Two-Stage IoT Device Scheduling with Dynamic Programming for Energy Internet Systems

Laihyuk Park, Chunghyun Lee, Joongheon Kim, Aziz Mohaisen, and Sungrae Cho

***Abstract*—With the rapid evolution of electric systems, there has been a significant demand for Energy Internet (EI) systems that allow sustainable and environmentally friendly energy management. Several research efforts regarding EI systems have been aimed at providing reliable, efficient, and cost-effective techniques. In this paper, we propose a novel algorithm and system for real-time electricity pricing and scheduling. Our algorithm consists of a two-stage operation. The first stage performs real-time pricing to determine the maximum electricity consumption while the second stage performs Internet of Things (IoT) device scheduling. In the second stage, the optimization framework for scheduling is modeled as a 0-1 Knapsack problem; therefore, the solutions to the optimization problem are computed using a dynamic programming framework. Through intensive simulations with well-defined parameters, it is verified that the proposed scheme provides several features, especially reductions in electricity bills with the appropriate parameter settings.**

***Index Terms*—Energy management problem, dynamic programming, IoT device scheduling, 0-1 Knapsack problem.**

I. Introduction

Recently, there have been considerable efforts to minimize fossil fuel usage, given that fossil fuels have been one of the major contributors to various environmental problems leading to global warming [1]–[4]. On the other hand, global electricity consumption has significantly increased over the past decade [1]–[4], making it difficult to address energy needs in the face of fluctuating yet increasing energy demands. Electric power retailers who try to respond to these demands rapidly must cope with the variations in these demands. To this end, Energy Internet (EI) systems have been envisioned as a type of future power system to address many of today's electricity problems [1]–[4]. In particular, EI systems aim to reduce the peak electricity load and induce effective electricity consumption through an *energy management problem (EMP)* in order to optimize electricity usage in an intelligent manner [5]. The definition of EMP, which is one of the fundamental challenges in a power system, is similar to the demand response in a smart grid [5]. Over the past several years, demand response researchers have focused on incorporating and implementing different components such as energy aggregator, dynamic pricing, and load shifting [6]–[9]. In general, the implementation of such components is aimed at supporting cost-effective and reliable energy management in real time. Previous studies have focused on the scheduling of IoT devices, assuming that the aggregator gets a real time pricing. In prior researches, exemplified by the works of Li *et al.* [6] and Muratori *et al.* [7], EMP algorithms were proposed for controlling peak demand and for dynamic pricing. These previous studies had considered device scheduling assuming that the aggregator receives real-time pricing results from the power retailer. However, in the proposed algorithms, all of the EI components send the resources, priorities, and other constraints to integrate the EMP systems. Nguyen *et al.* [8] presented an optimal pricing design for the distribution network using a bilevel programming approach. For forecasting the optimal pricing, they assumed that the power retailer and EMP brokers collect the load preference information for each hour in a day. In the EI environment, however, operations of the IoT devices can be initiated anytime, and it is unpredictable. Unlike the scheme presented in [8], the proposed scheme determines the real-time pricing based on real-time load information, and schedules the IoT devices. McKenna *et al.* [9] proposed the load modeling of a price-based EMP system. In [9], the device model was classified according to the consumer occupancy, activity, device electrical, thermal operation, and electrical demand. However, that study [9] did not consider the delay tolerance, which is significantly important for facilitating user convenience during IoT device scheduling. Moreover, meaningful energy harvest-assisted approaches [10]–[13] have been proposed to handle the energy-related issues in IoT. For large-scale EI systems, several game theoretic approaches have been proposed recently to reduce the computational complexity of scheduling [14]–[17]. Latifi *et al.* [14] proposed a Stackelberg game based approach, in which strategies are exchanged between the leader (power retailer) and the followers (IoT devices). In [15], a Newton method based algorithm was proposed for improving the computational efficiency of finding Nash equilibrium. In [16], a resilient distributed real-time scheduling based on population game theory was proposed. The above game theoretic approaches have tried to reduce the computation and exchanging strategies by customizing their methods. In an IE environment, however, they still require a large number of iterations to reach Nash equilibrium, which in turn result in communication overheads among IoT devices. This shortcoming is a problem since the communication overhead causes a large amount of energy consumption as well as delay. The communication overhead problem of game theoretic approach is analyzed in [17]. In order to handle the issue, a non-iterative Stackelberg model and a historical real time pricing algorithm were proposed in [17]. However, they assumed massive manufacturing systems where the operations of IoT devices are static. In

This research was supported by Korea Electric Power Corporation(Grant Number: RX17A05-43.) (Corresponding author: Sungrae Cho.)

L. Park, C. Lee, J. Kim, and S. Cho are with School of Software, Chung-Ang University, South Korea (email: srcho@cau.ac.kr). A. Mohaisen is with the Department of Computer Science, University of Central Florida, Orlando, FL 32816-2362 USA (e-mail: mohaisen@ucf.edu).

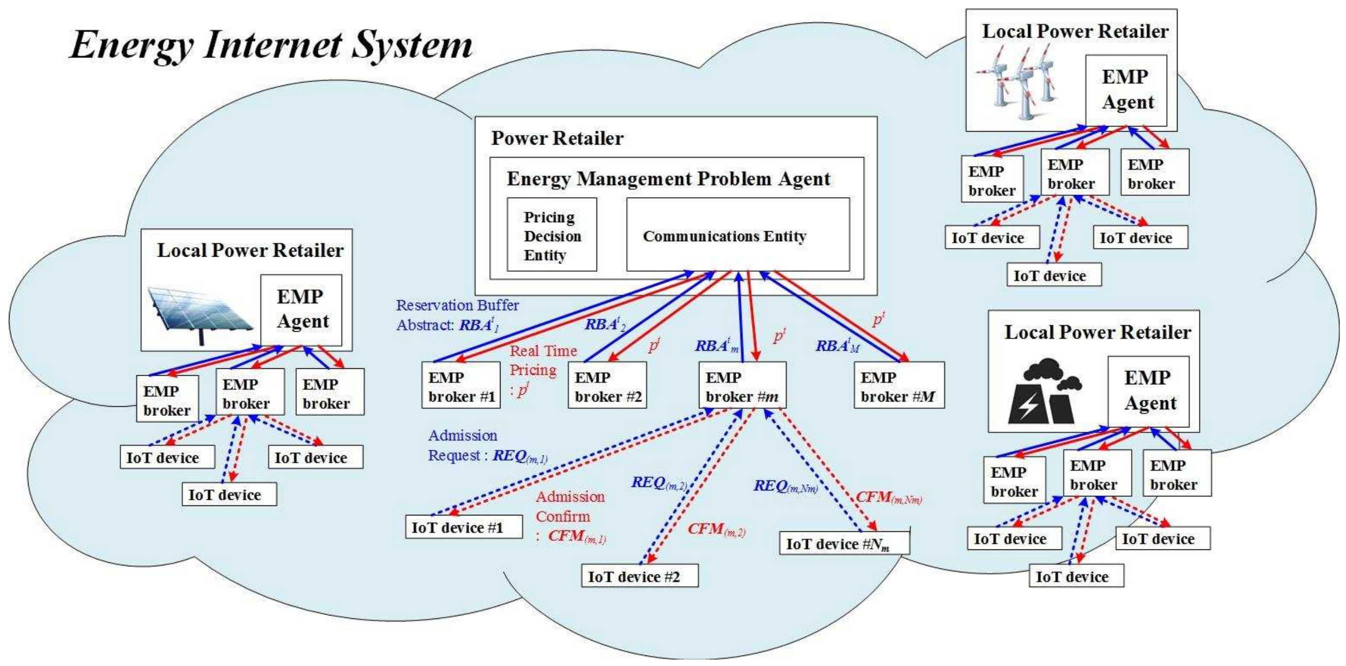

Fig. 1: The architectural view of an EI systems, consisting of an EMP agent that performs the pricing and communication decisions, an EMP broker, and IoT devices. (Since all components are connected by the Internet, EMP can manage the operation of the IoT devices by reflecting their demand.)

order to overcome the above shortcomings of prior schemes, we propose a novel architecture[1] definition for the energy management problem of EI systems, as illustrated in Fig. 1. An EI system consists of a number of EMP groups and each EMP group includes an EMP agent, EMP brokers, and IoT devices. This paper assumes that all EI components are equipped with network interfaces such as Wi-Fi, local area network (LAN), and power line communications (PLC). Through the network interface, the EI components share their strategies and information for the EMP. This networking with power retailers or other components promises intelligent EMP since more information can be gathered [18]. In order to solve the EMP, this paper introduces the concept of EMP agent and EMP broker. The EMP agent in the power retailer system efficiently supports the scheduling of IoT devices. The EMP agent is equipped with the electricity pricing policy of the power retailer, and collects electricity usage requests from each EMP broker. Based on the collected information, the real-time pricing is calculated to aid the electricity scheduling of the EMP broker. The EMP broker, which is located between the power retailer and the IoT device, manages the IoT device's operational schedule to accommodate both requests. For our system architecture, this paper proposes a two-stage approach with dynamic programming for IoT device scheduling (also called admission control for IoT devices) in the EMP of the EI system. The first stage involves taking the decision of real-time pricing. This real-time pricing is computed by a pricing decision entity in the power retailer EMP agent. If an IoT device is scheduled for operation, it sends an admission request to the EMP broker through its network interface. In each time slot, the EMP broker forwards the collected admission requests to the centralized EMP agent through the Internet. Therefore, the EMP agent can compute in real time the pricing of the next time slot based on these requests in a centralized manner.

The second stage in our approach is delay tolerant IoT device scheduling. When the EMP agent decides in real-time the pricing of the next time slot, it will inform the EMP brokers of the pricing results. Based on the obtained real-time pricing information, the EMP broker conducts IoT device scheduling (i.e., admission control) and announces the results to the IoT devices. In the second stage, the IoT device scheduling scheme is modeled as a 0-1 Knapsack problem considering the delays. According to the 0-1 Knapsack modeling, the solutions of this optimization framework can be derived via a dynamic programming-based technique in pseudo-polynomial time.

The rest of this paper is organized as follows. Section II provides the basic assumptions, system models, and corresponding optimization problems. Section III presents the proposed two-stage algorithm, which is inspired by the 0-1 Knapsack formulation and dynamic programming-based solution approaches. In Section IV, the performance of the proposed scheme is evaluated through intensive simulations of its various aspects, and the conclusions of this paper and future research directions are summarized in Section V.

[1]To resolve the communication overhead and computational efficiency in massive EI systems, the proposed architecture is hierarchical and centralized. In contrast with previous works, the proposed scheme exchanges the information once in each timeslot via hierarchical architecture. In addition, the centralized EMP broker schedules grouped IoT devices, which improves computational efficiency.

II. System Models

Before delving into the details of our approach, we introduce the preliminaries in this section. Namely, we review the

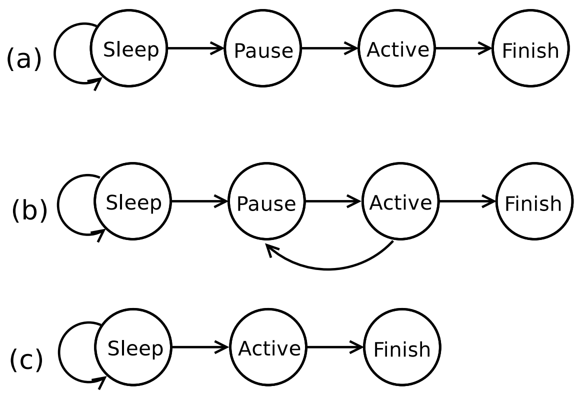

Fig. 2: State diagrams of three types of IoT devices used in our system: a) non-interruptible, (b) interruptible and (c) emergency.

basic assumptions and definitions in Subsection II-A, pricing decision models in Subsection II-B, and admission control models in Subsection II-C.

A. Basic Assumptions and Definitions

In this paper, in order to formulate the models considered for the EMP of the EI system, the time scale is assumed to be divided into equivalent time slots t such that

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

The time slot can be represented using any unit of time; for simplicity, an hour is taken as the unit time in this paper. Note that the proposed algorithms in this paper are scalable, i.e., the algorithms can work as desired with any time scale.

In this paper, the notations for the set of IoT devices that belong to the m-th EMP broker is defined as $\mathcal{A}_m$, i.e.,

$$\mathcal{A}_m = \left\{a_{(m,1)}, a_{(m,2)}, \cdots, a_{(m,n)}, \cdots, a_{(m,N_m)}\right\}, \quad (2)$$

where m indicates the index of the EMP broker and N_m represents the total number of IoT devices that belong to the m-th EMP broker.

This paper also assumes that an IoT device $a_{(m,n)}$ consumes $E^{on}_{(m,n)}$ energy per time slot t during operation.

Some IoT devices may consume electricity continuously. On the other hand, some IoT devices (especially, those equipped with a lithium battery) may consume electric power intermittently. For example, a smart robotic vacuum does not need to be continuously charged before its actual operation. Therefore, this paper categorizes the given IoT devices according to how controllable they are with respect to the energy consumption as follows:

1) **Non-interruptible IoT devices.** There are some IoT devices that cannot be suspended during operation. These IoT devices are referred to *non-interruptible IoT devices*, for example, smart dishwashers and smart washing machines. When an EI system operates its own scheduling strategies, the non-interruptable IoT devices should be scheduled continuously (without any suspension). For more clarity, Fig. 2 (a) shows the state diagram of non-interruptable IoT devices. When the IoT device is not in operation in the given time slots, it sleeps. If it receives an operation command from the centralized load controller, it will wait until its scheduled time to be activated for operation.
2) **Interruptible IoT devices.** In contrast with non-interruptible IoT devices, some IoT devices can be suspended during operation. These IoT devices are referred to *interruptible IoT devices*, for example, house Heat Ventilation Air Conditioning (HVAC), water heaters, and Plugged Hybrid Electric Vehicles (PHEVs). They can be scheduled at discrete times in order to avoid peak load increase. Fig. 2 (b) shows the state diagram of interruptible IoT devices; when the IoT device in not in the operation, it sleeps. If the IoT device receives an operation command from the load controller, it will wait until the scheduled time of operation, and then it will be activated. After activation, its operation can be stopped, and then it will operate for the remainder of the jobs.
3) **Emergency IoT devices** There are some IoT devices that should be activated immediately when their operations are initiated. These IoT devices are referred to as *emergency IoT devices*, for example, fire alarm, flame sensor, and flammable gas detector. They should be scheduled immediately since they are time-critical IoT devices. Fig. 2 (c) shows the state diagram of emergency IoT devices. In contrast with interruptible or non-interruptible devices, their operations are not paused when they wake up.

The parameter $\phi_{(m,n)}$ denotes the IoT device controllable type $a_{(m,n)}$ and is represented as

$$\phi_{(m,n)} \in [IT, NIT, ED], \quad (3)$$

where IT stands for an interruptible IoT device, NIT stands for a non-interruptible IoT device, and ED stands for an emergency IoT device. In our system, interruptible and non-interruptible IoT devices may not operate immediately upon receiving an operation request from the system user. Therefore, in this paper, it is assumed that the user can set the tolerance delay for each IoT device ($a_{(m,n)}$), which is denoted by $\tau_{(m,n)}$. For example, if a user wants the operation of $a_{(m,n)}$ to be completed within the next 5 hours and the time slot unit is 1 hour, $\tau_{(m,n)}$ will be set to 5. In this paper, it is assumed that the user can set the threshold of the electricity bill for each EMP broker for a time slot t. For this purpose, $\mathcal{P}^t_m$ is defined as the threshold of the m-th EMP broker for time slot t. For example, the summation of electricity bills for all IoT devices belonging to the m-th EMP broker at time slot t should not exceed γ dollars if $\mathcal{P}^t_m$ is set to γ dollars. It is assumed that all the EMP brokers set $\mathcal{P}^t_m$ to infinity by default, i.e., all admissions will be accepted with a default value. Then, the users can configure the thresholds if they need to reduce the electricity bill.

B. Pricing Decision Models

In our EMP for the EI system, real-time pricing is determined by a centralized EMP agent in the power retailer system. The EMP agent decides the real-time pricing based on the electricity consumption in each time slot. In this paper, E^t denotes the electricity consumption at time slot t, and the total electricity pricing at time slot t (referred to as $B^t(\cdot)$) is described as follows:

$$B^t(E^t) \triangleq \alpha_t \left(E^{t^2}\right) + \beta_t E^t + \gamma_t, \tag{4}$$

where $\alpha_t \geq 0$ and $\beta_t, \gamma_t \geq 0$ at each hour $t \in \mathcal{T}$ [19], [20].

The parameter p^t denotes the real-time pricing. This can be calculated by

$$p^t \triangleq \frac{B^t(E^t)}{E^t}. \tag{5}$$

In this paper, it is assumed that the EMP agent announces the approximated value of p^t before t. In order to approximate p^t, the EMP agent collects the amounts of electricity require by IoT devices from all the deployed EMP brokers. In our EMP for the EI system, the system broadcasts admission requests to all the deployed EMP brokers through their network interface when an IoT device initiates an operation. The admission request of $a_{(m,n)}$ is represented by

$$\mathcal{REQ}_{(m,n)} \triangleq \left[a_{(m,n)}, \phi_{(m,n)}, E^{on}_{(m,n)}, \rho_{(m,n)}, \tau_{(m,n)}\right], \tag{6}$$

where $a_{(m,n)}$ stands for an IoT device that requests admission, $\phi_{(m,n)}$ stands for the IoT device type, $\rho_{(m,n)}$ indicates the required operating time, and $\tau_{(m,n)}$ is a delay tolerance time [2]. In the proposed system, $\tau_{(m,n)}$ decreases at every timeslot, and $\rho_{(m,n)}$ decreases when $\mathcal{REQ}_{(m,n)}$ is operated.

When an EMP broker receives admission requests, it stores the requests in its reservation buffer, which is denoted by $\mathbf{RB}_m$. $\mathbf{RB}_m$ is represented by the set of $\mathcal{REQ}_{(m,n)}$, and can be described by

$$\mathbf{RB}_m = [\mathcal{REQ}_{(m,1)}, \mathcal{REQ}_{(m,2)}, \cdots, \mathcal{REQ}_{(m,|\mathbf{RB}_m|)}], \tag{7}$$

where $|\mathbf{RB}_m|$ is the number of $\mathcal{REQ}_{(m,n)}$ in $\mathbf{RB}_m$.

In each time slot, each EMP broker sends the abstract information regarding the reservation buffer to the EMP agent to obtain real-time pricing. The reason that the EMP broker delivers abstract information instead of its reservation buffer is to prevent information loss due to Internet congestion. In this case, RBA^t_m is referred to as the abstract information of the m-th EMP broker at time slot t, which can be modeled as follows:

$$RBA^t_m \triangleq \left[\mathcal{P}^t_m, MaxE^t_m, MinE^t_m\right], \tag{8}$$

where $MaxE^t_m$ stands for the required maximum electricity amount for IoT devices that belong to the m-th EMP broker at time slot t (i.e., the m-th EMP broker would like to consume $MaxE^t_m$, if available) and $MinE^t_m$ stands for the required minimum electricity amount for IoT devices that belong to m-th EMP broker at time slot t (i.e., m-th EMP broker will at least consume $MinE^t_m$ in the given time slot t).

Then, $MaxE^t_m$ is determined by the m-th EMP broker and can be calculated as follows:

$$MaxE^t_m \triangleq \sum_{i=1}^{|\mathbf{RB}_m|} E^{on}_{(m,i)}. \tag{9}$$

Consequently, $MinE^t_m$ will be also determined by the m-th EMP broker and can be calculated as follows:

$$MinE^t_m \triangleq \sum_{i=1}^{|\mathbf{RB}_m|} \left\lfloor \frac{\rho_{(m,i)}}{\tau_{(m,i)}} \right\rfloor \cdot E^{on}_{(m,i)}. \tag{10}$$

C. Admission Control Models

As mentioned previously, the users in our system are able to set $\mathcal{P}^t_m$ in order to control the final electricity bills. Depending on the determined $\mathcal{P}^t_m$ and p^t, which can be obtained from the EMP agent, each EMP broker affects the limitation of electricity in each time slot t. In this paper, Ψ^t_m is defined as the limitation of electricity in each time slot t, and this Ψ^t_m can be obtained as follows:

$$\Psi^t_m = \frac{\mathcal{P}^t_m}{p^t}. \tag{11}$$

When an EMP broker decides Ψ_t, it conducts its own admission control for the IoT devices in $\mathbf{RB_m}$. The scheduling vector x_i, where $x_i \in \{0,1\}$, determines whether an IoT device a_i is scheduled or not. Obviously, $x_i = 1$ when the IoT device a_i is scheduled, and vice versa. The optimization problem considered in this paper aims to maximize user satisfaction, and it is assumed that the user is satisfied when an IoT device is operated within the pre-defined delay tolerance time. Therefore, the corresponding objective function of the optimization problem can be formulated as:

$$\text{maximize} \sum_{i=1}^{|\mathbf{RB}_m|} \frac{\rho_{(m,i)}}{\tau_{(m,i)}} \cdot x_i, \tag{12}$$

subject to

$$\sum_{i=1}^{|\mathbf{RB}_m|} E^{on}_{(m,i)} \cdot x_i \leq \Psi^t_m, \tag{13}$$

$$x_i \in \{0,1\}, \forall i, \tag{14}$$

where $i \in \{1, \cdots, |\mathbf{RB}_m|\}$. The values of $\rho_{(m,i)}$, $\tau_{(m,i)}$, $E^{on}_{(m,i)}$, and $|\mathbf{RB}_m|$ are taken from $\mathbf{RB}_m$, and Ψ^t_m is calculated using (11).

It is not difficult to see that this is a conventional 0-1 Knapsack problem, given that all the values except x_i are constants. If an IoT device a_i is scheduled (i.e., x_i is determined to be 1) with the optimization framework (12)-(14), this information ($\mathcal{CFM}_{(m,i)}$) is stored in the working buffer ($\mathbf{WB}_m$) of the EMP broker. The working buffer is described by

$$\mathbf{WB}_m = \left[\mathcal{CFM}_{(m,1)}, \cdots, \mathcal{CFM}_{|\mathbf{WB}_m|}\right], \tag{15}$$

where $|\mathbf{WB}_m|$ stands for the number of $\mathcal{CFM}_{(m,i)}$ in $\mathbf{WB}_m$. Before starting time slot t, the $\mathcal{CFM}_{(m,i)}$ in $\mathbf{WB}_m$ are delivered to the IoT devices in order to announce the results of the admission control procedures.

[2] Since the EI system can become larger with thousands or even millions of devices, the users cannot completely set all the delay tolerances of the IoT devices. Therefore, we assume that the default delay tolerance of the IoT device can be set according to the application. Then, the users can configure the delay tolerances of the IoT device according to their preference.

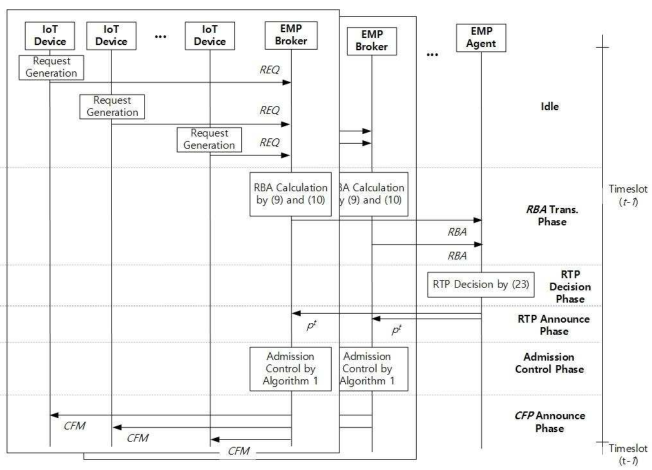

Fig. 3: Phases in each time slot.

The proposed model can also be applied to energy harvesting environments. If the IoT device is equipped with an energy generation device and a battery, the generated energy can be stored in its battery. In this environment, $E^{on}_{(m,n)}$ of the admission request should be recalculated since the device can use the harvested energy. Accordingly, let $h^t_{(m,n)}$ denotes the amount of stored electricity of IoT device $a_{(m,n)}$ at timeslot t. Then, $E^{on}_{(m,n)}$ is updated as follows:

$$E^{on}_{(m,n)} \Leftarrow \max\left(\frac{E^{on}_{(m,n)} \cdot \rho_{(m,n)} - h^t_{(m,n)}}{\rho_{(m,n)}}, 0\right) \tag{16}$$

III. IoT device Scheduling Algorithms

In this section, we provide detailed descriptions of the proposed IoT device scheduling algorithms.

Fig. 3 shows the detailed operations of our system in each time slot. As shown in Fig. 3, each time slot is divided into multiple phases, including idle, reservation buffer abstract (RBA) transmission phase, real-time pricing (RTP) decision phase, RTP announcement phase, admission control phase, and confirmation (CFM) announcement phase. For all of the defined phases in each time slot, an IoT device can send an admission request message to the EMP broker through its network interface. However, only requests that are not sent until the RBA transmission phase are considered for approximating the real-time pricing. In the RBA transmission phase, each EMP broker transmits RBA^t_m to the centralized EMP agent through the Internet. According to (8), (9), and (10), the EMP broker generates RBA^t_m from $\mathbf{RB_m}$.

During the real-time pricing decision phase, the proposed EMP agent calculates the real-time pricing information for the operation of the next time slot. In this formulation, k^t_m denotes the amount of electricity expected for the IoT devices that belong to the m-th EMP broker in each time slot t. RBA^t_m contains the upper limit of the electricity bill in each time slot t. Based on this definition, the following constraint can be derived:

$$k^t_m \cdot p^t \leq \mathcal{P}^t_m. \tag{17}$$

Algorithm 1: IoT device scheduling (admission control for IoT devices) for EMP broker

```
1:  procedure IoT DEVICE SCHEDULING($m$-th EMP broker)
2:      Step 1: Transfer $\mathbf{WB}_m$ to $\mathbf{RB}_m$
3:      for $i = 0$ to $|\mathbf{WB}_m|$ do
4:          if $\phi_{(m,i)} = IT$ then
5:              update $REQ_{(m,i)}$ from $\mathbf{WB}_m$ to $\mathbf{RB}_m$
6:          end if
7:          $i \Leftarrow i + 1$
8:      end for
9:      Step 2: Emergency IoT device scheduling
10:     for $i = 0$ to $|\mathbf{RB}_m|$ do
11:         if $\phi_{(m,i)} = ED$ then
12:             update $REQ_{(m,i)}$ from $\mathbf{RB}_m$ to $\mathbf{WB}_m$
13:         end if
14:         $i \Leftarrow i + 1$
15:     end for
16:     Step 3: Calculate size of Knapsack
17:     $\psi^t \Leftarrow \Psi^t_m$
18:     for $i = 0$ to $|\mathbf{WB}_m|$ do
19:         $\psi^t \Leftarrow \psi^t - E^{on}_i$
20:         $i \Leftarrow i + 1$
21:     end for
22:     if $\psi^t \leq 0$ then
23:         Alarm to user
24:         go to 35
25:     end if
26:     Step 4: Scaling the electric power
27:     set $\eta$ as GCD among $\psi^t$, $E^{on}_{(m,1)}$, $E^{on}_{(m,2)}$, $\cdots$, $E^{on}_{(m,|\mathbf{RB}|_m)}$
28:     $\psi^t \Leftarrow \psi^t \div \eta$
29:     for $i = 0$ to $|\mathbf{RB}_m|$ do
30:         $\omega_i \Leftarrow E^{on}_{(m,i)} \div \eta$
31:         $i \Leftarrow i + 1$
32:     end for
33:     Step 5: Admission control by dynamic programming
34:     DP($|\mathbf{RB}_m|, \psi_t, \mathbf{w}, \mathbf{v}, \mathbf{r}$)
35: end procedure
```

In addition to this constraint, an additional constraint can be consequently defined for the maximum and minimum of electricity required amounts as follows:

$$MinE^t_m \leq k^t_m \leq MaxE^t_m, \tag{18}$$

where $MaxE^t_m$ and $MinE^t_m$ are obtained by RBA^t_m. According to (4) and (5), the approximated real-time pricing (p^t) can be calculated using the following constraint:

$$p^t \geq \frac{B^t(\sum_{m=1}^{M} k^t_m)}{\sum_{m=1}^{M} k^t_m}, \tag{19}$$

where $B^t(\cdot)$ is obtained by (4).

In our proposed system, it is assumed that all the EMP brokers want to consume as much electricity as possible.

Therefore, the proposed EMP agent utilizes the following optimization problem:

$$\text{maximize} \sum_{m=1}^{M} k_m^t, \tag{20}$$

subject to

$$k_m^t \cdot p^t \leq \mathcal{P}_m^t, \forall m \tag{21}$$

$$MinE_m^t \leq k_m^t \leq MaxE_m^t, \forall m \tag{22}$$

$$p^t \geq \frac{B^t(\sum_{m=1}^{M} k_m^t)}{\sum_{m=1}^{M} k_m^t}. \tag{23}$$

In order to obtain the solutions for this optimization problem, i.e., by calculating the values k_m^t, the proposed EMP agent utilizes software tools such as MOSEK [21] or CPLEX [22]–[24]; these tools are widely used for solving optimization problems in power electronics systems and networks. When the proposed EMP agent finally obtains the optimal k_m^t, it can find the real-time pricing information as follows:

$$p^t \triangleq \frac{B^t(\sum_{m=1}^{M} k_m^t)}{\sum_{m=1}^{M} k_m^t}, \tag{24}$$

where $B^t(\cdot)$ is as modeled in (4).

During the real-time pricing announcement phase, the EMP agent broadcasts the approximation of p^t (which is calculated using (24)) to all the EMP brokers through the Internet.

During the admission control phase, each EMP broker conducts admission control operations using its $\mathbf{RB}_m$ and p^t, which are received from the centralized EMP agent. As mentioned in Section II-C, the optimization problem for IoT device scheduling can be modeled using a 0-1 Knapsack problem under pricing limitations and delay tolerance. However, solving the Knapsack problem using brute-force algorithms is well known to be NP-hard because brute-force algorithms (a.k.a., full search algorithms) introduce a combinatorial order of the search space. In order to resolve this issue, an improved method using dynamic programming for solving the 0-1 Knapsack problem in pseudo polynomial-time, while ensuring optimal solutions [25], [26], is used.

The pseudo-code in Algorithm 1 shows the detailed procedure of the dynamic programming for admission control in the EMP brokers. The IoT device scheduling algorithm consists of following steps:

Step 1: In the first part of the algorithm, the interruptible IoT devices in $\mathbf{WB}_m$ are moved to $\mathbf{RB}_m$, since they can be rescheduled (lines 2-8 in Algorithm 1).

Step 2: Since the priorities of emergency IoT devices are the highest, the EMP broker has to accept the requests from the emergency IoT devices, i.e., each EMP broker moves the emergency IoT devices in $\mathbf{RB}_m$ to $\mathbf{WB}_m$ (lines 9-15 in Algorithm 1).

Step 3: On the basis of the updated reservation buffer and the upper limit of electricity, each EMP broker recalculates the amounts of available electricity (ψ_t) in order to utilize the Knapsack formulation (lines 18-21 in Algorithm 1). In this step, the electricity bill can be greater than the threshold due to the requests from the emergency IoT devices. If the electricity bill exceeds the threshold, the EMP broker will generate an alarm to inform to the user and stop device scheduling (lines 22-25 in Algorithm 1).

Step 4: To solve the dynamic programming for the given Knapsack formulation efficiently, scaling is conducted by calculating the greatest common divisor (GCD) among the Knapsack size and the power consumed in $\mathbf{RB}_m$ (lines 26-32 in Algorithm 1)

Step 5: After the scaling procedure, each EMP broker performs the dynamic programming algorithm. In order to design the dynamic programming framework, the following notations are introduced:

- $\mathbf{w}$ is defined for the set of ω_i, and all ω_i are computed by the scaling step.
- $\mathbf{v}$ is defined for the set of $\tau_{(m,i)}/\rho_{(m,i)}$ in $\mathbf{RB}_m$.
- $\mathbf{r}$ is used for describing the set of admission results r_i, which indicate whether or not the admission of IoT device $a_{(m,i)}$ is confirmed, for all IoT devices. That is, if the admission of $a_{(m,i)}$ is confirmed, r_i is set to 1. Otherwise, r_i is set to 0. At the beginning of the dynamic programming algorithm operation, all r_i are initialized to 0.

The optimal solutions of the proposed optimization problem (12)-(14) are obtained using the dynamic programming framework, and the recursive formulation for the dynamic programming is given as follows:

$$f_j(y) = \max\left[f_{j-1}(y), f_{j-1}(y-E_{(m,j)}^{on}) + \frac{\tau_{(m,j)}}{\rho_{(m,j)}}\right], \tag{25}$$

$$f_0(y) = 0, \tag{26}$$

$$\forall y \geq 0, \tag{27}$$

where y stands for the size of the Knapsack (i.e., y is set to ψ_t) and j in (25) stands for the number of scheduled IoT devices. The detailed procedure for solving the proposed dynamic programming framework is presented in [25], [26].

During the CFM announcement phase, each EMP broker broadcasts all $\mathcal{CFM}_{(m,n)}$ information to the IoT devices. If an IoT device receives a confirm message through its network interface, it will operate during the next time slot (i.e., by having the IoT device *scheduled* for the next time slot). In addition, each EMP broker updates the timer parameter for the reservation buffer and the working buffer. For the delay tolerance values, each EMP broker reduces (i.e., $\tau_{(m,n)} \leftarrow \tau_{(m,n)} - 1, \forall n$) from the reservation buffer and working buffer. For the operation time values, each EMP broker reduces (i.e., $\rho_{(m,n)} \leftarrow \rho_{(m,n)} - 1, \forall n$) from the working buffer. If $\rho_{(m,n)}$ becomes 0, the EMP broker will remove its request from the buffer (i.e., timeout).

IV. Performance Evaluation with Simulations

The performance of our IoT device scheduling scheme is evaluated through comprehensive simulations. For our performance evaluation, an event-driven simulator for dynamic programming is implemented using the C programming language. In this evaluation, we focus on a) the price convergence

TABLE I: Simulation Parameters

Parameter		Value
M		500
$N_m, \forall m$		20
Mean of inter-arrival time (Poisson distribution)		10
Unit of time slot		1 hour
$E^{on}_{(m,n)}, \forall m, n$ (Uniform distribution)		From $1w$ to $100w$
$\rho_{(m,n)}, \forall m, n$ (Uniform distribution)		From 1 to 10
$\tau_{(m,n)}, \forall m, n$ (Uniform distribution)		From 1 to 20
Topology 1	Ratio of IT	0.45
	Ratio of NIT	0.45
	Ratio of ED	0.1
Topology 2	Ratio of IT	0.8
	Ratio of NIT	0.1
	Ratio of ED	0.1
Topology 3	Ratio of IT	0.1
	Ratio of NIT	0.8
	Ratio of ED	0.1
Topology 4	Ratio of IT	0.3
	Ratio of NIT	0.3
	Ratio of ED	0.4

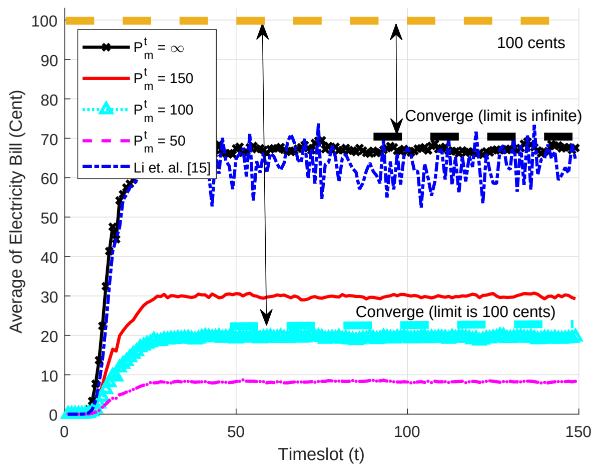

Fig. 4: Average of electricity bill over time for Topology 1.

(electricity bill), b) the impacts of admission requests and their frequency on the average electricity bill and accepted IoT device ratio, c) the peak electricity consumption capabilities (i.e., load balancing) of our scheme, and d) the electricity bill and accepted IoT device ratio for the energy harvesting environments. Moreover, the scheme presented in [15] is compared with the proposed scheme in terms of the electricity bill. In addition, the schemes in [12], [13] are compared with regard to the energy harvesting environments.

Parameters. In order to compute the quadratic cost function, which is modeled as (4), we assume that $\alpha_t = 5/9$, $\beta_t = 0$, and $\gamma_t = 0$ [19]. The settings and parameters used in our comprehensive simulation are shown in Table I. These parameters are used for analyzing the impact of $\mathcal{P}^t_m$.

When $\mathcal{P}^t_m$ is too small, it is difficult to confirm the admission requests. On the other hand, when $\mathcal{P}^t_m$ is sufficiently large, the electricity bill will not be decreased. In the event-driven simulation, requests that have passed the delay tolerance but have not received admission confirmation will be removed from the reservation buffer. If the delayed requests remain in the buffer, it will be difficult to evaluate the performance because the delayed requests will accumulate in the buffer. Moreover, the proposed algorithm is evaluated for the various topologies to measure the impact of the IoT device type ratio. Accordingly, in Topologies 1, 2, and 3, we measure the performance according to the distributions of NIT and IT, whereas Topology 4 is designed to measure the impact of emergency data generation (ED).

Price convergence. Fig. 4 shows the average electricity bill per EMP broker graph according to the time slot in Topology 1. In our simulation, $\mathcal{P}^t_m = \infty$ means that the proposed scheme is not being applied (i.e., if an IoT device generates operation requests, electricity will be consumed immediately during the requested operation time). As shown in Fig. 4, it can be observed that the electricity bills increase with time, for each time slot, and converge at some time slot. This is because there are no requests at the beginning of the simulation, and the generation of requests also converges to a fixed number.

From the same result, we can also see that the proposed scheme reduces the electricity bills in comparison to the case where the proposed scheme is not applied (i.e., $\mathcal{P}^t_m = \infty$). The proposed scheme, with $\mathcal{P}^t_m = 150$, $\mathcal{P}^t_m = 100$, and $\mathcal{P}^t_m = 50$, reduce the electricity bills as much as 57%, 72%, and 90%, respectively, in comparison to the case with $\mathcal{P}^t_m = \infty$, as clearly presented in Fig. 4. In particular, although the electricity bill with $\mathcal{P}^t_m = \infty$ is less than 100 cents, the bill with $\mathcal{P}^t_m = 100$ is still reduced to 72% compared to the bill with $\mathcal{P}^t_m = \infty$. The reason for this drastic reduction in the electricity bill is that the peak electricity consumption has a significant impact on the bill due to the fact that the pricing model is designed as a quadratic function. Furthermore, the average electricity bill per EMP broker is much smaller than $\mathcal{P}^t_m$, since the IoT device does not always generate a request. If the IoT device always generates a request, the average electricity bill per EMP broker will obviously be $\mathcal{P}^t_m$. In contrast with the results of the proposed scheme, the result of the [15] fluctuates over time. Since the comparison scheme is designed for day-ahead scheduling, it is not applicable to the time-varying EI environment. Moreover, the comparison scheme does not reduce the electricity bill significantly since it does not conduct admission control, i.e., all requests are permitted.

Admission requests. In Fig. 5, the performance of the proposed scheme and comparison scheme is presented with respect to the admission request frequency for the various topologies. In this figure, the average of the electricity bill per EMP broker is illustrated according to the inter-arrival time of the admission requests. As shown in Fig. 5, the electricity bills increase exponentially when the admission requests increase. In addition, it can be also observed that the electricity bills with smaller $\mathcal{P}^t_m$ are less than the bills with larger $\mathcal{P}^t_m$. This is mainly due to the fact that the proposed scheme suppresses the peak demands through $\mathcal{P}^t_m$. Moreover, the proposed scheme shifts the electricity loads (by indirectly performing load-balancing over time). Instead, the comparison

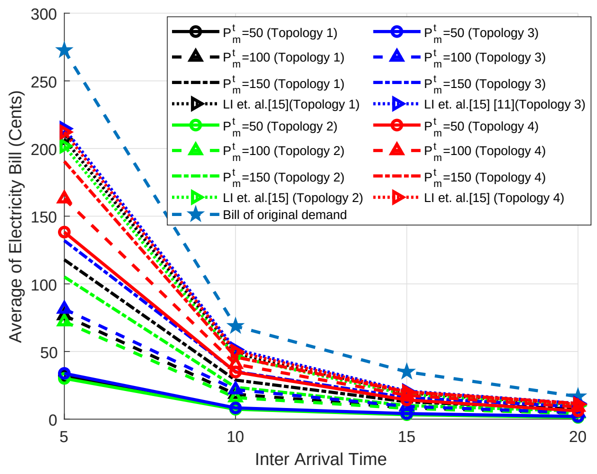

Fig. 5: Average electricity bill in relation with the inter-arrival time. The inter-arrival time captures the admission request frequency/rate.

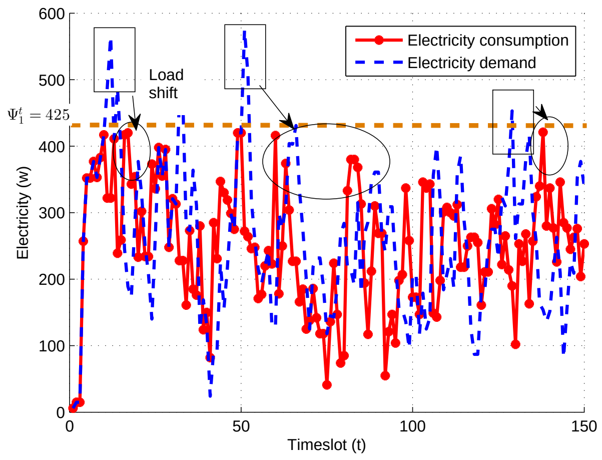

Fig. 7: Electricity consumption over time, highlighting our scheme's capabilities in controlling peak consumption through consumption control and load-balancing.

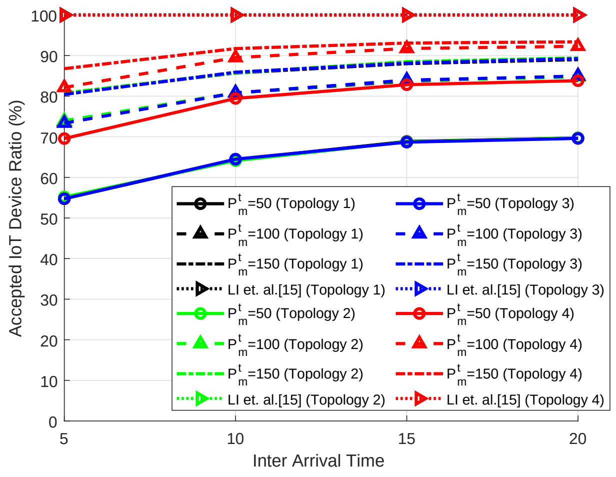

Fig. 6: Accepted IoT device ratio versus inter-arrival time.

scheme does not significantly reduce the electricity bill as much as the proposed scheme. The proposed scheme controls the energy consumption by the threshold $\mathcal{P}_m^t$. However, the comparison scheme only conducts the load shifting, i.e., all requests will be operated. Furthermore, the proposed scheme efficiently reduces the electricity bill as the number of non-interruptible devices decreases since they are not rescheduled. Moreover, we can observe that the proposed scheme reduces the electricity bill more efficiently if emergency data are not frequently generated. This is due to the reason that the emergency data have a higher priority than the threshold in the proposed scheme. As well as, when the emergency data is frequently generated, the proposed scheme provides a larger reduction in electricity bill than the comparison scheme.

As presented in Fig. 6, the accepted IoT device ratio according to the admission request frequencies was simulated and evaluated. In this figure, the accepted IoT device ratio increases when $\mathcal{P}_m^t$ increases. When a user sets $\mathcal{P}_m^t$ sufficiently low, the accepted ratio will be very low even if requests are not generated frequently. It is also noticeable that significant number of service provisions is not available when $\mathcal{P}_m^t$ is equal to 50. Therefore, $\mathcal{P}_m^t$ should be defined and set above a certain value in order to provide the expected level of service. The accepted ratio is over 90 % when the inter-arrival time is set to 15 and $\mathcal{P}_m^t$ is set to 150. On the other hand, a lower accepted IoT device ratio is observed when the inter arrival rate increases. It is obvious that in order to provide the desired level of service, $\mathcal{P}_m^t$ should increase according to the increase in admission request generation. Therefore, it is necessary to set appropriate $\mathcal{P}_m^t$ values according to the amount of request generations. We can also observe an interesting result that the accepted IoT device ratio increases if the amount of generated emergency data increases, i.e., the accepted ratio of Topology 4 is greater than those of Topologies 1, 2, and 3. In the proposed scheme, the IT and NIT type devices can be rejected, and ED type devices have to be permitted. In Topology 4, the number of ED is proportionally larger than the number of IT and NIT. Therefore, increasing the number of ED does not affect the number of rejection rather it only increases the electricity bill in the proposed scheme.

Peak avoidance. Next, we evaluate the performance of our scheme in terms of avoiding electricity peaks to reduce the electricity bills. For this evaluation, the following settings are used: $M = 1$, $N_1 = 20$, and $\mathcal{P}_1^t = 0.1$. M is set to 1 (i.e., single) because it is difficult to observe the shifting of the load since multiple electricity requests and consumptions will be combined in the graph. Accordingly, $\mathcal{P}_1^t$ is set to 0.1. When $\mathcal{P}_1^t$ is set to 0.1, Ψ_1^t will be 450 watt, according to (4). Based on this setting, Fig. 7 shows the electricity demands and electricity consumptions per time slot. As presented in Fig. 7, the electricity demand exceeds Ψ_1^t at $t = 10$, and the proposed scheme efficiently shifts the electricity loads to the spare time slot (i.e., $t = 16$ in Fig. 7). When electricity is directly consumed at $t = 10$, the electricity bill will increase. Therefore, the proposed scheme efficiently avoids

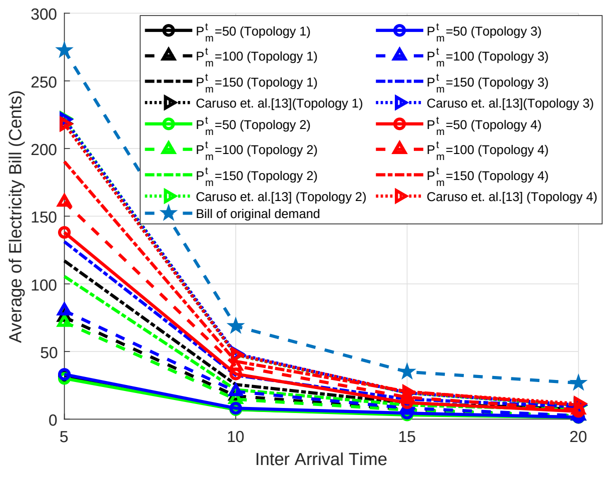

Fig. 8: Average electricity bill in relation with the inter-arrival time in the energy harvesting environment. The scenario of the energy harvesting [13] is applied.

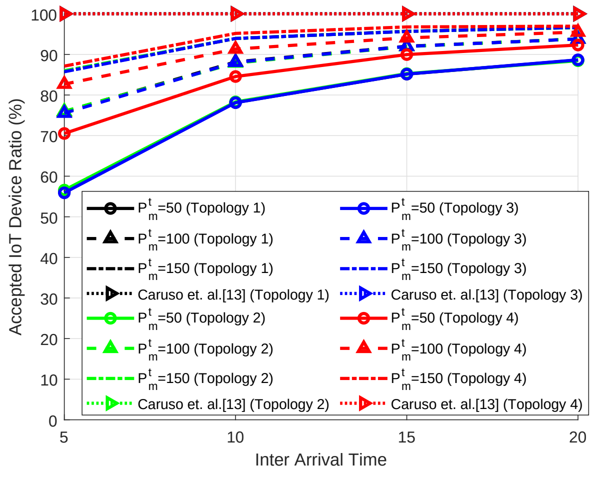

Fig. 9: Accepted IoT device ratio versus inter-arrival time in the energy harvesting environment. The scenario of the energy harvesting [13] is applied.

peaks in electricity consumption via load shifting (see the black rectangles in Fig. 7).

Energy Harvesting Environment. As mentioned in Section II, the proposed algorithm can be operated in energy harvesting environments. For this evaluation, the energy harvesting scenario, that is presented in [13], is applied. In addition, the comparison schemes of [12], [13] are modified as follows. In the original schemes of [12], [13], the device operates with only the battery since the operation requests are generated intermittently. However, in this simulation, the device operations occur frequently as shown in Table I. Therefore, the schemes of [12], [13] were modified to buy insufficient electricity from the power retailer. Fig. 8 and Fig. 9 show the average electricity and the accepted IoT device ratio according to the inter-arrival time, respectively. As shown in Fig. 8, the proposed scheme efficiently controls the energy consumption according to the threshold $\mathcal{P}^t_m$. On the other hand, the comparison scheme schedules tasks based on their quality, under the battery and energy harvesting constraints. As mentioned above, the comparison scheme is modified that the insufficient energy is bought from the power retailer. Therefore, when the comparison scheme is applied, all generated requests will be operated. In addition, the electricity bill is determined based on the peak energy consumption in the proposed system. Therefore, the comparison scheme does not sufficiently reduce the electricity bill as much as the proposed scheme. In this figure, we can observe that the electricity bill of [13] is not affecting the topology. This is due to the reason that all request are accepted and consumed the energy. On the other hand, the proposed scheme rejects the operation requests according to the device type. For example, in the topology 4, the electricity bill is larger than others since the emergency devices are distributed more than other topologies, i.e., more requests are accepted. Moreover, the results show that the proposed scheme's electricity bill is similar to the scenarios, in which the energy harvesting model is not applied. This is because the electricity bill of the proposed scheme is managed according to $\mathcal{P}^t_m$. In other words, the energy harvesting does not affect the electricity bill in the proposed scheme when the energy consumption is significantly larger than the harvested energy. However, the accepted ratio increases in the energy harvesting model as shown in Fig. 9. Especially, for the inter arrival time of 20, the accepted ratio of the energy harvesting scenario is up to 20% greater than the ratio of the scenario, in which the energy harvesting is not applied. This is because that $E^{on}_{(m,n)}$ is reduced by the harvested energy, and thus more requests can be accepted. In other words, the required electricity is reduced but the threshold is keep the same.

V. Concluding Remarks and Future Work

In this paper, we proposed a two-stage approach with dynamic programming for IoT device scheduling in EI systems using optimization formulations and their corresponding solver algorithms. The optimization framework for IoT device scheduling was formulated using real-time pricing decision models. Noting that the proposed optimization framework is equivalent to a 0-1 Knapsack problem which considers the delay tolerance and real-time pricing of all given IoT devices, a two-stage algorithm was designed with dynamic programming to solve the 0-1 Knapsack problem in pseudo-polynomial-time. The first stage in our algorithm includes procedures for determining the maximum electricity consumption in each time slot, while the second stage provides tools for computing the IoT device scheduling in each time slot. In order to evaluate the performance of the proposed algorithm, intensive simulations were conducted. As a result, it was shown that the proposed algorithm efficiently reduces electricity bills with appropriate parameter settings.

As a future research direction, the design and implementation of adaptive decision policies for key parameters such as threshold based on IoT device usage patterns using machine learning algorithms [27] will be pursued.

References

[1] T. Chiu, Y. Shih, A. Pang, and C. Pai, "Optimized Day-Ahead Pricing With Renewable Energy Demand-Side Management for Smart Grids," *IEEE IoT Journal,* Vol. 4, No. 2, April 2017, pp. 374–383.

[2] P. Gope and B. Sikdar, "An Efficient Data Aggregation Scheme for Privacy-Friendly Dynamic Pricing-Based Billing and Demand-Response Management in Smart Grids," *IEEE IoT Journal,* Vol. 5, No. 4, Aug. 2018, pp. 3126–3135.

[3] L. Yu, D. Xie, T. Jiang, Y. Zou, and K. Wang, "Distributed Real-Time HVAC Control for Cost-Efficient Commercial Buildings Under Smart Grid Environment," *IEEE IoT Journal,* Vol. 5, No. 1, Feb. 2018, pp. 44–55.

[4] L. de M. B. A. Dib, V. Fernandes, M. de L. Filomeno and M. V. Ribeiro, "Hybrid PLC/Wireless Communication for Smart Grids and Internet of Things Applications,"*IEEE IoT Journal,* Vol. 5, No. 2, April 2018, pp. 655–667.

[5] H.Zhang, Y. Li, D. W. Gao, and J. Zhou, "Distributed Optimal Energy Management for Energy Internet," *IEEE Transactions on Industrial Informatics,* Vol. 13, No. 6, December 2017, pp. 3081–3097.

[6] X. Li and J.-C. Lo, "Pricing and Peak Aware Scheduling Algorithm for Cloud Computing," in *Proceedings of the IEEE Conference on Innovative Smart Grid Technologies*, January 2012, pp. 17.

[7] M. Muratori and G. Rizzoni, "Residential Demand Response: Dynamic Energy Management and Time-Varying Electricity Pricing," *IEEE Transactions on Power Systems*, Vol. 31, No. 2, March 2016, pp. 1108–1117.

[8] D. T. Nguyen, H. T. Nguyen, and L. B. Le, "Dynamic Pricing Design for Demand Response Integration in Power Distribution Networks," *IEEE Transactions on Power Systems*, Vol. 31, No. 5, September 2016, pp. 3457–3472.

[9] K. McKenna and A. Keane, "Residential Load Modeling of Price-Based Demand Response for Network Impact Studies," *IEEE Transactions on Smart Grid*, Vol. 7, No. 5, September 2016, pp. 2285–2294.

[10] J. Yao and N. Ansari, "Caching in Energy Harvesting Aided Internet of Things: A Game-Theoretic Approach," to appear in *IEEE Internet of Things Journal*, 2019.

[11] A. Shahini, A. Kiani, and N. Ansari, "Energy Efficient Resource Allocation in EH-enabled CR Networks for IoT," to appear in *IEEE Internet of Things Journal*, 2019.

[12] S. Escolar, A. Caruso, S. Chessa, X. d. Toro, F. J. Villanueva, and J. C. Lopez, "Statistical Energy Neutrality in IoT Hybrid Energy-Harvesting Networks," *in proc. of ISCC 2018*, June 2018, pp. 444–449.

[13] A. Caruso, S. Chessa, S. Escolar, X. d. Toro, and J. C. Lopez, "A Dynamic Programming Algorithm for High-Level Task Scheduling in Energy Harvesting IoT," *IEEE Internet of Things Journal*, Vol. 5, No. 3, June 2018, pp. 2234–2248.

[14] M. Latifi, A. Khalili, A. Rastegarnia, and S. Sanei, "Fully Distributed Demand Rsponse Using the Adaptive Diffusion-Stackelberg Algorithm," *IEEE Transactions on Industrial Informatics,* May 2017, pp.2291-2031.

[15] C. Li, X. Yu, W. Yu, G. Chen, and J. Wang, "Efficient Computation for Sparse Load Shifting in Demand Side Management," *IEEE Transactions on Smart Grid,* Vol.8, No. 1, January 2017, pp. 250–261.

[16] P. Srikantha and D. Kundur, "Resilient Distributed Real-Time Demand Response via Population Games," *IEEE Transactions on Smart Grid,* Vol. 8, No. 6, November 2017, pp. 2532–2543.

[17] C. Lee, L. Park, and S. Cho, "Light-Weight Stackelberg Game Theoretic Demand Response Scheme for Massive Smart Manufacturing Systems," *IEEE Access,* Vol. 6, No. 1, April 2018, pp. 23316–23324.

[18] A. E. Shadare, M. N. O. Sadiku, and S. M. Musa, "Electromagnetic compatibility issues in critical smart grid infrastructure," *IEEE Electromagnetic Compatibility Magazine,* Vol. 6, No. 4, 2017, pp. 63–70.

[19] A. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous Demand-Side Management Based on Game-Theoretic Energy Consumption Scheduling for the Future Smart Grid," *IEEE Transactions on Smart Grid,* Vol. 1, No. 3, December 2010, pp. 320–331.

[20] F. Kamyab, M. Amini, S. Sheykhha, M. Hasanpour, and M. M. Jalali, "Demand Response Program in Smart Grid using Supply Function Bidding Mechanism," *IEEE Transactions on Smart Grid,* Vol. 7, No. 3, May 2016, pp. 1277–1284.

[21] M. R. Dorostkar-Ghamsari, M. Fotuhi-Firuzabad, M. Lehtonen, and A. Safdarian, "Value of Distribution Network Reconfiguration in Presence of Renewable Energy Resources," *IEEE Transactions on Power Systems*, Vol. 31, NO. 3, May 2016, pp. 1879–1888.

[22] M. Silbernagl, M. Huber, and R. Brandenberg, "Improving Accuracy and Efficiency of Start-Up Cost Formulations in MIP Unit Commitment by Modeling Power Plant Temperatures," *IEEE Transactions on Power Systems*, Vol. 31, No. 4, July 2016, pp. 2578–2586.

[23] W. Yao, J. Zhao, F. Wen, Y. Xue, and G. Ledwich, "A Hierarchical Decomposition Approach for Coordinated Dispatch of Plug-in Electric Vehicles," *IEEE Transactions on Power Systems*, Vol. 28, No. 3, August 2013, pp. 2768–2778.

[24] F. Ceja-Gomez, ,S. S. Qadri, and F. D. Galiana, "Under-Frequency Load Shedding Via Integer Programming," *IEEE Transactions on Power Systems*, Vol. 27, No. 3, August 2012, pp. 1387–1394.

[25] J. Kleinberg and E. Tardos, *Algorithm Design*, Pearson, March 2005.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 2009.

[27] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.