

Article

Energy Efficient Cooperative Computation Algorithm in Energy Harvesting Internet of Things

Haneul Ko ¹, Jaewook Lee ², Seokwon Jang ², Joonwoo Kim ² and Sangheon Pack ^{2,*}

¹ Department of Computer Convergence Software, Korea University, Sejong 30019, Korea; heko@korea.ac.kr

² School of Electrical Engineering, Korea University, Seoul 02841, Korea; iioioioio123@korea.ac.kr (J.L.); imsoboy2@korea.ac.kr (S.J.); starjoon0202@korea.ac.kr (J.K.)

* Correspondence: shpack@korea.ac.kr

Received: 12 October 2019; Accepted: 22 October 2019; Published: 24 October 2019



Abstract: The limited battery capacity of Internet of Things (IoT) devices is a major deployment barrier for IoT-based computing systems. In this paper, we propose an energy efficient cooperative computation algorithm (EE-CCA). In an EE-CCA, a pair of IoT devices decide whether to offload some parts of the task to the opponent by considering their energy levels and the task deadline. To minimize the energy outage probability while completing most of tasks before their deadlines, we formulate a constraint Markov decision process (CMDP) problem and the optimal offloading strategy is obtained by linear programming (LP). Meanwhile, an optimization problem of finding pairs of IoT devices (i.e., IoT device pairing problem) is formulated under the optimal offloading strategy. Evaluation results demonstrate that the EE-CCA can reduce the energy outage probability up to 78% compared with the random offloading scheme while completing tasks before their deadlines with high probability.

Keywords: offloading; Internet of Things (IoT); energy; constraint Markov decision process (CMDP); optimization

1. Introduction

From the recent advancement of Internet of Things (IoT) devices with high computing power, complicated computation can be handled without remote servers [1]. However, the development speed of batteries for IoT devices is inferior to that of computing module, and thus the limited battery capacity of IoT devices is being a major deployment barrier for IoT-based computing systems. Therefore, there is an increasing interest on the energy harvesting technique that converts wasted energy to electricity [2,3]. With this technique, IoT devices do not need to recharge and/or replace their batteries anymore, and thus the operating expenditure of IoT-based computing systems can be reduced [4]. However, the energy that can be harvested from external energy sources is generally uncontrollable and intermittent. Moreover, the harvested energy volume has temporal and spatial variations. Therefore, it is difficult to provide a reliable power supply to IoT devices. In this situation, if an IoT device cannot harvest energy for a long time and it processes lots of tasks requiring high computing power, its energy can be depleted. To mitigate this problem and improve the energy efficiency of harvesting IoT devices, a number of works (e.g., sleep scheduling, CPU cycle adjustment, and so on) have been investigated in the literature [5–10]. One of the possible solutions is offloading tasks to nearby IoT devices [5–7]. IoT-based computing systems have advantages compared to remote servers-based offloading systems. For example, offloading to remote servers consumes huge resources in networks when IoT devices generate lots of tasks. In addition, longer latency is needed when offloading to remote servers. Especially when an energy-scarce IoT device offloads tasks to a nearby energy-abundant IoT device, energy depletion of the energy-scarce IoT devices probably does not occur.

However, unplanned offloading can cause another energy depletion. For example, when an offloader (i.e., an energy-scarce IoT device) always offloads all tasks to an offloadee (i.e., an energy-abundant IoT device) having lots of own tasks and/or small harvesting rate, the energy of the offloadee can be depleted within a short duration. Moreover, tasks cannot be completed within their deadlines due to the high load of the offloadee. Therefore, a sophisticated offloading algorithm should be devised.

In this paper, we propose an energy efficient cooperative computation algorithm (EE-CCA). In an EE-CCA, each IoT device is paired to its partner and a pair of IoT devices conduct cooperative computing. Specifically, a centralized controller collects information such as the distribution about temporal and spatial variations of external energy sources, the task occurrence rates of IoT devices, and the energy levels of IoT devices. Based on this information, the controller can construct and distribute offloading decision tables to IoT devices. Then, when a task occurs in IoT devices, they decide whether to offload some parts of the task to the opponent by following the decision tables. To minimize the energy outage probability while completing most of the tasks before their deadlines, we formulate a constraint Markov decision process (CMDP) problem, and the optimal offloading strategy is obtained by linear programming (LP). Meanwhile, an optimization problem of finding pairs of IoT devices (i.e., IoT device pairing problem) is formulated under the optimal offloading strategy. Evaluation results demonstrate that the EE-CCA can reduce the energy outage probability up to 78% compared with the random offloading scheme while completing tasks before their deadlines with high probability. In addition, the EE-CCA operates adaptively even when the operating environment (e.g., inter-task occurrence rate) changes.

The contribution of this paper can be summarized as follows: (1) we develop the cooperative computation algorithm called EE-CCA for IoT devices, while optimizing the EE-CCA by means of CMDP formulation; (2) optimal pairs of IoT devices are decided based on the optimization problem; and (3) extensive evaluation results are presented and analyzed under various environments, providing valuable guidelines for the design of cooperative computing in energy harvesting IoT.

The remainder of this paper is organized as follows. Related works are summarized in Section 2, and the EE-CCA is described in Section 3. The CMDP model for cooperative computing and the optimization problem for the IoT device pairing are developed in Section 4. Evaluation results are given in Section 5, and followed by the concluding remarks in Section 6.

2. Related Works

A number of studies on the computation offloading have been conducted to improve the energy efficiency of energy-constraint devices [11–22]. These can be categorized into: (1) framework design [11–16]; and (2) algorithm design [17–22].

Wang et al. [11] proposed an evolutionary mobile network architecture called MobiScud that integrates the cloud services into the mobile networks by means of software defined networking (SDN) and network function virtualization (NFV) technologies in a backwards compatible fashion. Tong et al. [12] organized edge cloud servers into a hierarchical architecture which enables aggregation of the peak loads across different tiers of cloud servers. Specifically, when the loads exceed the capacities of lower tiers of edge cloud servers, they can be aggregated and offloaded by other servers at higher tiers in the edge cloud hierarchy to maximize the amount of mobile workloads being served. Taleb and Ksentini [13] introduced a follow me cloud concept that enables mobile cloud services to follow their respective mobile devices by migrating services to the optimal cloud. Liu et al. [14] proposed convergence of cloud and cellular systems, abbreviated as CONCERT, based on a concept of control/data plane decoupling and hierarchically placement of the resources within the network to manage flexibly and elastically networks and cloud services. Puente et al. [15] presented a seamless approach for the deployment of edge clouds where conventional mobile traffic and computing related traffic are segregated and handled individually at base stations. However, since these works do not consider the device-to-device offloading, IoT devices with high computing power cannot be exploited efficiently. Shukla and Munir [16] proposed a computation offloading architecture where an IoT device

first tries to offload tasks to another IoT device instead of directly offloading to the cloud to process the huge amount of data while guaranteeing the task completion before the deadline. However, they did not provide any optimization method.

Ko et al. [17] proposed a spatial and temporal computation offloading decision algorithm where an energy-constraint device decides where and when to process tasks by means of a Markov decision process (MDP) by considering the energy consumption and the transmission cost. Zhao et al. [18] developed an optimization problem whose objective function is to maximize the probability that task execution satisfies the given delay bound. The problem was proved to be concave, and an optimal algorithm was proposed. Tang and Chen [19] studied a social-aware computation offloading game and designed a distributed computation offloading algorithm to achieve the Nash equilibrium. Similarly, Chen et al. [20] modeled a multi-user computation offloading game and designed a distributed computation offloading algorithm that can achieve the Nash equilibrium of the game. Zheng et al. [21] formulated the mobile users' offloading decision process under a dynamic environment as a stochastic game. Then, they proposed a multi-agent stochastic learning algorithm that can run in a fully distributed manner without any information exchange. Yu et al. [22] developed an optimal collaborative offloading strategy under a distributed caching scenario. Specifically, they formulated a problem of users' allocation as a coalition formation game with the consideration of relationships between the offloading and caching and then proposed an optimal offloading with a caching-enhancement scheme. These works improve the performance of computation offloading; however, no previous studies optimize the performance of IoT-based computing systems.

3. Energy Efficient Cooperative Computation Algorithm (EE-CCA)

Figure 1 shows the system model of this paper. In our system model, there are N IoT devices with the energy harvesting capability. We assume heterogeneous IoT environment, where IoT devices have different computing power and current energy level. In real systems, some IoT devices may not have sufficient computing power and/or harvesting capability. To support this situation, a study on the robustness of the proposed algorithm (e.g., a resilient multiscale coordination control [23]) should be conducted, which is one of our future works. In addition, since they are installed at different spots and the condition of external energy sources is volatile, energy volumes that can be harvested at each IoT device are different from each other. In addition, with different rates, these IoT devices periodically generate tasks that can be abstracted into the input data and the completion deadline before which the task should be completed [24]. In addition, we consider an application where the input data do not have dependency, e.g., binomial classification that determines whether each input is larger than a given threshold or not, and therefore the input data can be partitioned and offloaded. When the task occurs in a particular IoT device, it decides whether to offload some parts of the task (i.e., input data) to a neighbor IoT device or not with the consideration of the energy level and the deadline of the task. Note that, even though the formulation of this paper is based on the assumption where IoT devices can offload all or half of the tasks, it can be easily extended to consider other portions of the task.

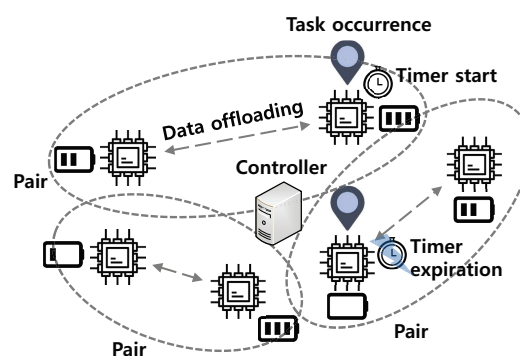


Figure 1. System model.

Intuitively, when an IoT device offloads some parts of the task to a neighbor IoT device, the task can be processed in a distributed manner, which can reduce the energy consumption of the task owner. Since IoT devices offload their tasks to nearby IoT devices by exploiting transmission technologies with low power consumption (e.g., Bluetooth), the energy consumption for transmission of the task can be neglected compared to that for processing the task. Moreover, if the neighbor IoT device does not have its own task, the task completion time can be shortened. However, if the task is offloaded to an energy-scarce IoT device, it causes the energy depletion of the IoT device, and then the offloaded task cannot be processed due to the energy depletion. In addition, when the neighbor IoT device has its own task, it should process its own task and the offloaded task simultaneously, and thus its processing time can increase. Then, both tasks may not be completed within their deadlines. To prevent these situations, we propose the EE-CCA and its flow chart is shown in Figure 2. First, the controller collects and/or maintains information such as the distribution about temporal and spatial variations of external energy sources, the task occurrence rates of IoT devices, and the energy levels of IoT devices (Step 1 in Figure 2). Based on this information, the controller determines appropriate partners of IoT devices for cooperative computation and transmits the pairing information to IoT devices (Step 2 in Figure 2). In addition, the controller constructs an offloading decision table consisting of the current status and the operation in a centralized manner. If some parameters (e.g., the task occurrence rates of IoT devices) are changed, the offloading decision table can be reconstructed by the controller and transmitted to IoT devices again. Therefore, if the parameters are frequently changed, extra signaling overhead can occur. To mitigate the signaling overhead, several techniques such as aggregation and delta encoding can be exploited [25]. Note that the offloading decision table can be obtained by CMDP, which will be elaborated in Section 4. After that, the controller transmits the optimal offloading decision table to IoT devices (Step 3 in Figure 2). On the basis of this table, IoT devices can conduct the cooperative computing (i.e., decide whether to offload some parts of the task or not) (Step 4 in Figure 2). By means of table deployments in IoT devices, the CMDP model can be applied to resource-constrained IoT devices without any high computation overhead in IoT devices [26].

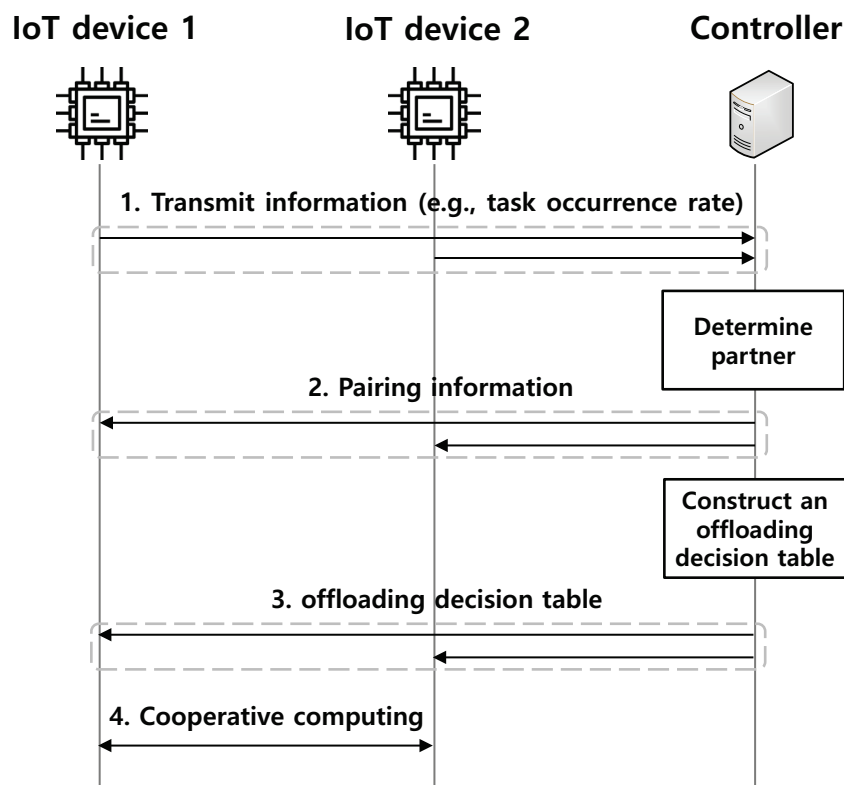


Figure 2. Flow chart.

4. Constraint Markov Decision Process (CMDP)

To obtain the optimal offloading strategy, we formulate a CMDP model with five elements. (Since the CMDP model that is a mathematical framework to model decision-making when outcomes need to be constrained, and they are partially random and under the control of the decision maker [27], it is suitable to construct the optimal offloading decision table.): (1) decision epoch; (2) state; (3) action; (4) transition probability; and (5) cost and constraint functions. Subsequently, we convert the CMDP model to an equivalent LP problem to obtain the optimal policy. After that, the IoT device pairing problem is formulated under the optimal offloading policy. Important notations for the CMDP model and IoT device pairing problem are summarized in Table 1.

Table 1. Summary of notations.

Notation	Description
S_t	State at the decision epoch t
A_t	Action chosen at the decision epoch t
τ	Duration of each decision epoch
S	Overall state space
T_i^M	State for denoting the occurrence and processing status for the task of IoT device i
T_j^M	State for denoting the occurrence and processing status for the task of IoT device j
T_i^O	State for denoting the processing status for the offloaded task of IoT device i
T_j^O	State for denoting the processing status for the offloaded task of IoT device j
E_i	State for denoting the energy level of IoT device i
E_j	State for denoting the energy level of IoT device j
D_i	State for denoting whether the timer for the deadline of the task of IoT device i expires or not
D_j	State for denoting whether the timer for the deadline of the task of IoT device j expires or not
A	Action space
A_i	Action space for IoT device i
A_j	Action space for IoT device j
E_{MAX}	Maximum battery capacity of IoT device
$r(S, A)$	Cost function on the energy outage
$c_i(S, A)$	Constraint function on the timer expiration of IoT device i
$c_j(S, A)$	Constraint function on the timer expiration of IoT device j
ζ^E	Energy outage probability
ζ_i^T	Timer expiration probability of IoT device i
ζ_j^T	Timer expiration probability of IoT device j
θ_i^T	Upper limit on the timer expiration probability of IoT device i
θ_j^T	Upper limit on the timer expiration probability of IoT device j
ζ_{ij}^E	Individual energy outage probability of IoT device i when it is paired with IoT device j
x_{ij}	Decision variable to denote whether IoT device i is paired with IoT device j or not

4.1. Decision Epoch

Figure 3 shows the timing diagram for the CMDP model. A sequence $T = \{1, 2, 3, \dots\}$ represents the time epochs when successive decisions are made [28]. S_t and A_t denote the state and the action chosen at the decision epoch $t \in T$, respectively. τ represents the duration of each decision epoch.

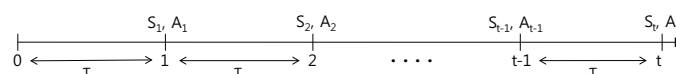


Figure 3. Timing diagram.

4.2. State Space

We define the overall state space \mathbf{S} as (The state space is constructed based on the assumption where IoT devices can offload all or half of the task. However, it can be easily extended to consider other portions of the task to add elements to \mathbf{T}_i^M , \mathbf{T}_i^O , \mathbf{T}_j^M , and \mathbf{T}_j^O .)

$$\mathbf{S} = \mathbf{T}_i^M \times \mathbf{T}_i^O \times \mathbf{E}_i \times \mathbf{D}_i \times \mathbf{T}_j^M \times \mathbf{T}_j^O \times \mathbf{E}_j \times \mathbf{D}_j, \quad (1)$$

where \mathbf{T}_i^M and \mathbf{T}_j^M are the states for representing the occurrence and processing status for the task of IoT devices i and j , respectively. \mathbf{T}_i^O and \mathbf{T}_j^O denote the states for the processing status of the offloaded task of IoT devices i and j , respectively. \mathbf{E}_i and \mathbf{E}_j are the states for the energy level of IoT devices i and j , respectively. \mathbf{D}_i and \mathbf{D}_j represent the states for denoting whether the timers for the deadline of the task of IoT devices i and j expire or not, respectively.

\mathbf{T}_i^M , \mathbf{T}_i^O , \mathbf{E}_i , and \mathbf{D}_i are the states for IoT device i , and these states are defined as follows.

First, \mathbf{T}_i^M is given by

$$\mathbf{T}_i^M = \{0, 1, 2, 3, 4\}, \quad (2)$$

where T_i^M represents the occurrence and processing status for the task of IoT device i . In other words, $T_i^M = 0$ denotes that the task does not occur in IoT device i , whereas $T_i^M = 1$ refers to the situation immediately after the task occurs in IoT device i . $T_i^M = 2$ represents the situation where IoT device i processes all of the task by itself. Meanwhile, $T_i^M = 3$ and $T_i^M = 4$ represent the situations where half of the task and all of the tasks are offloaded to IoT device j , respectively. Note that, when $T_i^M = 3$, IoT device i processes the remaining half of the task.

\mathbf{T}_i^O is represented by

$$\mathbf{T}_i^O = \{0, 1, 2\}, \quad (3)$$

where T_i^O represents the processing status of the offloaded task of IoT device i . Specifically, $T_i^O = 0$ describes the situation where any task is not offloaded to its partner (i.e., IoT device j). Meanwhile, $T_i^O = 1$ and $T_i^O = 2$ represent the situation where half of the task and all of the tasks are offloaded to IoT device j , respectively, and it is being processed in IoT device j .

\mathbf{E}_i is represented as [29]

$$\mathbf{E}_i = \{0, 1, 2, \dots, E_{MAX}\}, \quad (4)$$

where E_{MAX} is the maximum battery capacity of an IoT device.

$$\mathbf{D}_i = \{0, 1\}, \quad (5)$$

where D_i denotes whether the timer for the deadline of the task of IoT device i expires or not. In other words, $D_i = 0$ and $D_i = 1$ represent that the timer for the deadline of the task of IoT device i does not expire and expires, respectively.

\mathbf{T}_j^M , \mathbf{T}_j^O , \mathbf{E}_j , and \mathbf{D}_j are the states for IoT device j , and these states can be defined as similar with the states for IoT device i . These definitions are omitted in this paper due to the page limitation and for simple descriptions, which can be found in [30].

4.3. Action Space

When the task occurs in IoT devices, each IoT device can decide whether to offload to its partner or not and the portion to be offloaded based on the current state information. The action set is constructed based on the assumption where IoT devices can offload all or half of the task. However, it can be easily

extended to consider other portions of the task to define additional actions. Therefore, the action set can be described by

$$\mathbf{A} = \mathbf{A}_i \times \mathbf{A}_j, \quad (6)$$

where \mathbf{A}_i and \mathbf{A}_j are the action spaces for IoT devices i and j , respectively, which can be defined as

$$\mathbf{A}_i = \{0, 1, 2\} \quad (7)$$

and

$$\mathbf{A}_j = \{0, 1, 2\}, \quad (8)$$

where $A_i = 0$ and $A_j = 0$ represent that IoT devices i and j do not offload its task, respectively. $A_i = 1$ and $A_j = 1$ denote that IoT devices i and j offload half of the task to its partner, respectively. In addition, $A_i = 2$ and $A_j = 2$ are the actions where IoT devices i and j offload all of the task to its partner, respectively.

4.4. Transition Probability

The state transition probability of IoT device i is affected by the state of IoT device j . Specifically, the processing speed of the task occurring in IoT device i (i.e., the transition probability of T_i^M) is dependent on whether the task occurring at IoT device j is processed in IoT device i or not (i.e., T_j^O). In addition, the transition probability of T_i^O is affected by whether IoT device j processes its own task or not (i.e., T_j^M). Similar to that, the state transition probability of IoT device j is also influenced by the state of IoT device i (especially T_i^M and T_i^O). Therefore, the transition probability with the chosen action A from the current state S to the next state S' can be described by

$$P[S'|S, A] = P[S'_i|S_i, T_j^M, T_j^O, A] \times P[S'_j|S_j, T_i^M, T_i^O, A], \quad (9)$$

where S'_i and S'_j denote the next state of IoT devices i and j , respectively. In addition, S_i and S_j represent the current state for IoT devices i and j , respectively.

Meanwhile, T_i^M and T_i^O are influenced by the chosen action A , and these states are dependently changed with each other. In addition, T_i^M is affected by T_j^O . For example, when the task of IoT device j is processed in IoT device i , the processing speed of the task of IoT device i can decrease. Similarly, T_i^O is influenced by T_j^M . For example, when IoT device j does not process its own task, it can focus on processing the offloaded task from IoT device i , and therefore the offloaded task can be completed within a short duration. Meanwhile, when the task is processed in IoT device i , its energy level can decrease. That is, the transition of E_i is influenced by T_i^M . The timer for the deadline of the task operates only when the task occurs, and therefore the transition of D_i is affected by T_i^M and T_i^O . Meanwhile, other states change independently of each other. Therefore, for the chosen action A , the transition probability from the current state of IoT device i , $S_i = [T_i^M, T_i^O, E_i, D_i]$, to the next state of IoT device i , $S'_i = [T_i^M, T_i^O, E_i, D_i]$, can be described by

$$P[S'_i|S_i, T_j^M, T_j^O, A] = P[T_i^M|T_i^M, T_i^O, T_j^O, A] \times P[T_i^O|T_i^O, T_i^M, T_j^M, A] \times P[E'_i|E_i, T_i^M] \times P[D'_i|D_i, T_i^M, T_i^O]. \quad (10)$$

We assume that the inter-task occurrence time of IoT device i follows an exponential distribution with mean $1/\lambda_i$. Then, the probability that the task occurs in IoT device i during a decision epoch can be calculated as $\lambda_i\tau$ [27,31]. Therefore, $P[T_i^M|T_i^M = 0, T_i^O = 0, A]$ can be represented by

$$P[T_i^M|T_i^M = 0, T_i^O = 0, T_j^O, A] = \begin{cases} 1 - \lambda_i\tau, & \text{if } T_i^M = 0, \\ \lambda_i\tau, & \text{if } T_i^M = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

Before receiving the result of the offloaded task, IoT device i does not generate the task. Therefore, $P[T_i^M|T_i^M = 0, T_i^O \neq 0, T_j^O, A]$ can be defined as

$$P[T_i^M|T_i^M = 0, T_i^O \neq 0, T_j^O, A] = \begin{cases} 1, & \text{if } T_i^M = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

Meanwhile, when the task occurs (i.e., $T_i^M = 1$), IoT device i decides whether to offload to IoT device j or not and the offloaded portion (i.e., half of the task and all of the task). If IoT device i decides not to offload it (i.e., when $A = 0$), the task state will change to 2 representing the situation where IoT device i processes all of the task by itself (i.e., $T_i^M = 2$). On the other hand, when IoT device i decides to offload half of the task and all of the task (i.e., when $A = 1$ and $A = 2$), the next states of the occurrence and processing status for the task of IoT device i (i.e., T_i^M) become 3 and 4, respectively. Therefore, the corresponding transition probabilities can be represented as

$$P[T_i^M|T_i^M = 1, T_i^O, T_j^O, A = 0] = \begin{cases} 1, & \text{if } T_i^M = 2, \\ 0, & \text{otherwise,} \end{cases} \quad (13)$$

$$P[T_i^M|T_i^M = 1, T_i^O, T_j^O, A = 1] = \begin{cases} 1, & \text{if } T_i^M = 3, \\ 0, & \text{otherwise,} \end{cases} \quad (14)$$

and

$$P[T_i^M|T_i^M = 1, T_i^O, T_j^O, A = 2] = \begin{cases} 1, & \text{if } T_i^M = 4, \\ 0, & \text{otherwise,} \end{cases} \quad (15)$$

respectively.

We assume that the processing time of IoT device i for its own task follows an exponential distribution with mean $1/\mu_i^{F,S}$ when it processes all of its own task and any task of IoT device j is not offloaded to IoT device i (i.e., $T_i^M = 2$ and $T_j^O = 0$). In this case, the probability that the task is completed during a decision epoch is $\mu_i^{F,S}\tau$ [27,31]. Then, the probability that a task is not completed during a decision epoch is $1 - \mu_i^{F,S}\tau$. On the other hand, if some portion of the task of IoT device j is offloaded to IoT device i (i.e., $T_j^O \neq 0$), the processing speed of IoT device i for its own task decreases, and thus it is assumed that the processing time of IoT device i for its own task follows an exponential distribution with mean $1/\mu_i^{F,D}$ ($> 1/\mu_i^{F,S}$). In this case, the probability that the task is completed (or not completed) during a decision epoch is $\mu_i^{F,D}\tau$ (or $1 - \mu_i^{F,D}\tau$) [27,31]. Therefore, $P[T_i^M|T_i^M = 2, T_i^O, T_j^O = 0, A]$ and $P[T_i^M|T_i^M = 2, T_i^O, T_j^O \neq 0, A]$ can be derived as

$$P[T_i^M|T_i^M = 2, T_i^O, T_j^O = 0, A] = \begin{cases} 1 - \mu_i^{F,S}\tau, & \text{if } T_i^M = 2, \\ \mu_i^{F,S}\tau, & \text{if } T_i^M = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (16)$$

and

$$P[T_i^M | T_i^M = 2, T_i^O, T_j^O \neq 0, A] = \begin{cases} 1 - \mu_i^{F,D} \tau, & \text{if } T_i^M = 2, \\ \mu_i^{F,D} \tau, & \text{if } T_i^M = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (17)$$

Meanwhile, when offloading half of the task to IoT device j (i.e., $T_i^M = 3$), the remained task can be completed with shorter time. It is assumed that the processing time of the remained task follows an exponential distribution with mean $1/\mu_i^{H,S}$ when any task of IoT device j is not offloaded to IoT device i (i.e., $T_j^O = 0$), and then the probability that the remained task is completed during a decision epoch is $\mu_i^{H,S} \tau$ [27,31]. On the other hand, when IoT device i offloads the half of its task to IoT device j and processes the offloaded task from IoT device j (i.e., $T_i^M = 3$ and $T_j^O \neq 0$), the processing time of the remained task of IoT device i follows an exponential distribution with mean $1/\mu_i^{H,D}$. In this case, the probability that the remained task is completed during a decision epoch is $\mu_i^{H,D} \tau$ [27,31]. Thus, the corresponding transition probabilities can be represented as

$$P[T_i^M | T_i^M = 3, T_i^O, T_j^O = 0, A] = \begin{cases} 1 - \mu_i^{H,S} \tau, & \text{if } T_i^M = 3, \\ \mu_i^{H,S} \tau, & \text{if } T_i^M = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (18)$$

and

$$P[T_i^M | T_i^M = 3, T_i^O, T_j^O \neq 0, A] = \begin{cases} 1 - \mu_i^{H,D} \tau, & \text{if } T_i^M = 3, \\ \mu_i^{H,D} \tau, & \text{if } T_i^M = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

When the task does not occur, the processing status of the offloaded task of IoT device i does not change. Therefore, $P[T_i^O | T_i^O = 0, T_i^M \neq 1, A]$ can be denoted as

$$P[T_i^O | T_i^O = 0, T_i^M \neq 1, T_j^M, A] = \begin{cases} 1, & \text{if } T_i^O = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (20)$$

Meanwhile, when the task occurs (i.e., $T_i^M = 1$), the processing status of the offloaded task of IoT device i changes according to the chosen action A . Therefore, corresponding transition probabilities can be represented as

$$P[T_i^O | T_i^O = 0, T_i^M = 1, T_j^M, A = 0] = \begin{cases} 1, & \text{if } T_i^O = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (21)$$

$$P[T_i^O | T_i^O = 0, T_i^M = 1, T_j^M, A = 1] = \begin{cases} 1, & \text{if } T_i^O = 1, \\ 0, & \text{otherwise,} \end{cases} \quad (22)$$

and

$$P[T_i^O | T_i^O = 0, T_i^M = 1, T_j^M, A = 2] = \begin{cases} 1, & \text{if } T_i^O = 2, \\ 0, & \text{otherwise,} \end{cases} \quad (23)$$

respectively.

When some portion of the task is offloaded (i.e., $T_i^O = 1$ or $T_i^O = 2$), it is processed by IoT device j . Meanwhile, the processing time of the offloaded task to IoT device j depends on the portion of the

offloaded task (i.e., T_i^O) and whether IoT device j processes its own task or not (i.e., T_j^M). Specifically, when half of the task (or all of the task) of IoT device i is offloaded and IoT device j does not process its own task, we assume that the processing time of the offloaded task follows an exponential distribution with mean $1/\mu_j^{H,S}$ ($1/\mu_j^{F,S}$). On the other hand, if half of the task (or all of the task) of IoT device i is offloaded and IoT device j processes its own task, the processing time of the offloaded task follows an exponential distribution with mean $1/\mu_j^{H,D}$ ($1/\mu_j^{F,D}$). Then, the probabilities that the offloaded task is completed during a decision epoch for each case can be derived as $\mu_j^{H,S}\tau$, $\mu_j^{F,S}\tau$, $\mu_j^{H,D}\tau$, and $\mu_j^{F,D}\tau$, respectively [27,31]. Therefore, the corresponding transition probabilities can be denoted as

$$P[T_i^O | T_i^O = 1, T_i^M, T_j^M = 0, A] = \begin{cases} 1 - \mu_j^{H,S}\tau, & \text{if } T_i^O = 1, \\ \mu_j^{H,S}\tau, & \text{if } T_i^O = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (24)$$

$$P[T_i^O | T_i^O = 2, T_i^M, T_j^M = 0, A] = \begin{cases} 1 - \mu_j^{F,S}\tau, & \text{if } T_i^O = 1, \\ \mu_j^{F,S}\tau, & \text{if } T_i^O = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (25)$$

$$P[T_i^O | T_i^O = 1, T_i^M, T_j^M \neq 0, A] = \begin{cases} 1 - \mu_j^{H,D}\tau, & \text{if } T_i^O = 1, \\ \mu_j^{H,S}\tau, & \text{if } T_i^O = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (26)$$

and

$$P[T_i^O | T_i^O = 2, T_i^M, T_j^M \neq 0, A] = \begin{cases} 1 - \mu_j^{F,D}\tau, & \text{if } T_i^O = 1, \\ \mu_j^{H,S}\tau, & \text{if } T_i^O = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (27)$$

The IoT device can harvest energy only when its environments provide energy (e.g., when the wind blows above a certain speed). Therefore, the probability that IoT device i harvests one unit energy at an arbitrary decision epoch is modeled by a Bernoulli random process with the probability p_i^H [32]. Then, when the IoT device i does not process any task (i.e., $T_i^M = 0$ or $T_i^M = 1$) and its battery is not fully charged (i.e., $E_i \neq E_{MAX}$), E_i increases by one unit with the probability p_i^H . If the battery of IoT device i is full (i.e., $E_i = E_{MAX}$), it cannot harvest energy anymore. Therefore, the corresponding transition probabilities can be represented as

$$P[E_i' | E_i \neq E_{MAX}, T_i^M = 0] = \begin{cases} 1 - p_i^H, & \text{if } E_i' = E_i, \\ p_i^H, & \text{if } E_i' = E_i + 1, \\ 0, & \text{otherwise,} \end{cases} \quad (28)$$

$$P[E_i' | E_i = E_{MAX}, T_i^M = 0] = \begin{cases} 1, & \text{if } E_i' = E_i, \\ 0, & \text{otherwise,} \end{cases} \quad (29)$$

$$P[E_i' | E_i \neq E_{MAX}, T_i^M = 1] = \begin{cases} 1 - p_i^H, & \text{if } E_i' = E_i, \\ p_i^H, & \text{if } E_i' = E_i + 1, \\ 0, & \text{otherwise,} \end{cases} \quad (30)$$

and

$$P[E'_i|E_i = E_{MAX}, T_i^M = 1] = \begin{cases} 1, & \text{if } E'_i = E_i, \\ 0, & \text{otherwise.} \end{cases} \quad (31)$$

When IoT device i processes the task (i.e., $T_i^M = 2$ or $T_i^M = 3$) and it has energy (i.e., $E \neq 0$), it consumes one unit energy. On the other hand, if IoT device i does not have any energy, it cannot process for any task the sensed data, and thus no energy is consumed. In addition, its energy E_i increases by one unit with the probability p_i^H . Therefore, the corresponding transition probabilities can be expressed by

$$P[E'_i|E_i \neq 0, T_i^M = 2] = \begin{cases} 1 - p_i^H, & \text{if } E'_i = E_i - 1, \\ p_i^H, & \text{if } E'_i = E_i, \\ 0, & \text{otherwise,} \end{cases} \quad (32)$$

$$P[E'_i|E_i = 0, T_i^M = 2] = \begin{cases} 1, & \text{if } E'_i = E_i, \\ 0, & \text{otherwise,} \end{cases} \quad (33)$$

$$P[E'_i|E_i \neq 0, T_i^M = 3] = \begin{cases} 1 - p_i^H, & \text{if } E'_i = E_i - 1, \\ p_i^H, & \text{if } E'_i = E_i, \\ 0, & \text{otherwise,} \end{cases} \quad (34)$$

and

$$P[E'_i|E_i = 0, T_i^M = 3] = \begin{cases} 1, & \text{if } E'_i = E_i, \\ 0, & \text{otherwise.} \end{cases} \quad (35)$$

Meanwhile, when all of the tasks are offloaded to IoT device j (i.e., $T_i^M = 4$), IoT device i does not consume its own energy. Therefore, the corresponding transition probability can be denoted as

$$P[E'_i|E_i, T_i^M = 4] = \begin{cases} 1, & \text{if } E'_i = E_i, \\ 0, & \text{otherwise.} \end{cases} \quad (36)$$

When the task does not occur (i.e., $T_i^M = 0$ or $T_i^O = 0$), the timer for the deadline of the task does not start, and therefore it does not expire. Therefore, $P[D'_i|D_i, T_i^M = 0, T_i^O = 0]$ can be represented as

$$P[D'_i|D_i, T_i^M = 0, T_i^O = 0] = \begin{cases} 1, & \text{if } D'_i = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (37)$$

We assume that the timer for the deadline of the task of IoT device i follows an exponential distribution with mean $1/\kappa_i$ [33,34]. Then, when the task is not completed (i.e., $T_i^M \neq 0$ or $T_i^O \neq 0$), the probability that the timer expires during a decision epoch is $\kappa_i\tau$. Thus, the corresponding transition probabilities can be represented as

$$P[D'_i|D_i = 0, T_i^M \neq 0, T_i^O] = \begin{cases} 1 - \kappa_i\tau, & \text{if } D'_i = 0, \\ \kappa_i\tau, & \text{if } D'_i = 1, \\ 0, & \text{otherwise,} \end{cases} \quad (38)$$

and

$$P[D'_i|D_i = 0, T_i^M, T_i^O \neq 0] = \begin{cases} 1 - \kappa_i\tau, & \text{if } D'_i = 0, \\ \kappa_i\tau, & \text{if } D'_i = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (39)$$

Meanwhile, when the task is completed (i.e., $T_i^M = 0$ and $T_i^O = 0$), the timer is reset and does not operate, which means that there is no expiration. Therefore, $P[D'_i|D_i = 1, T_i^M = 0, T_i^O = 0]$ can be denoted as

$$P[D'_i|D_i = 1, T_i^M = 0, T_i^O = 0] = \begin{cases} 1, & \text{if } D'_i = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (40)$$

If the timer expires (i.e., $D_i = 1$) and the task is not completed (i.e., $T_i^M \neq 0$ or $T_i^O \neq 0$), the timer remains in the expired state. Therefore, the corresponding transition probabilities can be represented as

$$P[D'_i|D_i = 1, T_i^M \neq 0, T_i^O] = \begin{cases} 1, & \text{if } D'_i = 1, \\ 0, & \text{otherwise,} \end{cases} \quad (41)$$

and

$$P[D'_i|D_i = 1, T_i^M, T_i^O \neq 0] = \begin{cases} 1, & \text{if } D'_i = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (42)$$

The transition probability for the states of IoT device j can be defined as similar to that of IoT device i . These are omitted in this paper due to the page limitation and for simple descriptions, which can be found in [30].

4.5. Cost and Constraint Functions

4.5.1. Cost Function

To define the cost function, we consider the energy outage of IoT devices. The energy outage occurs when batteries of IoT devices are empty. Therefore, the cost function can be defined as

$$r(S, A) = \begin{cases} 1, & \text{if } E_i = 0, \\ 1, & \text{if } E_j = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (43)$$

4.5.2. Constraint Function

To prevent the situation where the task cannot be finished before the timer expiration, the constraint functions for the timer expiration of IoT devices i and j can be represented by

$$c_i(S, A) = \begin{cases} 1, & \text{if } D_i = 1, \\ 0, & \text{otherwise,} \end{cases} \quad (44)$$

and

$$c_j(S, A) = \begin{cases} 1, & \text{if } D_j = 1, \\ 0, & \text{otherwise,} \end{cases} \quad (45)$$

respectively.

4.6. Optimization Problem Formulation

Since $\sum_{t'}^t \mathbb{E} [r(S_{t'}, A_{t'})]$ means the number of energy outages, the average energy outage probability ζ^E can be defined as

$$\zeta^E = \limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{t'}^t \mathbb{E} [r(S_{t'}, A_{t'})], \quad (46)$$

where \lim denotes the value that a function approaches as the input approaches a specific value. In addition, \sup (i.e., supremum) means the least upper bound.

Meanwhile, the average timer expiration probabilities of IoT devices i and j , denoted as, ζ_i^T , and ζ_j^T , respectively, can be defined as

$$\zeta_i^T = \limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{t'}^t \mathbb{E} [c_i(S_{t'}, A_{t'})] \quad (47)$$

and

$$\zeta_j^T = \limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{t'}^t \mathbb{E} [c_j(S_{t'}, A_{t'})]. \quad (48)$$

Then, the optimization problem in the CMDP model can be formulated as

$$\min_{\pi} \zeta^E, \quad (49)$$

$$\text{s.t. } \zeta_i^T \leq \theta_i^T \text{ and } \zeta_j^T \leq \theta_j^T, \quad (50)$$

where θ_i^T and θ_j^T are the upper limits on the timer expiration probabilities of IoT devices i and j , respectively.

The formulated optimization problem can be transformed into an equivalent LP problem [28]. That is, when $\phi(S, A)$ represents the stationary probability of state S and action A , the solution of the LP problem $\phi^*(S, A)$ can be mapped to that of the CMDP-based optimization model. The equivalent LP model can be expressed as

$$\max_{\phi(S,A)} \sum_S \sum_A \phi(S, A) r(S, A), \quad (51)$$

$$\text{s.t. } \sum_S \sum_A \phi(S, A) c_i(S, A) \leq \theta_i^T, \quad (52)$$

$$\sum_S \sum_A \phi(S, A) c_j(S, A) \leq \theta_j^T, \quad (53)$$

$$\sum_A \phi(S', A) = \sum_S \sum_A \phi(S, A) P[S'|S, A], \quad (54)$$

$$\sum_S \sum_A \phi(S, A) = 1, \quad (55)$$

$$\phi(S, A) \geq 0. \quad (56)$$

The objective function in (51) is to minimize the energy outage probability of IoT devices. Meanwhile, the constraints in (52) and (53) are to maintain the timer expiration probabilities of IoT devices i and j below θ_i^T and θ_j^T , respectively. The constraint in (54) satisfies the Chapman–Kolmogorov equation. The constraints in (55) and (56) are for the probability properties.

The optimal policy $\pi^*(S, A)$, which is the probability of taking a particular action at a certain state, can be obtained from the solution of the above LP problem. The optimal policy can be derived from

$$\pi^*(S, A) = \frac{\phi^*(S, A)}{\sum_{A'} \phi^*(S, A')} \text{ for } S \in \mathbf{S}, \sum_{A'} \phi^*(S, A') > 0. \quad (57)$$

Note that, if $\sum_{A'} \phi^*(S, A') = 0$, which means that there is no solution to satisfy all constraints, IoT devices do not offload any task. LP problem can be solved in polynomial time [35–37]. Therefore, our proposed algorithm can be implemented to real systems without high computational power.

4.7. IoT Device Pairing Problem

The optimal offloading policy in the previous subsection is obtained given the paired IoT devices i and j . In this subsection, we formulate an IoT device pairing problem whose objective is to minimize the summation of energy outage probabilities of all IoT devices.

Let ζ_{ij}^E denote an individual energy outage probability of IoT device i when it is paired with IoT device j . ζ_{ij}^E can be calculated as (We assume that the individual energy outage probability of IoT device i when it is paired with IoT device j is lower than that when it is not paired (i.e., $\zeta_{ij}^E < E_{ii}^E$). This assumption is reasonable because paired IoT devices operate by following the optimal policy obtained by CMDP:

$$\zeta_{ij}^E = \sum_S \sum_A \pi^* \left([T_i^M, T_i^O, E_i = 0, D_i, S_j], A \right). \quad (58)$$

Then, the optimization problem for pairing IoT devices can be defined as

$$\min_{x_{ij}} \sum_i \zeta_{ij}^E x_{ij}, \quad (59)$$

$$\text{s.t. } \sum_j x_{ij} = 1, \forall i, \quad (60)$$

where x_{ij} is a decision variable that is 1 if IoT devices i and j are paired and 0 otherwise. The objective function in (59) is to minimize the summation of energy outage probabilities of all IoT devices. Meanwhile, the constraint in (60) to ensure that all IoT devices are paired with only one IoT device. This optimization problem is solved in the controller by using several algorithms (e.g., brute-force approach, LP relaxation, and branch-and-bound algorithm), and therefore there is no burden in IoT devices.

5. Evaluation Results

For performance evaluation, we compare the proposed algorithm, EE-CCA, with the following four schemes: (1) ALL where IoT devices always offload all of the tasks; (2) HALF where IoT devices always offload half of the task; (3) NON where IoT devices do not offload any task (i.e., process their tasks by themselves); and (4) RAND where IoT devices randomly offload their tasks at each decision epoch. For pair comparison, IoT devices are paired based on the solution of the optimization problem in Section IV-G. Meanwhile, the objective of this paper is to minimize the energy outage probability

while maintaining the probability that the task is completed before their deadline above a certain level. Therefore, the average energy outage probability, ζ^E , and the average probabilities that the task of IoT device i and j are completed before their deadline, η_i and η_j , are used as performance measures of the EE-CCA. Note that η_i and η_j can be calculated by $1 - \zeta_i^T$ and $1 - \zeta_j^T$, respectively.

To improve the reliability of the simulation results, we have conducted over 10,000 simulation runs with different seed values independently. The default number of IoT devices is set to 6. The other default parameter settings are summarized in Table 2, where $[a b]$ denotes a random value between a and b .

Table 2. Default parameter settings.

τ	λ	$\mu^{F,S}$	$\mu^{F,D}$	$\mu^{H,S}$	$\mu^{H,D}$	p^H	κ	θ^T
1	[0.2 0.3]	[0.3 0.4]	[0.15 0.2]	[0.6 0.8]	[0.3 0.4]	[0.4 0.9]	[0.2 0.3]	0.99

5.1. Effect of the Harvesting Probability

Figure 4 shows the effect of the harvesting probability p_j^H of IoT device j on the average energy outage probability and the average probabilities that tasks of IoT devices i and j are completed before their deadlines. As shown in Figure 4, it can be found that the EE-CCA can reduce significantly the energy outage probability of IoT devices (see Figure 4a). For example, when p_i^H is 0.1, EE-CCA can reduce the energy outage probability by 78% compared to RAND. while maintaining the probability that the task is completed before the deadline (i.e., 0.99) (see Figure 4b,c). This is because IoT devices in the EE-CCA decide whether to offload some parts of the task to the opponent with the consideration of the energy harvesting probability, the task occurrence rate, and the current energy levels of IoT devices. For example, IoT device does not offload its task to the partner when the current energy level of the partner is low and predicted to be decreased due to low harvesting probability and high task occurrence rate.

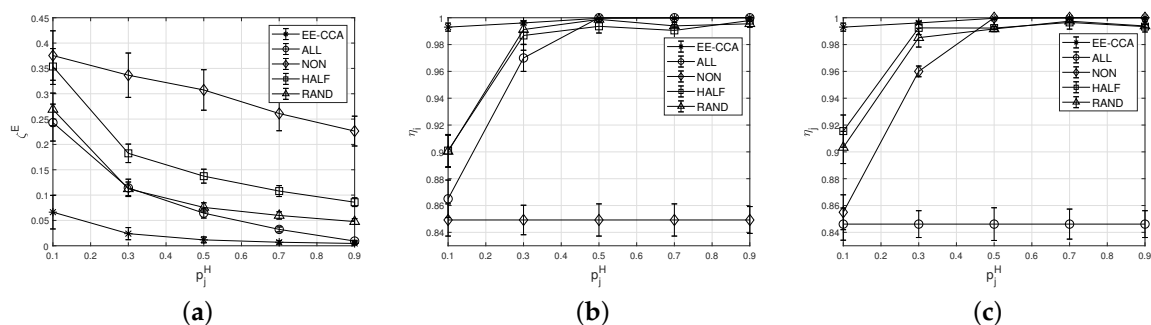


Figure 4. Effect of the harvesting probability. (a) average energy outage probability; (b) average probability that the task of IoT device i is completed before the deadline; (c) average probability that the task of IoT device j is completed before the deadline.

From Figure 4a, it can be shown that the average energy outage probabilities of all schemes decrease as p_j^H increases. This is because a high battery level of IoT device j can be maintained regardless of whether to offload or not when p_j^H is high. Meanwhile, from Figure 4b,c, it can be seen that the probabilities that tasks of IoT devices i and j are completed before their deadlines increases with the increase of p_H except specific cases (i.e., NON in Figure 4b and ALL in Figure 4c). This is because the tasks can be completed only when IoT devices have sufficient energy. In other words, IoT devices cannot complete their tasks within the deadline if they cannot harvest sufficiently energy. On the other hand, in NON, IoT device i does not offload its task to IoT device j . Therefore, the harvesting probability p_j^H of IoT device j does not affect the probability η_i that the task of IoT device i is completed

within the deadline. Similarly, in ALL, all tasks of IoT device j are processed in IoT device i , and thus η_j is not affected by p_j^H .

5.2. Effect of the Inter-Task Occurrence Rate

Figure 5 shows the effect of the inter-task occurrence rate λ_i of IoT device i . From Figure 5, it can be found that the average energy outage probabilities of all schemes increase with the increase of the inter-task occurrence rate. This is because IoT devices consume more energy when tasks occur frequently. However, the incremental ratio of the EE-CCA is smallest among comparison schemes. This is because IoT devices in the EE-CCA operate adaptively even when the operating environment changes. Specifically, as the inter-task occurrence rate of IoT device i increases, it offloads more tasks to its opponent to avoid the energy depletion.

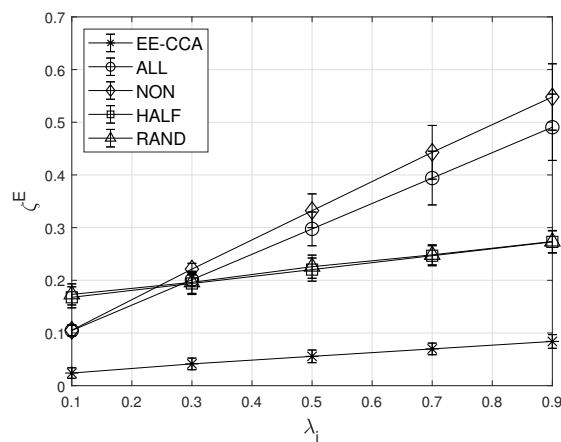


Figure 5. Effect of the inter-task occurrence rate on the average energy outage probability.

5.3. Effect of the Average Deadline

The effects of the average deadline of the task on the average energy outage probability are demonstrated in Figure 6. From Figure 6, it can be observed that the average energy outage probability of the EE-CCA decreases with the increase of the deadline. This is because, when a sufficient deadline is given, IoT devices in the EE-CCA can handle the task within the deadline by themselves even though they do not offload any tasks to energy-scarce opponents. On the other hand, the other schemes follow the fixed policy regardless of the deadline of the task, and thus their average energy outage probabilities do not change according to the deadline.

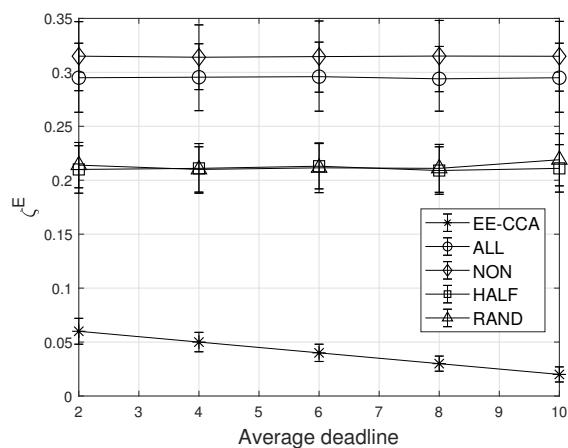


Figure 6. Effect of the average deadline of the task.

5.4. Comparison between the Optimal IoT Device Pairing and the Random Pairing

Figure 7 shows the average energy outage probabilities of the EE-CCA when pairing IoT devices based on the optimization problem (denoted by OPTIMAL) and pairing IoT devices randomly (denoted by RAND) as a function of the number of IoT devices. As shown in Figure 7, the average energy outage probability of OPTIMAL decreases as the number of IoT devices increases. This can be explained as follows: a large number of IoT devices means that there are lots of candidate IoT devices to be matched to a specific IoT device. In this situation, each IoT device can be paired to more appropriate IoT device. For example, an energy-scarce IoT device can be paired to more energy-abundant IoT device. On the other hand, since IoT devices are paired randomly in RAND regardless of the number of IoT devices, its energy outage probability is not affected by that number.

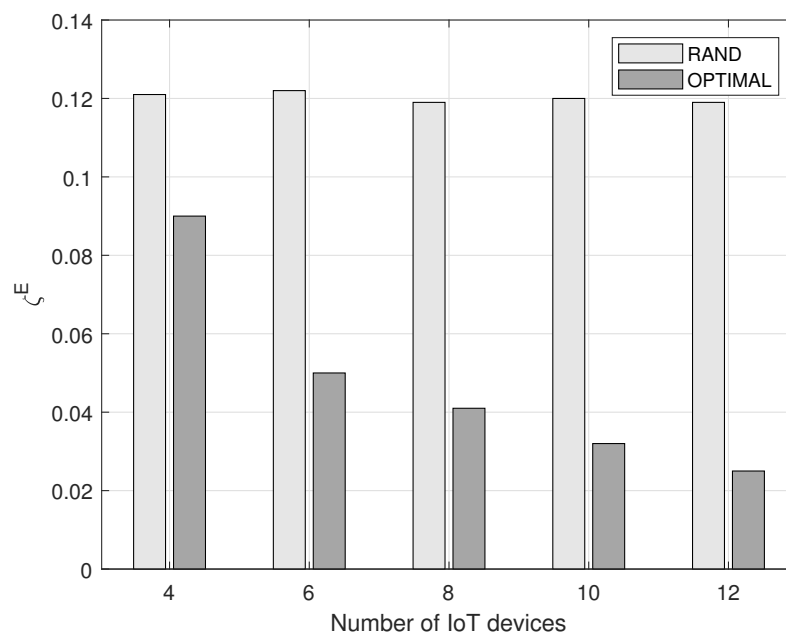


Figure 7. Comparison between the optimal pairing and random pairing.

6. Conclusions

In this paper, we proposed an energy efficient cooperative computation algorithm (EE-CCA), in which a pair of IoT devices decide whether to offload some parts of the task to the opponent with the consideration of their energy harvesting probabilities, task occurrence rates, and current energy levels. The optimal offloading decision can be obtained by means of a constraint Markov decision process (CMDP). Moreover, an optimization problem for IoT device pairing is formulated under the optimal offloading strategy. The evaluation results demonstrate that the EE-CCA offloads tasks appropriately, and thus the energy outage probability can be reduced by up to 78% compared to the random offloading scheme while providing the desired probability that tasks are completed before the deadline. Moreover, it can be seen that the EE-CCA operates adaptively even when the operating environment (e.g., inter-task occurrence rate) is changed. In our future works, we will investigate an incentive mechanism to encourage IoT devices to process tasks. In addition, a study for the robustness of the proposed algorithm will be conducted for supporting heterogeneous functionality of IoT devices.

Author Contributions: Conceptualization, H.K. and S.P.; methodology, S.J. and J.K.; software, J.L.; validation, H.K., S.J., J.K., and J.L.; investigation, S.P.; writing, original draft preparation, H.K.; review and editing, S.J., J.K., J.L., and S.P.; visualization, H.K.; supervision, S.P.

Funding: This research was supported by the National Research Foundation (NRF) of Korea Grant funded by the Korean Government (MSIP) (No. 2019R1C1C1004352).

Acknowledgments: The authors are grateful to the anonymous reviewers for their comments and valuable suggestions.

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

References

1. Arslan, M.; Singh, I.; Singh, S.; Madhyastha, H.; Sundaresan, K.; Krishnamurthy, S. Computing While Charging: Building a Distributed Computing Infrastructure Using Smartphones. In Proceedings of the 2012 International Conference on Emerging Network Experiment and Technology (CoNEXT), Nice, France, 10–12 December 2012; pp. 193–204.
2. Saleem, U.; Jangsher, S.; Qureshi, H.; Hassan, S. Joint Subcarrier and Power Allocation in Energy Harvesting-Aided D2D Communication Sign in or Purchase. *IEEE Tran. Ind. Inf.* **2018**, *14*, 2608–2617. [[CrossRef](#)]
3. Huang, L.; Qian, L.; Bi, S.; Xia, Z. Adaptive Scheduling in Energy Harvesting Sensor Networks for Green Cities. *IEEE Tran. Ind. Inf.* **2018**, *14*, 1575–1584. [[CrossRef](#)]
4. Bi, S.; Zeng, Y.; Zhang, R. Wireless Powered Communication Networks: An Overview. *IEEE Wirel. Commun.* **2016**, *23*, 10–18. [[CrossRef](#)]
5. Pu, L.; Chen, X.; Xu, J.; Fu, X. D2D Fogging: An Energy-Efficient and Incentive-Aware Task Offloading Framework via Network-Assisted D2D Collaboration. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3887–3901. [[CrossRef](#)]
6. He, Y.; Ren, J.; Yu, G.; Cai, Y. Joint Computation Offloading and Resource Allocation in D2D Enabled MEC Network. In Proceedings of the 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
7. Jin, S.; Zhu, Z.; Yang, Y.; Zhou, M.; Luo, X. Alternate Distributed Allocation of Time Reuse Patterns in Fog-enabled Cooperative D2D Networks. In Proceedings of the 2017 IEEE Fog World Congress (FWC), Santa Clara, CA, USA, 30 October–1 November 2017; pp. 1–6.
8. Das, S.; Misra, S.; Wolfinger, B.; Obaidat, M. Temporal-Correlation-Aware Dynamic Self-Management of Wireless Sensor Networks. *IEEE Trans. Ind. Inf.* **2016**, *12*, 2127–2138. [[CrossRef](#)]
9. Du, R.; Gkatzikis, L.; Fischione, C.; Xiao, M. On maximizing sensor network lifetime by energy-balancing. *IEEE Trans. Control Netw. Syst.* **2018**, *5*, 1206–1218. [[CrossRef](#)]
10. Lin, C.; Deng, D.; Shu, L.; Wang, K.; Wang, S.; Tsai, I. On lifetime enhancement of dynamic wireless sensor networks with energy-harvesting sensors. In Proceedings of the 2016 IEEE/CIC International Conference on Communications in China (ICCC), Paris, France, 27 June–1 July; pp. 1206–1218.
11. Wang, K.; Shen, M.; Cho, J.; Banerjee, A.; Merwe, J.; Webb, K. MobiScud: A Fast Moving Personal Cloud in the Mobile Network. In Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges, London, UK, 17–21 August 2015; pp. 19–24.
12. Tong, L.; Li, Y.; Gao, W. A Hierarchical Edge Cloud Architecture for Mobile Computing. In Proceedings of the 2016 IEEE International Conference on Computer Communications (ICC), San Francisco, CA, USA, 10–15 April 2016; pp. 1–9.
13. Taleb, T.; Ksentini, A. Follow Me Cloud: Interworking Federated Clouds and Distributed Mobile Networks. *IEEE Netw.* **2013**, *27*, 12–19. [[CrossRef](#)]
14. Liu, J.; Zhao, T.; Zhou, S.; Cheng, Y.; Niu, Z. CONCERT: A Cloud-Based Architecture for Next-Generation Cellular Systems. *IEEE Wirel. Commun.* **2014**, *21*, 14–22.
15. Puente, M.; Becvar, Z.; Rohlik, M.; Lobillo, F.; Strinati, E. A Seamless Integration of Computationally-Enhanced Base Stations into Mobile Networks towards 5G. In Proceedings of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, Scotland, 11–14 May 2015; pp. 1–5.
16. Shukla, R.; Munir, A. An Efficient Computation Offloading Architecture for the Internet of Things (IoT) Devices. In Proceedings of the The 14th Annual IEEE Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2017; pp. 728–731.
17. Ko, H.; Lee, J.; Pack, S. Spatial and Temporal Computation Offloading Decision Algorithm in Edge Cloud-Enabled Heterogeneous Networks. *IEEE Access* **2018**, *6*, 18920–18932. [[CrossRef](#)]

18. Zhao, T.; Zhou, S.; Guo, X.; Niu, Z. Tasks Scheduling and Resource Allocation in Heterogeneous Cloud for Delay-Bounded Mobile Edge Computing. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–7.
19. Tang, L.; Chen, X. An Efficient Social-Aware Computation Offloading Algorithm in Cloudlet System. In Proceeding of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
20. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE/ACM Trans. Netw.* **2016**, *24*, 2795–2808. [[CrossRef](#)]
21. Zheng, J.; Cai, Y.; Wu, Y.; Shen, X. Dynamic Computation Offloading for Mobile Cloud Computing: A Stochastic Game-Theoretic Approach. *IEEE Trans. Mob. Comput.* **2019**, *18*, 771–786. [[CrossRef](#)]
22. Yu, S.; Langar, R.; Fu, X.; Wang, L.; Han, Z. Computation Offloading With Data Caching Enhancement for Mobile Edge Computing. *IEEE Trans. Veh. Technol.* **2018**, *67*, 11098–11112. [[CrossRef](#)]
23. Shang, Y. Resilient Multiscale Coordination Control against Adversarial Nodes. *Energies* **2018**, *11*, 1–17. [[CrossRef](#)]
24. Zhang, W.; Wen, Y.; Guan, K.; Kilper, D.; Luo, H.; Wu, D. Energy-Optimal Mobile Cloud Computing under Stochastic Wireless Channel. *IEEE Trans. Wirel. Commun.* **2013**, *12*, 4569–4581. [[CrossRef](#)]
25. Samteladze, N.; Christensen, K. DELTA: Delta Encoding for Less Traffic for Apps. In Proceedings of the 37th Annual IEEE Conference on Local Computer Networks, Clearwater, FL, USA, 22–25 October 2012; pp. 212–215.
26. Alsheikh, N.; Hoang, D.; Niyato, D.; Tan, H.; Lin, S. Markov Decision Processes with Applications in Wireless Sensor Networks: A Survey. *IEEE Comm. Surv. Tutor.* **2015**, *17*, 1239–1267. [[CrossRef](#)]
27. Ko, H.; Pack, S. A Software-Defined Surveillance System with Energy Harvesting: Design and Performance Optimization. *IEEE Internet Things J.* **2018**, *5*, 1361–1369. [[CrossRef](#)]
28. Puterman, M. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley and Sons: Hoboken, NJ, USA, 2009.
29. Shang, Y. Vulnerability of networks: Fractional percolation on random graphs. *Phys. Rev. E* **2014**, *89*, 1–4. [[CrossRef](#)]
30. Ko, H.; Lee, J.; Jang, S.; Kim, J.; Pack, S. Energy Efficient Cooperative Computation Algorithm in Energy Harvesting Internet of Things. Available online: <https://hoy.kr/B2KEm> (accessed on 10 October 2019).
31. Guo, T.; Wang, N.; Tafazolli, R. Local Mobility Management for Networked Femtocells Based on X2 Traffic Forwarding. *IEEE Trans. Veh. Technol.* **2013**, *62*, 326–340. [[CrossRef](#)]
32. Zheng, J.; Cai, Y.; Shen, X.; Zheng, Z.; Yang, W. Green Energy Optimization in Energy Harvesting Wireless Sensor Networks. *IEEE Commun. Mag.* **2015**, *53*, 150–157. [[CrossRef](#)]
33. Fu, H.; Lin, P.; Lin, Y. Reducing Signaling Overhead for Femtocell/Macrocell Networks. *IEEE Trans. Mob. Comput.* **2013**, *12*, 1587–1597. [[CrossRef](#)]
34. Ko, H.; Lee, J.; Pack, S. Performance Optimization of Delayed WiFi Offloading in Heterogeneous Networks. *IEEE Trans. Veh. Technol.* **2017**, *66*, 9436–9447. [[CrossRef](#)]
35. Spielman, D.; Teng, S. Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time. *IEEE J. ACM* **2004**, *51*, 385–463. [[CrossRef](#)]
36. Altman, E. *Constrained Markov Decision Processes*; Chapman & Hall: London, UK, 1999.
37. Heyman, D.; Sobel, M. *Stochastic Models in Operations Research: Stochastic Optimization*; Courier Corporation: Chelmsford, MA, USA, 2003.

