# Pattern-Identified Online Task Scheduling in Multi-tier Edge Computing for Industrial IoT Services

Nhu-Ngoc Dao[a], Duc-Nghia Vu[a], Yunseong Lee[a], Sungrae Cho[a,*], Chihyun Cho[b], Hyunbum Kim[c]

[a]*School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea*
[b]*Samsung Research, Samsung Electronics Co., Ltd. Seoul 06765, Republic of Korea*
[c]*Department of Computer Science, University of North Carolina at Wilmington, Wilmington, NC 28403, USA*

**Abstract**

In smart manufacturing, production machinery and auxiliary devices, referred to as industrial Internet of things (IIoT), are connected to a unified networking infrastructure for management and command deliveries in a precise production process. However, providing autonomous, reliable, and real-time offloaded services for such a production is an open challenge since these IIoT devices are assumed lightweight embedded platforms with limited computing performance. In this paper, we propose a pattern-identified online task scheduling (PIOTS) mechanism for the networking infrastructure, where multi-tier edge computing is provided, in order to handle the offloaded tasks in real-time. First, historical IIoT task patterns in every timeslot are used to train a self-organizing map (SOM), which represents the features of the task patterns within defined dimensions. Consequently, offline task scheduling among edge computing-enabled entities is performed on the set of all SOM neurons using the Hungarian method to determine the expected optimal task assignments. In real-time context, whenever a task arrives at the infrastructure, the expected optimal assignment for the task is scheduled to the appropriate edge computing-enabled entity. Numerical simulation results show that the proposed PIOTS mechanism overcomes existing solutions in terms of computation performance and service capability.

*Keywords:* industrial Internet of things, online task scheduling, edge computing

## 1. Introduction

Recently, smartization in manufacturing has been considered as one of major trends realizing the fourth industrial revolution [1]. In the smart factory, production machinery and auxiliary devices, which are referred to as industrial Internet of things (IIoT), maintain permanent connections to a unified networking infrastructure, where Internet service is available. In this environment, all of the connected IIoT devices acquire mutual cooperation with each other and they request the working commands from central management entities in the network. For instance, precise production processes require trigger feedback from control entities to adjust reactions of machines and robots if an unexpected issue occurs. Another example is when smoke is detected in a warehouse, a real-time shutdown command should be immediately dispatched to electrical working chains, and a real-time activation command should force the fire extinguishing system to be activated [2]. However, since these IIoT devices are assumed to be lightweight embedded platforms with limited computing resources, it is inappropriate to execute real-time services using its own power of the devices.

Fortunately, thanks to the advances of the emerging fifth-generation (5G) technologies, the 5G mobile edge computing (MEC) can provide autonomous, reliable, and real-time offloaded services for these IIoT devices and applications. Defined by the European telecommunications standards institute (ETSI), the MEC provides *cloud-computing capabilities and an IT service environment at the edge of the network* [3]. To be more spe-

[*]Correspondence should be addressed to Sungrae Cho (srcho@cau.ac.kr).
[**]Part of this work was presented at ICTC conference in Jeju, Korea, Oct. 18–20, 2017.
*Email addresses:* `dnngoc@uclab.re.kr` (Nhu-Ngoc Dao), `dnvu@uclab.re.kr` (Duc-Nghia Vu), `yslee@uclab.re.kr` (Yunseong Lee), `srcho@cau.ac.kr` (Sungrae Cho), `ch79.cho@samsung.com` (Chihyun Cho), `kimh@uncw.edu` (Hyunbum Kim)

cific, multi-tier MEC (mMEC) is a virtualized hierarchical MEC framework where edge computing-enabled entities (ECEs) are layered according to its computing performance. The operation of mMEC framework is supervised by a central orchestrator [4]. The orchestrator schedules the tasks offloaded from IIoT devices among ECEs in order to maximize computation performance and decrease energy consumption while keeping the IIoT applications' requirements.

Aiming at this objective, we propose a pattern-identified online task scheduling (PIOTS) mechanism for the mMEC framework in smart factory for handling the offloaded IIoT tasks in real-time. First, historical IIoT task patterns in every timeslot are used to train a self-organizing map (SOM), which represents the features of the task patterns within defined dimension [5]. As a result, a set of typical SOMs for IIoT task patterns are identified. At the beginning of each timeslot, offline task scheduling among ECEs is performed on the neuron set of the typical SOM using the Hungarian method to determine expected optimal task assignments. Thereafter, whenever a task arrives at the mMEC framework, the task is matched into the current SOM to find the best matching SOM neuron for it. Based on that, the expected optimal assignment for this matched SOM neuron is scheduled for the task, assigning to appropriate ECE in online manner. The main contributions of this paper are three-fold:

- First, expected optimal assignments are scheduled for offloaded IIoT tasks arrived at the mMEC framework in real-time. In other words, a task assignment delay is reduced significantly.
- Second, the duration of task assignment calculation is decreased because this operation works on the SOM neuron set within defined dimension. The neuron set is a representative of the historical incoming tasks, which is not greater than the size of real task patterns in each timeslot. As a result, the computational complexity and calculation latency are reduced as well.
- Third, the expected optimal task assignment calculation is performed independently on the task scheduling timeline. Therefore, the task assignment can be operated continuously in real-time without waiting for the task assignment calculation to be completed.

The remainder of this paper is organized as follows. Section 2 surveys the state-of-the-art related work. Section 3 describes the proposed pattern-identified online task scheduling mechanism in detail. Section 4 provides the system setup, evaluation methodology, and evaluation metric definitions. Based on that, the performance of the PIOTS mechanism in comparison to other techniques is analyzed in Section 5. Finally, the paper is concluded in Section 6.

## 2. Related Work

Optimal task assignments in multi-tier edge computing have been classified into three main categories: latency awareness, energy awareness, and quality-of-services (QoS) awareness including their variants [6, 7].

The latency-aware approaches focus on minimizing execution latency of the task offloaded to the edge servers. The execution latency involves three portions: transmission duration from the IIoT devices to the edge servers, queuing and processing duration at the edge servers, and return duration for successful reception of the result in the IIoT devices [8, 9]. The mMEC framework can be designed and supported by several emerging technologies such as network functions virtualization (NFV) and software defined networking (SDN). In [10], Dao *et al.* proposed an adaptive balancing scheme (ARB) to distribute tasks among edge servers in the remote radio heads in order to improve the serviceability of the network, especially in term of task execution latency. The ARB scheme combines the Hungarian method and the backpressure algorithm for this purpose. In [11], Mao *et al.* aim at reducing the task execution latency by using a low-complexity Lyapunov optimization-based dynamic computation offloading (LODCO) algorithm. The LODCO algorithm handles CPU utilization in the edge servers according to the task arrival. On the other hand, Liu *et al.* [12] considered application buffer queuing state, available computing powers, and channel quality to conduct the optimal offloading decision. The tasks will be assigned to the edge servers if the total execution latency made by the edge server is shorter than the duration spent to execute the task locally by the IIoT devices. Otherwise, the IIoT devices will execute their tasks themselves.

The energy-aware approaches aim at minimizing energy consumption for task execution in edge computing. Typically, the energy consumption is considered on task delivery and task computation. In [13], an energy reduction method was proposed based on the Bak-Tang-Wiesenfeld sandpile. When an edge server exceeds a certain capacity threshold, it collapses and initiates an avalanche of migrating tasks in order to balance the workload among edge servers. The achievement results in a significant reduction in task assignment errors and redundancy, which dominant the overhead of energy consumption. On the other hand, You *et al.* [14] considered the transmission energy for multiple IIoT devices offloaded tasks to the edge computing framework. The optimal solution was developed and resolved so as to minimize the weighted sum mobile energy consumption under the constraint on computation latency. In [15], the problem of interdependent task scheduling was addressed within an edge system of deep memory hierarchies to obtain energy efficiency. The intermediate data of the tasks are prioritized and assigned at appropriate levels of cache memory to optimize latency and energy in data access. In order to make the offloading decision, Li *et al.* formulated the problem in a 0-1 nonlinear integer programming with a consideration of channel interference threshold and the time deadline [16]. Based on that assumption, a reverse auction based offloading policy has been proposed to obtain energy efficiency improvement for task execution.

The QoS-aware approaches consider multiple criteria of IIoT service requirements such as execution latency, service availability, transmission throughput, and security as trade-off problems, which have to be optimized. In [17], a distributed optimization algorithm that cooperated among edge servers, called *offload forwarding*, is proposed by using the distributed alternating direction method of multipliers (ADMM) via variable splitting to maximize QoS and energy efficiency. For focusing on maximizing the transmission throughput for task offloading, Vu *et al.* [18] utilize the Hungarian method to optimize the downlink sum-rate in fog computing-enabled networks. On the other hand, Zeng *et al.* [19] proposed a security-aware and budget-aware workflow scheduling strategy (SABA), which considers the task distribution among providers to achieve cost effective and secure services in the context of convergent network. This strategy is applicable to the edge computing environment where edge servers act the role of service providers for offloaded tasks from the IIoT devices. Within the same purpose, Xu *et al.* proposed the green offloading (GO) scheme [20] that uses the reverse auction theory to develop the offloading decision while satisfying the user quality of services (QoS) requirements, bandwidth, and the maximum transmit power.

Although the existing approaches have significantly contributed to improve the performance of the edge computing, most of existing approaches face a drawback in online task handling, where the tasks that arrive at the edge framework have to be queued before the scheduling decision made. This givening generates significant latency, especially in cases of smart manufacturing, where real-time response requirement is a crucial criterion for precise production processes. In the next Section, our proposed PIOTS mechanism, which overcomes these issues, is described in detail.

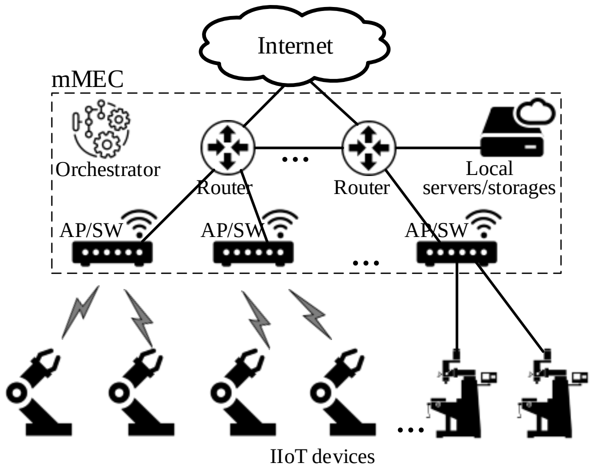

Figure 1: System model where the IIoT devices are supported by the mMEC framework in smart manufacturing environment.

## 3. Pattern Identified Online Task Scheduling

### *3.1. Basic Assumptions*

In this paper, we consider a networking infrastructure for smart manufacturing where mMEC framework is covered. Machinery and auxiliary devices participate in a smart production process generate their IIoT tasks and offload them to the mMEC framework for centralized handling. The IIoT tasks are processed and responded to the IIoT devices on demands. Typically, the production process operates continuously in a long duration (e.g., weeks, months, and even years). Therefore, we assume that the IIoT task patterns maintain their permanent trends even though instant tasks are generated randomly by the IIoT devices. In this circumstance, real-time response is considered as a crucial requirement for precise productions. Figure 1 illustrates the system model where the IIoT devices are supported by the mMEC framework in smart manufacturing environment. The IIoT devices are connected to the network via wireless access points (APs) and switches (SWs). The mMEC framework consists of APs/SWs, routers, and local servers/storages, which have various computing capacities. In a mMEC framework, all these ECEs are managed and scheduled their computations by a central device named orchestrator.

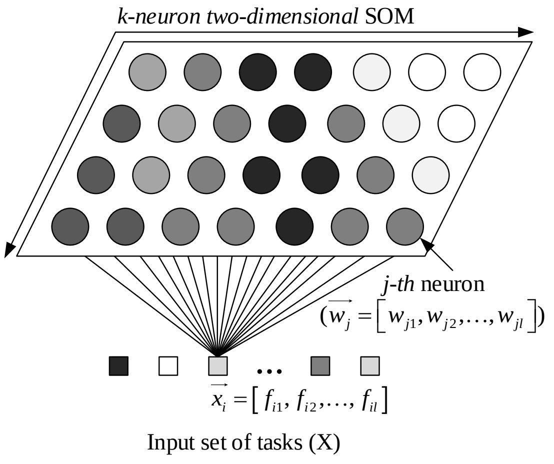
k-neuron two-dimensional SOM
j-th neuron
$(\overrightarrow{w_j} = [w_{j1}, w_{j2}, \ldots, w_{jl}])$
$\overrightarrow{x_i} = [f_{i1}, f_{i2}, \ldots, f_{il}]$
Input set of tasks (X)

Figure 2: Self organizing map (SOM).

### *3.2. Self Organizing Map*

Self organizing map is an artificial neural network (ANN) that utilizes unsupervised learning on the high-dimensional data to produce a defined low-dimensional representation called *map*, see Figure 2. Each SOM is formed by a defined number of neurons, which reflect the map dimension. The operation of the SOMs includes two modes: training and mapping. The training mode develops a map based on the input set of tasks. Meanwhile, the mapping mode is used for classifying new arrived tasks.

The SOM algorithm to train the map is iterated by two steps after Initialization as follows:

*Initialization:* Assume that there are $n$ tasks in the input set $X$, wherein each task $i$ ($1 \leq i \leq n$) is characterized by $l$ features forming a corresponding vector $\overrightarrow{x_i} = [f_{i1}, f_{i2}, \ldots, f_{il}]$. Accordingly, each neuron $j$ in the SOM ($1 \leq j \leq k$) has a $l$-dimensional weight vector $\overrightarrow{w_j}$, where $\overrightarrow{w_j} = \left[w_{j1}, w_{j2}, \ldots, w_{jl}\right]$. To start the SOM algorithm in order to train the map, each neuron initiates its own weight vector with random values. Afterwards, two the steps below are contiguously repeated for each task in the $X$. In detail, for task $i$, the SOM algorithm performs:

- *Step 1 (Matching):* Euclidean distance between task $i$ and each neuron in the SOM is calculated in order to find the best matching unit. The best matching unit for the task $i$ is the neuron $\overrightarrow{w_i^*}$, which has the smallest Euclidean distance to the task $i$. In other words, $\overrightarrow{w_i^*}$ is given by

$$\overrightarrow{w_i^*} = \underset{\forall j, 1 \leq j \leq k}{\arg\min} \sqrt{\sum_{s=1}^{l} (f_{is} - w_{js})^2}. \tag{1}$$

- *Step 2 (Neuron update):* All neurons in the neighborhood of neuron $\overrightarrow{w_i^*}$ update their weight vectors to be closer to task $i$ by

$$\overrightarrow{w_j}(t+1) = \overrightarrow{w_j}(t) + \theta(t) \cdot \mu(t) \cdot (\overrightarrow{x_i} - \overrightarrow{w_j}(t)), \tag{2}$$

where $\overrightarrow{w_j}(t)$ is the weight vector of neuron $j$ at this iteration $t$. $\theta(t)$ is the neighborhood function, which is diminished every iteration. $\theta(t)$ determines the distance from neuron $\overrightarrow{w_i^*}$ to define the neighboring neurons. $\mu(t)$ is the learning factor, which calculates the amount of effect from the input task $i$ to the neuron $j$ based on distance between them [21]. The mathematical expressions of $\theta(t)$ and $\mu(t)$ are described as follows:

$$\theta(t) = \exp\left(-\frac{|\overrightarrow{w_i^*}|^2}{2\sigma^2(t)}\right), \tag{3}$$

$$\mu(t) = \mu_0 \times \exp\left(-\frac{t \lg(R)}{|X|}\right), \tag{4}$$

where $R$ is the maximum radius from the center of the map (i.e., $0.5 \max\{MapWidth, MapHeight\}$) and $\sigma(t)$ is the neighborhood radius that is given by

$$\sigma(t) = R \times \exp\left(-\frac{t^2 \lg(R)}{|X|}\right). \tag{5}$$

### *3.3. Pattern Identified Online Task Scheduling*

In this section, we describe the PIOTS mechanism in detail. The rationale behind the PIOTS includes three steps: (i) identifying the set of typical SOMs for IIoT task patterns, (ii) calculating expected optimal task assignment on the typical SOM prior to each timeslot, and (iii) online assigning new arrived tasks to appropriate ECEs. Steps (i) and (ii) are in offline mode and step (iii) is in online mode.

#### *3.3.1. Offline Mode*

**Typical SOM identification:** In terms of task execution offloading, a IIoT task $i$ is characterized by a four-dimensional feature vector as given by

$$\overrightarrow{x_i} \triangleq [u_i, c_i, r_i, \tau_i], \tag{6}$$

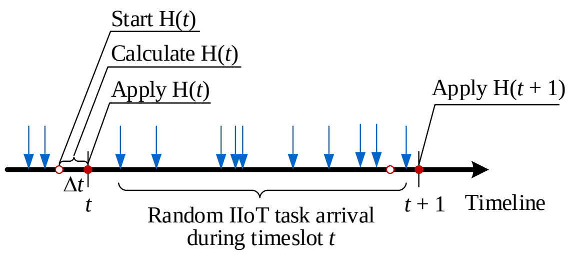

Figure 3: IIoT task arrival and PIOTS operation timeline.

where $u_i, c_i, r_i$, and $\tau_i$ are relative task size, relative average processing complexity, relative response size, and relative execution deadline compared to their minimum and maximum values of all trained tasks, respectively. As aforementioned in Section 3.1, the production process operates continuously during a given period. Within each timeslot in this period, given that there are an average number $N_x$ tasks in the set $\mathcal{X}(t)$ of IIoT tasks offloaded to the mMEC framework, the number $k$ of neurons in the SOM is selected as

$$k = \left\lfloor \frac{1}{\lambda} N_x \right\rceil, \quad \lambda \geq 1, \tag{7}$$

where $\lfloor \cdot \rceil$ is the nearest integer function, and $\lambda$ is a scale level, which supports reduction of the SOM size. Based on these settings, the $k$-neuron two-dimensional SOM is trained using all offloaded IIoT tasks in the given period. For each distinctive period, we obtain a typical trained SOM to represent the offloaded IIoT tasks. A distinctive period is defined as a cycle of the offloaded IIoT task arrival, after this duration the task arrival is repeated again in terms of volume and characteristics.

**Expected optimal task assignment calculation:** Figure 3 describes the PIOTS operation timeline in a random IIoT task arrival context. Prior to timeslot $t$, the PIOTS mechanism performs the H($t$) function at $t - \Delta t$ in order to determine the expected optimal task assignment during timeslot $t$. The H($t$) function uses the Hungarian method [22] to address the expected optimal task assignment problem between the neuron set of the typical SOM in timeslot $t$ and the current ECE set. The result of H(t) function will be applied for online task assignment in the whole duration from timeslot $t$ to timeslot $t + 1$ even though the H($t+1$) function will start at $(t+1) - \Delta t$.

The expected optimal task assignment problem is defined as *"Given the current performance status (i.e., task buffer and CPU frequency) of ECEs and the transmission status (i.e., access data rate between IIoT devices and the mMEC framework and forwarding data rate among ECEs), minimize the total latency of task execution when assigning the neuron set of the typical SOM to the ECEs"*.

The expected optimal task assignment problem can be formulated as follows. Let $\mathcal{W}$ and $\mathcal{M}$ denote the set of $k$ neurons in the SOM and the set of $m$ ECEs in the mMEC framework, respectively. Since the SOM neurons have trained by using the historical tasks, the weight vector of the neuron reflects the average values of the task features, accordingly. That is, $w_{j1}, w_{j2}, w_{j3}$, and $w_{j4}$ in the weight vector $\overrightarrow{w_j}$ of neuron $j$ reflect the average values of the relative task size, relative average processing complexity, relative response size, and relative execution deadline, respectively. Let the current buffer size of tasks waiting for processing in ECE $s$ be $b_s$ CPU cycles. Hence, the total latency $L_{js}$ when neuron $j$ is assigned to ECE $s$ is as follows.

$$L_{js} = \underbrace{\frac{w_{j1} + w_{j3}}{ra_{js}}}_{(a)} + \underbrace{\frac{b_s}{C_s}}_{(b)} + \underbrace{\frac{w_{j1} w_{j2}}{C_s}}_{(c)} + \underbrace{\frac{w_{j1} + w_{j3}}{rb_{js}}}_{(d)}, \tag{8}$$

where $C_s$, $ra_{js}$, and $rb_{js}$ are the CPU frequency of ECE $s$ in Hz, the access data rate from the IIoT device that owned task $j$ to the network in bps, and the internal forwarding rate to the ECE $s$ in bps, respectively. Equation (8) consists of (a) uploading latency, (b) queuing latency, (c) task processing latency, and (d) response latency. The access data rate $ra_{js}$ is given by

$$ra_{js} = \mathrm{BW}_{js} \log_2(1 + \mathrm{SINR}_j), \tag{9}$$

where $\mathrm{BW}_{js}$ and $\mathrm{SINR}_j$ are the bandwidth allocated for the IIoT device and the channel quality between the IIoT device and the network, respectively. Based on that, the expected optimal task assignment problem ($\mathcal{F}$) can be described by

$$(\mathcal{F}) \quad \text{minimize}: \sum_{j=1}^{k} \sum_{s=1}^{m} z_{js} L_{js} \tag{10}$$

$$\text{s.t. } z_{js} \in \{0, 1\}, \tag{11}$$

$$L_{js} \leq w_{j4}, \tag{12}$$

$$\sum_{j=1}^{k} z_{js} = 1, \ \forall s = 1, 2, \cdots, m,, \tag{13}$$

where the indicator $z_{js}$ is given by

$$z_{js} \triangleq \begin{cases} 1 & \text{if neuron } j \text{ is assigned to ECE } s \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Constraint (13) ensures that a neuron could only be assigned to one ECE and the maximum matchings are established for all $m$ neurons of the SOM.

**Algorithm 1** Pseudocode of function H(·).

```
1: Generate a square latency matrix $V$ from $\mathcal{W}$ and $\mathcal{M}$.
2: In each row, subtract all entries by the smallest row entry.
3: In each column, subtract all entries by the smallest column entry.
4: Find the minimum number $\upsilon$ of rows and columns that cover all
   zero entries.
5: If $\upsilon == |V|$, the optimal solution is found.
6: If $\upsilon < |V|$,
       Find the smallest entry $\gamma$ in the matrix,
       Subtract all uncovered rows by $\gamma$,
       Add all covered columns by $\gamma$,
       Return to Line (4).
```

Table 1: Simulation parameters

| Parameter | Value |
|---|---|
| Orchestrator configuration | CPU: Intel Core i5-6400 2.7GHz; RAM: 16GB |
| Number of ECEs | 20 |
| CPU frequency of ECEs | $\{1.5, 2.5, 5.0, 10.0, 25.0\}$ GHz |
| Number of IIoT devices | 300 |
| Task processing complexity | $\{10, 50, 100, 500, 1000\}$ cycles/bit |
| Execution deadline | 0.5 – 1.5 s |
| Timeslot duration | 0.1 s |
| Simulation duration | 300 timeslots |

It is observed that task assignment problem $\mathcal{F}$ regards a bipartite graph of two sets $\mathcal{W}$ and $\mathcal{M}$. We apply the Hungarian method (referred as H(·)) on the graph to achieve the optimal solution. Function H(·) is performed as follows (see Algorithm 1):

- Augment the latency matrix of all possible task assignments from neuron set $\mathcal{W}$ to ECE set $\mathcal{M}$ into a square matrix (named $V$) with dimension $\max(|\mathcal{W}|, |\mathcal{M}|)$ by supplementing additional entries of constant number (e.g., 0).
- In each row of $V$, subtract the smallest row entry from all of the row entries. Similarly, in each column of $V$, subtract the smallest column entry from all of the column entries.
- Find the minimum number $\upsilon$ of rows and columns by which all zero entries are covered.
- If $\upsilon$ is equal to the size of $V$, the optimal solution is found. Pick up a set of 0 entries satisfying in which no more than two 0 entries are in the same row or column. Otherwise, if $\upsilon$ is less than the size of $V$, determine the smallest entry $\gamma$ that is uncovered in the previous step. Afterwards, subtract $\gamma$ from all uncovered rows and add $\gamma$ to all covered columns. Return to the previous step to find the minimum number $\upsilon$ again.

For each timeslot, function H(·) is calculated prior $\Delta t$ before the beginning of the timeslot in order to find the expected optimal task assignment solution.

### *3.3.2. Online Mode*

**Online task assignment:** In online mode, when an IIoT task arrives at the mMEC framework, the task is matched to the SOM so as to seek the best matching neuron, using Equation 1 with the computational complexity of $O(1)$. According to the optimal assignment for the found neuron that was determined by function H(·) in the offline mode, this assignment is performed on the arrived IIoT task immediately.

## 4. Evaluation Preparations

### *4.1. System Settings*

In order to evaluate the performance of the PIOTS scheme, we considered a network model, where the mMEC framework consists of 20 ECEs equipped with various CPU frequencies of {1.5, 2.5, 5.0, 10.0, 25.0} GHz. Total number of IIoT devices associated to the network is 300. The task processing complexity is determined as {10, 50, 100, 500, 1000} cycles/bit derived from the practical analysis presented in [23]. Execution deadline is randomly set within (0.5, 1.5) s. Lastly, timeslot duration is set to 0.1 s. These simulation parameters are summarized in Table 1.

In terms of task and response size settings, we derived these parameters from a part of the *CAIDA anonymized Internet traces* data set [24]. The CAIDA dataset contains anonymized passive traffic traces from the *equinix-chicago* Internet data collection monitor located at an Equinix datacenter in Chicago, IL, on high-speed Internet backbone links. Table 2 shows statistical indexes of the task samples used for SOM training and system evaluation. The tasks are selected for three types of traffic as follows:

- *Environmental sensor data:* The information of environmental conditions is reported to central applications in the mMEC framework during each fixed period. The task size is configured within constant and small dimension, and these tasks do not require response from the applications.

Table 2: Statistical indexes of task samples for SOM training and system evaluation.

| Index | Value |
|---|---|
| Number of tasks | 120000 |
| Min, max, and average values of task sizes | $\{129, 512, 316.45\}$ Kb |
| Min, max, and average values of response sizes | $\{0, 32, 15.47\}$ Kb |

- *Video surveillance data:* Live streaming data from monitoring cameras is delivered to surveillance applications in the mMEC framework for video analysis and storage. These tasks are packetized in fixed and large size. The response data might be issued from the applications if there are any alarms that should be announced.

- *Production control data:* These data are generated by the machinery during precise production processes. The tasks are regularly offloaded to the mMEC framework within a determined size. Tasks require responses from the applications to exactly handle production work.

### *4.2. Evaluation Methodology*

Within the aforementioned system settings, our proposed PIOTS scheme has been compared to the offline Hungarian task assignment (OHTA) algorithm and the online greedy task assignment (OGTA) algorithm. Initially 30,000 samples of the selected tasks derived from the CAIDA data set are used to train the SOM in the PIOTS scheme. Then, 90,000 other samples are used for evaluation; see Table 2. The operations of these schemes are described as follows:

- The PIOTS scheme is performed as shown in Section 3.3.

- The OHTA algorithm gathers all arrived tasks at the input buffer of the mMEC framework during one timeslot. At the end of each timeslot, Hungarian method is utilized to decide the optimal task assignment for all tasks in this timeslot [22].

- The OGTA algorithm determines an ECE, which provides the lowest latency for task execution following equation 8, for the arrived task [25].

The simulation results are logged in terms of task execution latency and execution error rate. The execution error rate evaluates ratio of the over-deadline task executions and the total offloaded tasks.

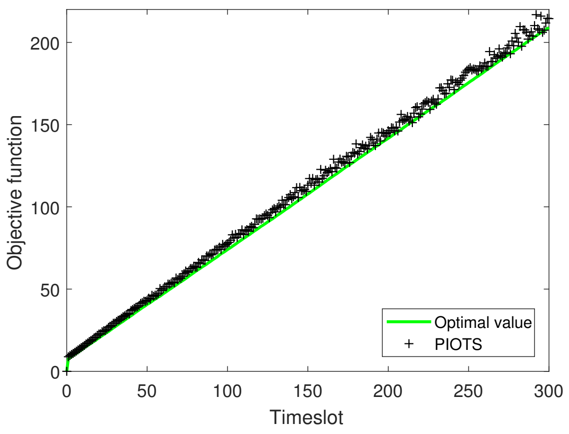

Figure 4: PIOTS scheme vs. Optimal solution for objective function $\mathcal{F}$.

## 5. Numerical Result Analysis

First, the proposed PIOTS scheme is compared to the optimal solution of the objective function $\mathcal{F}$. The optimal solution for the integer programming problem $\mathcal{F}$ used in this paper is a combination of Hungarian method and *backpressure* algorithm [10]. Figure 4 illustrates outcomes of $\mathcal{F}$ depending on various IIoT task arrivals through 300 timeslots. Numerical results reveal that the PIOTS scheme provides an approximate performance compared to the optimal solution. The differential in average task processing latency is 0.159 ms. It is worth noting that the PIOTS scheme performs the expected task assignment based on the typical task set derived from SOM map; then, it does real task assignment online. Moreover, the SOM map is trained by using the collected IIoT tasks arrived at the network in the past. Therefore, within a sufficient number of neurons in the SOM map that well represents for the typical characteristics of the incoming IIoT tasks, the mMEC framework can classify the incoming tasks immediately and then it handles the tasks by approximately optimal assignment. That is, each incoming task is immediately assigned to appropriate ECE right after the task arrives at the network. Meanwhile, the optimal solution is calculated based on the set of gathered IIoT tasks in each timeslot.

Table 3 shows statistical indexes of time consumption in ms for task assignment decision of the orchestrator when applying the PIOTS, OHTA, and OGTA schemes, respectively. The average decision making duration that the PIOTS scheme consumes is 0.038 ms, which is smaller than the OHTA scheme's 100.1027 ms

Table 3: Time consumption in ms for task assignment decision of the orchestrator.

| Scheme | Mean | Std. deviation | Std. error mean |
|---|---|---|---|
| PIOTS | 0.0380 | 0.00242 | 0.00011 |
| OHTA | 100.1407 | 0.00265 | 0.00012 |
| OGTA | 0.0025 | 0.00111 | 0.00005 |

(approximate 2635 times of reduction) and greater than the OGTA scheme's 0.0355 ms (approximate 15 times of increase). The reason is because the PIOTS scheme calculates Equation 1 for all neurons in the SOM map, while the OHTA scheme must wait until the end of each timeslot to calculate the optimal solutions. The OGTA scheme achieves the smallest duration since it performs Equation 8 for only 20 ECEs and selects the smallest latency value. Although there is a differentiate between the PIOTS and OGTA schemes, the average values are considered to insignificantly affect the task processing latency. The standard deviation and standard error mean show that all schemes are performed in stable condition.

Figure 5 demonstrates a comparison of the average task processing latency when applying three competitors (PIOTS, OHTA, and OGTA). The PIOTS scheme achieves an effective performance since the typical task set is used to pre-determine expected task assignment for incoming tasks. Meanwhile, the OHTA scheme performs task assignment based on the gathered incoming tasks during each timeslot. Although the OHTA scheme provides better adaptation to varying task arrivals, it must wait until the end of each timeslot to collect the tasks and then determine optimal assignment. On the other hand, the OGTA scheme greedily assigns tasks to the ECEs of lowest latency. In statistic perspective, for average take processing latency, the PIOTS scheme overcomes the OHTA scheme and the OGTA scheme by 41.47% and 4.47%, respectively.

In order to evaluate the service capability of the network, we utilize the execution error rate, which is defined by the percentage of deadline-violated IIoT devices in the total associated devices in the network. Figure 6 shows simulation results corresponding to three thresholds of execution deadline including 0.5 s, 1.0 s, and 1.5 s as aforementioned in the system settings; see Table 1. During 300 simulated timeslots, the execution error rates for 0.5-second deadline are approximate among PIOTS, OHTA, and OGTA schemes (0.970%, 0.976%, and 0.977%, respectively). Meanwhile, in term of 1.5-second deadline, the PIOTS scheme decreases the execution error rate to 0.826% (approximate 5.7% and 1.2% decreases compared to the OGTA and OHTA

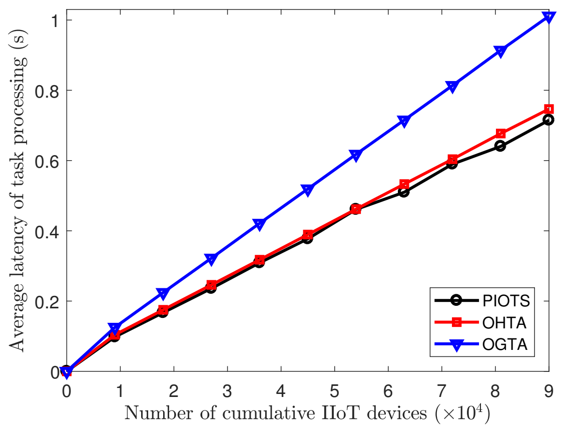

Figure 5: Average latency of task processing.

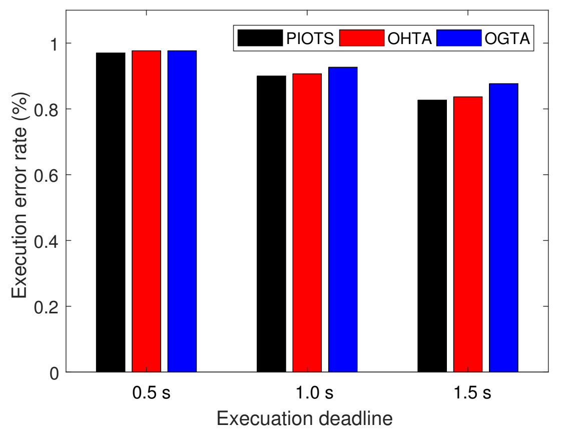

Figure 6: Percentage of deadline-violated IIoT devices.

schemes, respectively). This analysis demonstrates that the PIOTS scheme provides better balanced task distribution among ECEs in order to satisfy task execution deadlines.

Figure 7 plots average buffering latency for IIoT tasks arrived at the ECEs during 100, 200, and 300 timeslots. Since the arrived task volume has been configured to over-capacitate the ECE performances leading to a saturated condition in the network, buffering latency increases by timeslot. It is observed that the OGTA scheme makes significant buffering latency in every timeslots. On the other hand, the PIOTS scheme achieves the lowest buffering latency by 44.33% and 4.32% in timeslot 300, compared to the OGTA scheme and OHTA scheme, respectively. Figure 8 depicts the reason for that. The y-axis represents the assigned task

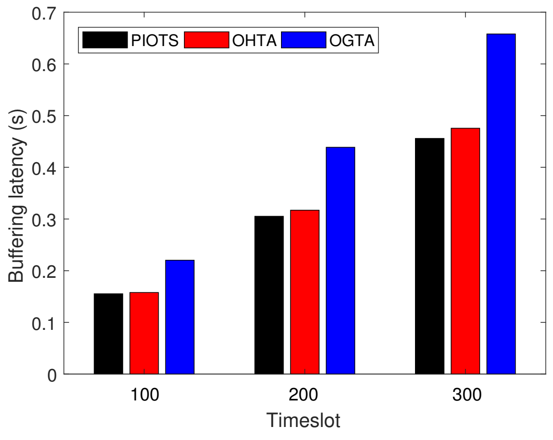

Figure 7: Average buffering latency for IIoT tasks arrived at the ECEs.

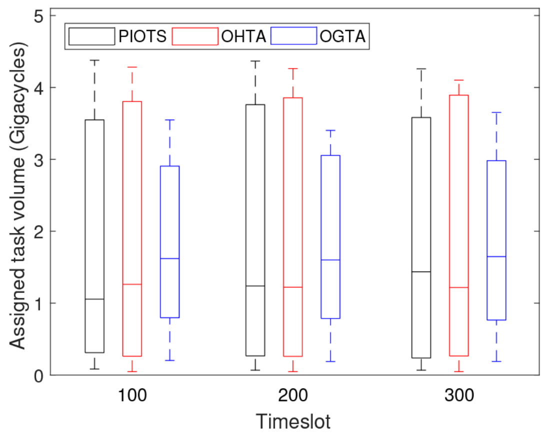

Figure 8: Task distribution among ECEs.

volume in Gigacycles (a.k.a. computational CPU cycles) among ECEs, which is given by a production function of task size and task processing complexity for all assigned tasks in each ECE. In the OGTA scheme, the arrived tasks are assigned more equally among ECEs (represented by the width of the box plotted) in comparison with other schemes. Since the PIOTS scheme and OHTA scheme are able to adapt task distribution according to the diversity of ECE computation performances, they provide better task assignments resulting in lower buffering latency.

## 6. Concluding Remarks

In this paper, a pattern-identified online task scheduling mechanism has been proposed to deliberate on real-time task assignment in the smart manufacturing system. The proposed PIOTS scheme utilizes SOM technology for task identification and then assigns the task to appropriate ECE by using the Hungarian method. Simulation results demonstrate that the PIOTS scheme overcomes the existing algorithms in terms of task processing latency and service capability for satisfying IIoT applications. In future research, individual requirements of IIoT applications will be considered and verified via several popular datasets within the purpose of achieving the optimal performance for task handling in the entire network. Moreover, a consideration of applying game-theoretic approach should be studied to develop a distributed computational mMEC framework.

## Disclosure

Part of this work was presented at ICTC conference in Jeju, Korea, Oct. 18–20, 2017.

## Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

## Acknowledgements

This research was supported by the Chung-Ang University Young Scientist Scholarship (CAYSS) Program and National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1A2B4009802).

## References

[1] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari, and R. Ranjan, "Internet of things and edge cloud computing roadmap for manufacturing," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 66–73, 2016.

[2] N.-N. Dao, Y. Kim, S. Jeong, M. Park, and S. Cho, "Achievable multi-security levels for lightweight IoT-enabled devices in infrastructureless peer-aware communications," *IEEE Access*, vol. 5, pp. 26 743–26 753, 2017.

[3] ETSI, "Multi-access edge computing," (cited Nov. 2, 2017). [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing

[4] N.-N. Dao, Y. Lee, S. Cho, E. Kim, K.-S. Chung, and C. Keum, "Multi-tier multi-access edge computing: The role for the fourth industrial revolution," in *IEEE International Conference on ICT Convergence (ICTC)*, Jeju, Korea, Oct. 18–20, 2017, pp. 1280–1282.

[5] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[6] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, 2016.

[7] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[8] A. C. Baktir, A. Ozgovde, and C. Ersoy, “How can edge computing benefit from software-defined networking: A survey, use cases, and future directions,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[9] S. Peng, J. O. Fajardo, P. S. Khodashenas, B. Blanco, F. Liberal, C. Ruiz, C. Turyagyenda, M. Wilson, and S. Vadgama, “QoE-oriented mobile edge service management leveraging SDN and NFV,” *Mobile Information Systems*, vol. 2017, p. 3961689, 2017.

[10] N.-N. Dao, J. Lee, D.-N. Vu, J. Paek, J. Kim, S. Cho, K.-S. Chung, and C. Keum, “Adaptive resource balancing for serviceability maximization in fog radio access networks,” *IEEE Access*, vol. 5, pp. 14 548–14 559, 2017.

[11] Y. Mao, J. Zhang, and K. B. Letaief, “Dynamic computation offloading for mobile-edge computing with energy harvesting devices,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[12] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, “Delay-optimal computation task scheduling for mobile-edge computing systems,” in *IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 1451–1455.

[13] J. L. J. Laredo, F. Guinand, D. Olivier, and P. Bouvry, “Load balancing at the edge of chaos: how self-organized criticality can lead to energy-efficient computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 517–529, 2017.

[14] C. You, K. Huang, H. Chae, and B.-H. Kim, “Energy-efficient resource allocation for mobile-edge computation offloading,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[15] T. Maqsood, N. Tziritas, T. Loukopoulos, S. A. Madani, S. Khan, and C.-Z. Xu, “Leveraging on deep memory hierarchies to minimize energy consumption and data access latency on single-chip cloud computers,” *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 154–166, 2017.

[16] L. Li, X. Zhang, K. Liu, F. Jiang, and J. Peng, “An energy aware task offloading mechanism in multi-user mobile-edge cloud computing,” *Mobile Information Systems*, vol. PP, pp. 1–1, 2017.

[17] Y. Xiao and M. Krunz, “QoE and power efficiency tradeoff for fog computing networks with fog node cooperation,” in *IEEE Conference on Computer Communications (INFOCOM)*, Atlanta, GA, USA, May 1–4, 2017, pp. 1–9.

[18] D.-N. Vu, N.-N. Dao, and S. Cho, “Downlink sum-rate optimization leveraging Hungarian method in fog radio access networks,” in *IEEE International Conference on Information Networking (ICOIN)*, Chiang Mai, Thailand, Jan. 10–12, 2018, pp. 1–1.

[19] L. Zeng, B. Veeravalli, and X. Li, “SABA: A security-aware and budget-aware workflow scheduling strategy in clouds,” *Journal of Parallel and Distributed Computing*, vol. 75, pp. 141–151, 2015.

[20] X. Xu, C. Yuan, J. Li, H. Zhang, and X. Tao, “Reverse auction based green offloading scheme for small cell heterogeneous networks,” *Mobile Information Systems*, vol. 2016, p. 5087525, 2016.

[21] T. V. Phan, N. K. Bao, and M. Park, “Distributed-SOM: A novel performance bottleneck handler for large-sized software-defined networks under flooding attacks,” *Journal of Network and Computer Applications*, vol. 91, pp. 14–25, 2017.

[22] D. Jungnickel, *Graphs, Networks and Algorithms*, 4th ed., ser. Algorithms and Computation in Mathematics 5. Springer Berlin Heidelberg, 2013, ch. 14. Weighted matchings.

[23] A. P. Miettinen and J. K. Nurminen, “Energy efficiency of mobile clients in cloud computing,” in *USENIX Conference on Hot Topics in Cloud Computing*, Boston, MA, USA, Jun. 22-25, 2010, pp. 1–7.

[24] C. Walsworth, E. Aben, K. Claffy, and D. Andersen, “The CAIDA UCSD anonymized Internet traces 2015,” (cited Nov. 14, 2017). [Online]. Available: http://www.caida.org/data/passive/passive_2015_dataset.xml

[25] H. To, L. Fan, L. Tran, and C. Shahabi, “Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints,” in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Sydney, Australia, Mar. 14-18, 2016, pp. 1–8.