$ only could be $connection$, and the probability of this event is $(p_u^{of} - p_u^{co})^{\phi-m}$. Thus, the summation of the above probability between t_{j+m+1} and $t_{j+\phi}$ as follows:

$$\Gamma(m,\phi) = p_{gs} + (p_u^{of} - p_u^{co})^{\phi-m}. \tag{9}$$

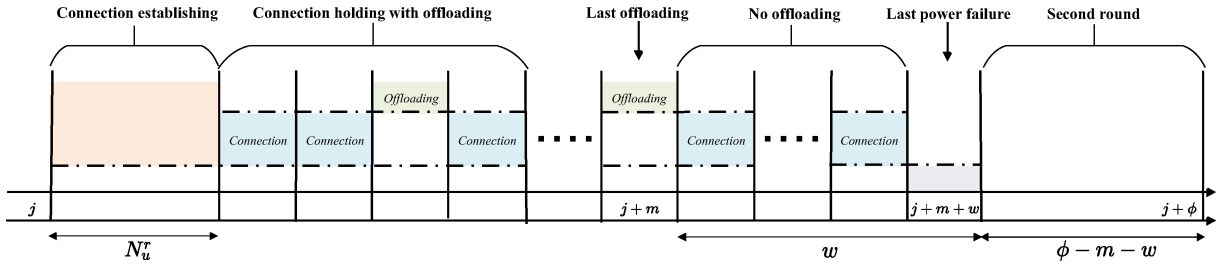


Fig. 1: The general illustration of offloading events within the $j+1$ to the $j+\phi$.

Therefore, we can obtain the probability of each possible event in (C1) by varying the value of m and k. The expected number of offloading slots in (C1) can be written as:

$$E[\ x \mid (\mathrm{C1}), \phi, N_u^r] = \tag{10}$$
$$(1-p_u^{co})^{N_u^r} \sum_{m=m_0}^{\phi} \sum_{k=0}^{m-N_u^r-1} \psi(m,k)\Gamma(m,\phi)(k+1),$$

(C2): Different from (C1), (C2) assumes that the number of remaining time slots is enough for *Second-Round*. The sum of probability within the t_{j+m+1} to the $t_{j+m+w+N_u}$ is:

$$\Theta(m,w) = \Gamma(m, m+w-1)\ p_u^{co}\ (1-p_u^{co})^{N_u}, \tag{11}$$

Next, we obtain the probability of the number of offloading time slots from $t_{j+m+w+N_u+1}$ to $t_{j+\phi}$ same as (C1) by letting $t_{j+m+w+n}$ be the last offloading time slot in *Second-Round*. The expected number of offloading time slots in (C2) is:

$$E[\ x \mid (\mathrm{C2}), \phi, N_u^r] = \tag{12}$$
$$(1-p_u^{co})^{N_u^r} \sum_{m=N_u^r+1}^{m_0-1} \sum_{k=0}^{m-N_u^r-1} \psi(m,k) \sum_{w=1}^{\phi-N_u-m-1} \Theta(m,w)$$
$$\sum_{n=N_u+1}^{\phi-m-w} \sum_{i=0}^{n-N_u-1} \psi(n,i)\Gamma(m+w+n,\phi)(k+i+2).$$

(C3): (C3) is similar to (C2) except that the device has not enough energy to offload before power failure occurs. Thus the expected number of offloading time slots for (C3) is:

$$E[\ x \mid (\mathrm{C3}), \phi, N_u^r] = (1-p_u^{co})^{N_u^r} \sum_{w=1}^{\phi-N_u-N_u^r-1} \Theta(N_u^r, w)$$
$$\sum_{n=N_u+1}^{\phi-N_u^r-w} \sum_{i=0}^{n-N_u-1} \psi(n,i)\Gamma(N_u^r+w+n,\phi)(i+1). \tag{13}$$

(C4): (C4) is similar to (C2). However, none of the time slots with enough energy to offload in the $t_{j+m+w+N_u+1}$ to the $t_{j+\phi}$. Thus, the expected number of offloading time slots is:

$$E[\ x \mid (\mathrm{C4}), \phi, N_u^r] = (1-p_u^{co})^{N_u^r} \sum_{m=N_u^r+1}^{m_0-1} \sum_{k=0}^{m-N_u^r-1} \psi(m,k)$$
$$\sum_{w=1}^{\phi-N_u-m-1} \Theta(m,w)\ \Gamma(m+w+N_u,\phi)(k+1). \tag{14}$$

(C5): We assume that a power failure occurs in the t_{j+s} and, thus, the connection is not completed. Because device u did not complete the connection, the energy states of the t_{j+1} to the t_{j+s-1} become irrelevant. Thus, the corresponding probability of the t_{j+1} to the t_{j+s} is equal to the probability of t_{j+s}, that is, p_u^{co}. Then, the device restarts another connection after the t_{j+s}. Therefore, the expected offloading time slots for (C5) is:

$$E[\ x \mid (\mathrm{C5}), \phi, N_u^r] = \sum_{s=1}^{m_1} p_u^{co} \sum_{i=(\mathrm{C1})}^{(\mathrm{C4})} E[\ x \mid i, \phi-s, N_u]. \tag{15}$$

Finally, we can approximate the total expected number of offloading slots by adding the expected number of offloading time slots from (C1) to (C5).

$$E[\ x \mid \phi, N_u^r] \triangleq \sum_{i=(\mathrm{C1})}^{(\mathrm{C5})} E[\ x \mid i, \phi, N_u^r]. \tag{16}$$

Within ϕ time slots, we assume that the allocated bandwidth for device u is fixed in each estimation period and denote it as b_u^*. The offloading efficiency can be expressed as:

$$\epsilon_u^{off} = \tau r_u(b_u^*) E[\ x \mid \phi, N_u^r]. \tag{17}$$

For local computing, the device u can compute locally in the j-th slot when the energy state is at least *computing*. Then the expected number of computing slots within ϕ slots is $\phi(1-p_u^{loc})$. Moreover, we can derive the expected CPU-cycle frequency of device u by replacing the $E_{u,j-1}^h$ in (2) to energy arrival mean λ_u. Therefore, the local computing efficiency can be estimated as:

$$\epsilon_u^{loc} = \frac{\tau\phi(1-p_u^{loc})\min(f_u^{max}, \sqrt[3]{\frac{\lambda_u - E_u^b - \rho\tau}{\kappa\tau}})}{X_u}. \tag{18}$$

B. Greedy Adaptive Bandwidth Allocation and Offloading

Based on the previous estimations, we proposed a greedy adaptive bandwidth allocation and offloading (GABAO) algorithm. Given the estimation on the offloading efficiency within the t_{j+1} to the $t_{j+\phi}$, we can approximate the time that device u will spend on offloading as:

$$\mathcal{C}(b_u^*) = \frac{L_{u,j}^{re}}{\tau r_u(b_u^*) E[x \mid \phi, N_u^r]} \tag{19}$$

The (19) will be treated as the cost function for device u.

The proposed GABAO algorithm is shown in Algorithm 1. The goal is to minimize the summation of task latency, that

Algorithm 1: Greedy Bandwidth Allocation Algorithm

input : $\mathcal{BW}$, $\mathcal{R}$, $\mathcal{S}$, N_u^r, $L_{u,j}^{re}$ $\forall u \in \mathcal{R}$
output: $\mathcal{B}$: bandwidth allocation result for devices in $\mathcal{R}$

$\mathcal{B} \leftarrow \emptyset$, $B_u^* \leftarrow \emptyset$ $\forall u \in \mathcal{R}$
for $i = 1$; $i \leq |\mathcal{S}|$ **do**
  $u^* \leftarrow argmax_{u \in \mathcal{R}}(f(b_u^* + |d_i|) - f(b_u^*))$
  $B_u^* \leftarrow B_u^* + d_i$ (assign bandwidth d_i to device u^*)
  $i = i + 1$
end
$\mathcal{B} = \{B_u^* | \forall u \in \mathcal{K}\}$
return $\mathcal{B}$

Algorithm 2: Adaptive Algorithm

input : $\mathcal{R}$, N_u^r, $L_{u,j}^{re}$ $\forall u \in \mathcal{R}$
output: $\nu_{u,j}$, b_u $\forall u \in \mathcal{R}$

Calculate $E[x|\phi, N_{c,u}^r]$ $\forall u \in \mathcal{R}$ according to (16).
Calculate the bandwidth allocation for devices in $\mathcal{R}$ by Algorithm 1.
Compare the offloading efficiency to the local computing efficiency and decide the offloading ν_u for device u.
Remove devices u with $\nu_u = 0$ from $\mathcal{R}$.
Use Algorithm 1 again for new $\mathcal{R}$.
return $\nu_{u,j}$, b_u $\forall u \in \mathcal{K}$

is, the summation of the costs $\mathcal{C}(b_u^*)$ of all devices. Thus, we define reward function $f(b_u^*) = -\sum_{u=1}^{|\mathcal{K}|} \mathcal{C}(b_u^*)$.

Moreover, we define d_w as a bandwidth unit size, and divide system bandwidth $\mathcal{BW}$ into $|\mathcal{BW}|/d_w$ units and form a set of units $\mathcal{S} = \{d_i | 1 \leq i \leq |\mathcal{BW}|/d_w\}$. Accordingly, we can assign d_i to device u and we define B_u to be the subset of $\mathcal{S}$ indicating that the set of bandwidth units assign to device u when offloading. The total bandwidth device u get $b_u = |B_u| d_w$ Hz. It can be shown that the reward function is sub-modular and monotonically increasing; thus, our proposed greedy bandwidth allocation algorithm is (1-1/e)-optimal [17] in terms of average latency. The proof is omitted here due to page limitation.

Algorithm 2 shows the procedure of the MEC server deciding offloading decision and bandwidth allocation. We first assuming every device is willing to offload and derive the bandwidth allocation by Algorithm 1 (line 2). Then, we exclude the devices which perform better if they compute locally from $\mathcal{R}$ (line 3 to line 4). Finally, we use Algorithm 1 on the restricted user set to improve the final bandwidth allocation (line 5).

We now summarize the procedure of IoT devices to conduct offloading. The IoT devices will periodically report their task completion states to the MEC server if possible. The MEC server will also periodically collect the above info from the records, and then decide the bandwidth allocation and offloading decisions with the GABAO algorithm. Note that if an IoT device failed to report its completion state, the MEC server would use the old state that the device reported previously. The IoT devices will receive the suggested offloading decision and the allocation result from the MEC server if they are in the connection state. Then, they will perform the actions if desired, that is, the device u will offload the task when 1) it's offloading decision $\nu_u = 1$, 2) the connection has been established, and 3) the amount of harvested energy allow device u to offload. For the rest, the device will compute locally if desired. The procedure will perform periodically following the period of ϕ until all tasks are completed.

VI. Simulations

We evaluate the performance of our proposed Greedy Adaptive algorithm through simulations. We consider a MEC system with one MEC server and multiple battery-less IoT devices. Each device is equipped with a solar panel to harvest energy. We let the length of time slot $\tau = 125$ ms, the number of devices $|\mathcal{K}| = 10$, the WiFi connection time for each device is $N_u = 1$ s, the allocation period $\phi = 24$ time slots, and the CPU-cycle frequency of MEC server $f_{MEC} = 10$ GHz. For any device u, the energy arrival follows a Poisson distribution as mentioned in Section III. The arrival mean λ_u is uniformly distributed within [26,32], the unit amount of energy arriving $E_0 = 2.5$ mJ, and the distance to the MEC server d_u is also uniformly distributed within [20,40] m. The maximum CPU-cycle frequency $f_{max,u} = 180$ MHz [18], the capacitor $C_u = 1$ mF, the backup voltage $V_u^b = 2.3$ V and the restore voltage $V_u^r = 2.8$ V [10], and the transmission power $p_u^t = 15$ dBm.

TABLE II: Comparison of local and offloading.

	Local	Offloading		
		Fixed	Average	GABAO
System Latency (s)	222.354 ± 0.003	13.153 ± 0.228	9.694 ± 0.109	8.6657 ± 0.107

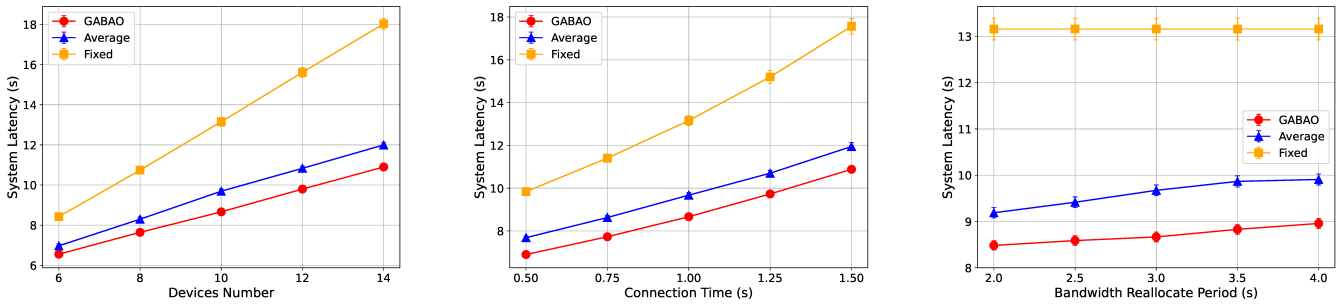
For the task, the size $L_u = 5$ MB and the processing density $X_u = 10^3$ cycles/bits [8]. For the wireless channel model, we let the total bandwidth $\mathcal{BW} = 10^7$ Hz, the power of noise $\sigma^2 = -174$ dBm/Hz, the path loss parameter $\theta = 4$, and the bandwidth unit $d_w = 10$ KHz. Based on the datasheet of ESP-12F [19], the power consumption of the wireless module on average is 71 mA, and the power consumption for Tx 802.11g with 15dBm is 140 mA. Thus, the energy consumption of connection $E_u^{co} = 29.2875$ mJ and the energy consumption of offloading $E_u^{of} = 57.75$ mJ. Besides, the coefficient in energy consumption model for local computing (1) $\kappa = 2.08 \cdot 10^{-26}$ and $\rho = 0.1159$ W [18].

We compare the performance of our proposed **GABAO** algorithm to the following algorithms.

- **Local**: All tasks are running locally at the devices themselves. This represents the legacy intermittent system where no MEC server is involved.
- **Fixed**: The bandwidth is fixed with $b_u = \frac{\mathcal{BW}}{|\mathcal{K}|}$.
- **Average**: The bandwidth is reallocated every ϕ time slots by Algorithm 2. However, instead of using Algorithm 1 for bandwidth allocation, we distribute bandwidth evenly to devices, that is, $b_u = \frac{\mathcal{BW}}{|\mathcal{R}|}$.

Table II summarizes the performance across four different algorithms. The result confirms that introducing the MEC server can greatly reduce the computation latency. Due to the great difference between local computing and offloading ones, we focus on the performance comparison among offloading algorithms in the following simulations.

In Fig. 2a, we compare 3 offloading algorithms' latency across different device numbers $|\mathcal{K}|$. We can see that our pro-



(a) Device Number (b) Connection Time (c) Bandwidth Reallocation Period

Fig. 2: Performance Evaluations

posed GABAO algorithm outperforms the Average and Fixed ones. This is because the GABAO algorithm uses the reward function which takes tasks remaining and offloading efficiency into account and tries to allocate more bandwidth to the devices which can get benefit from it. Also, Fixed algorithm performs worst since it does not reallocate bandwidth and, thus, the bottleneck devices cannot get more bandwidth to offload their tasks.

In Fig. 2b, we compare the latency with different connection times from 0.5s to 1.5s. As we can see, the system latency increase as the connection time increases in all cases. This is because the longer the connection time, the more difficult for a device to establish a connection.

In Fig. 2c, we compare the system latency with different bandwidth reallocation periods ϕ across offloading algorithms. We can see that the latency increases slightly as the allocation period increases in Average and Greedy algorithms. This is because some devices are more likely to complete their tasks at the very beginning of the period that makes the reserved bandwidth for these devices wasted. On the other hand, since Fixed algorithm does not reallocate bandwidth at all, the latency maintain the same no matter what period we choose.

VII. Conclusions

In this paper, we propose the integration of battery-less intermittent devices with the MEC system. The system latency minimization problem for the proposed system is investigated. We form a cost function including the expectation of offloading time slots and remaining task size to indicate the offloading efficiency. We propose a greedy adaptive bandwidth allocation and offloading (GABAO) algorithm, which can achieve (1-1/e)-optimal, to simplify the problem. Finally, the simulation results show that our proposed algorithms not only effectively minimize the system latency but also outperform the baseline algorithms.

References

[1] K. Takahashi, K. Kitamura, Y. Nishida, and H. Mizoguchi, "Battery-less shoe-type wearable location sensor system for monitoring people with dementia," in *Proc. of ICST*, 2019, pp. 1–4.

[2] D. Ma, G. Lan, M. Hassan, W. Hu, and S. K. Das, "Sensing, computing, and communications for energy harvesting IoTs: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1222–1250, 2020.

[3] D. Ma, G. Lan, M. Hassan, W. Hu, M. B. Upama, A. Uddin, and M. Youssef, "Solargest: Ubiquitous and battery-free gesture recognition using solar cells," in *Proc. of MobiCom*, 2019.

[4] M. Giordano, P. Mayer, and M. Magno, "A battery-free long-range wireless smart camera for face detection," in *Proc. of International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, Nov 2020, pp. 29–35.

[5] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, "Intermittent Computing: Challenges and Opportunities," in *Proc. of SNAPL*, vol. 71, 2017, pp. 8:1–8:14. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/7131

[6] N. Jackson, J. Adkins, and P. Dutta, "Capacity over capacitance for reliable energy harvesting sensors," in *Proc. of ACM/IEEE IPSN*, 2019, pp. 193–204.

[7] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, 2020.

[8] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.

[9] C. Li, W. Chen, J. Tang, and Y. Luo, "Radio and computing resource allocation with energy harvesting devices in mobile edge computing environment," *Computer Communications*, vol. 145, pp. 193 – 202, 2019.

[10] Y. Lin, P. Hsiu, and T. Kuo, "Autonomous I/O for intermittent IoT systems," in *Proc. of IEEE/ACM ISLPED*, 2019, pp. 1–6.

[11] C. K. Kang, H. R. Mendis, C. H. Lin, M. S. Chen, and P. C. Hsiu, "Everything leaves footprints: Hardware accelerated intermittent deep inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3479–3491, 2020.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K.B.Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[13] T. Burd and R. Brodersen, "Processor design for portable systems," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 13, pp. 203–221, 1996.

[14] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. of IEEE GLOBECOM*, 2016, pp. 1–6.

[15] Y. Mao, G. Yu, and C. Zhong, "Energy consumption analysis of energy harvesting systems with power grid," *IEEE Wireless Communications Letters*, vol. 2, no. 6, pp. 611–614, 2013.

[16] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[17] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167637703000622

[18] NXP Semiconductors, "Kinetis K66 Sub-Family 180 MHz ARM® Cortex®-M4F Microcontroller," https://www.pjrc.com/teensy/K66P144M180SF5V2.pdf, Jan. 2016.

[19] Ai-Thinker, "ESP-12F WIFI Module Datasheet," https://docs.ai-thinker.com/_media/esp8266/docs/esp-12f_product_specification_en.pdf, 2018.