Dynamic Collaborative Task Offloading for Delay Minimization in the Heterogeneous Fog Computing Systems

Hoa Tran-Dang and Dong-Seong Kim

***Abstract*—Fog computing systems have been widely integrated in IoT-based applications to improve quality of services (QoS) such as low response service delays. This improvement is enabled by task offloading schemes, which perform task computation near the task generation sources (i.e., IoT devices) on behalf of remote cloud servers. However, reducing delay remains challenging for offloading strategies owing to the resource limitations of fog devices. In addition, a high rate of task requests combined with heavy tasks (i.e., large task size) may cause a high imbalance of the workload distribution among the heterogeneous fog devices, which severely impacts the offloading performance in terms of delay. To address these issues, this paper proposes a dynamic collaborative task offloading (DCTO) approach, which is based on the resource states of fog devices, to dynamically derive the task offloading policy. Accordingly, a task can be executed by either a single fog or multiple fog devices through the parallel computation of subtasks to reduce the task execution delay. Through extensive simulation analysis, the proposed offloading solution showed potential advantages in reducing the average delay significantly in systems with a high rate of service requests and heterogeneous fog environment compared with the existing solutions. In addition, the proposed scheme can be implemented online owing to its low computational complexity compared with the algorithms proposed in related works.**

***Index Terms*—Data task fragmentation, fog computing systems, parallel communication and computation, task offloading.**

I. INTRODUCTION

THE Internet of things (IoT) has become an integral element for realizing smart systems, including smart cities [1], smart grids [2], smart factories [3], smart logistics and supply chains [4], [5]. The fundamental aspect of IoT is to connect all devices through the Internet protocol to exchange high volume data and process them to create smart services, and applications [6], [7]. Owing to limited computation resources, network, storage, and energy, IoT devices are inadequate for executing the computational tasks, especially tasks with huge volumes and complex data structures. Cloud computing is an essential solution to this problem because it provides powerful resources to fulfill tasks efficiently [8], [9]. However, cloud computing-based solutions do not always meet the expected quality of service (QoS) for delay-sensitive applications because of the long physical distance between the IoT devices and the remote cloud servers, scarce spectrum resources, and intermittent network connectivity.

This has led to the emergence of fog computing, which extends the cloud computing resources (i.e., computing, storage, and networking) closer to the data generation sources (i.e., IoT devices), thereby allowing for the prescribed QoS requirements of services and applications to be met by enabling the fog computing devices (e.g., switches, gateways, and hubs) to process and offload most tasks on behalf of the cloud servers in a distributed manner [10], [11]. Consequently, fog computing systems (FCSs) consisting of connected fog computing devices have become essential in IoT-based systems to support uninterrupted services and applications with low response delay along the things-to-cloud continuum [12].

To realize this benefit of fog computing, FCSs require efficient resource allocation strategies to perform task offloading [13]. However, there are many factors that challenge the delay-reduction objective of offloading algorithms. First, an FCS consists of heterogeneous computing devices with different storage capacity, computation, and networking characteristics. In addition, some fog devices support the processing of only one data type such as image, text, video, or audio [14]. In addition, applications such as artificial intelligence (AI) and machine learning (ML) algorithms require the computation of complex tasks, which typically include multiple types of input data. Second, task size also has a significant impact on offloading performance in the FCS. For example, some fog devices are unable to process the entire data of heavy tasks owing to a lack of storage and/or computational capacity. Consequently, more tasks are likely to be queued in more powerful resource fogs, causing long waiting times in the queues. Third, the request rate directly impacts the queuing state of the fogs. Therefore, without an efficient resource allocation policy, a high rate of task requests may lead to a high workload imbalance among the fog devices, as the fog nodes with powerful computing resources may receive more task requests. As a result, the requirements of latency-sensitive applications can be violated because of the excessive waiting time of long task queues.

Manuscript received October 28, 2022 revised February 2, 2023; approved for publication by Hyoil Kim, Division 3 Editor, February 15, 2023.

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2023-2020-0-01612) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation)", Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2018R1A6A1A03024003) and Korea Research Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2020R1I1A1A01073019).

The authors are with department of IT Convergence Engineering, Kumoh National Institute of Technology, Korea, emails: {hoa.tran-dang, dskim}@kumoh.ac.kr.

D.-S. Kim is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2023.000008

These typical issues necessitate the development of a dynamic approach based on the fog resource states to make the best context-aware offloading decisions. One possible solution to overcome these issues is to divide the tasks into subtasks, which can then be executed in parallel by the different limited-resource fog nodes to reduce the overall processing delay. In addition, as parallel computation is enabled for the task execution, the task division may also have the potentials to balance the workload in heterogeneous fog devices because fog devices with limited available resource can process subtasks instead of entire task. Practically, this advantageous feature of the divide and conquer concept has been widely adopted and adopted in parallel computing [15], grid computing [16], and service providing systems [17]. However, a few existing works used the technique in the context of fog computing, to explore and exploit the potential advantages of parallel computation, thereby improving the performance of the FCS in terms of task execution delay.

In theses regards, this paper proposes a dynamic collaborative task offloading (DCTO) scheme that can make the offloading decision dynamically and efficiently based on the available resources of fog computing devices. Summarily, this paper provides five key contribution as follows:

- We investigate the model of fog computing that can be applied to in some practical IoT systems to provide specific services and applications.
- Based on the properties of service requests (i.e., computation tasks) and the resource states of fog nodes, the full and partial offloading models are dynamically applied to utilize the available resources of fog computing devices efficiently.
- An optimal scheduling to schedule the data communication and data execution of subtasks is derived for each single fog node that significantly contributed to achieve the objective of delay reduction.
- A DCTO scheme is achieved as an optimal solution of an optimization problem that incorporates the dynamic task offloading and optimal scheduling of data communication and data execution of subtasks.
- Extensive simulations and comparative analysis are conducted to demonstrate and evaluate the performance of DCTO.

The remainder of this paper is structured as follows. Section II highlights the related works. Section III presents the proposed system model. Section IV establishes a mathematical optimization problem to minimize the task execution delay. Section V presents the simulation results and a comparative analysis with the performance of related works. Finally, Section VI concludes the paper and introduces future work.

II. Related Works

A number of offloading algorithms for minimizing task execution delay have been proposed in literature. However, only a few of them analyze the impact of multiple factors simultaneously on the performance of offloading policies such as the task request rate, dynamic task division, and queuing states of fog devices.

In most task offloading solutions, offloading multiple tasks of fog nodes (FNs) to multiple neighbor FNs (i.e., helper nodes (HNs)) is modeled as a multitask multihelper (MTMH) problem, which aims to allocate the fog computing resources for processing tasks to minimize the average delay of task execution. This problem can be formulated in the form of multi-objective optimization to examine the trade-off of performance in terms of energy consumption, delay, and execution cost [18]–[20]. Likewise, in [21], a paired offloading strategy of multiple tasks (POMT) in heterogeneous fog networks was modeled as a strategic game between the FNs and HNs to decide which tasks are offloaded by which HNs to minimize the average task execution delay. Game theory has been applied and developed to obtain the Nash equilibrium (NE) point for the POMT, at which the FCS can achieve near-optimal performance in terms of average delay. The work [22] assessed the role of the fog layer in reducing the service response delay in IoT-fog-cloud systems. An intensive analytical model was derived to examine the impact of communication delays between fog devices, queuing delay, and task sizes on the service response delay. In particular, fog-to-fog communication channels with good conditions were exploited in the proposed task offloading mechanisms to contribute to the overall delay reduction. According to the analytic model, all tasks should be processed by both the fog and cloud to reduce the delay and adaptively balance the workload at the computing fog nodes. However, fog resources were not fully used in these works because the majority of heavy tasks need to be processed by the cloud servers. In addition, none of the aforementioned task offloading approaches consider the division of heavy tasks, which can further reduce the delay through the parallel processing of subtasks while increasing the utilization ratio of fog nodes and reducing the cost of cloud usage.

Regarding task division, most existing works only considered dividing each task into two subtasks. For example, according to the FEMTO model in [23], each task is divided into two subtasks with different data sizes, which are then processed by the IoT node and offloaded to the fog entity. An optimization problem is associated with minimizing the energy consumption of task offloading while achieving a fair workload among the fog nodes and satisfying the deadline constraints of tasks. Similarly, in the task offloading methods introduced in [24], [25], each fog divides the tasks into only two parts, which are processed locally by itself and one HN in parallel. Dividing a task into multiple subtasks was first applied in [26] to exploit the parallel computation of subtasks, thereby reducing the task execution delay. Based on the MTMH problem, a game theory-based algorithm called POST was developed to find the optimal matching pair of subtasks and fog nodes to achieve the delay minimization objective. Accordingly, the number of subtasks for each task was dynamically optimized depending on the number of idle HNs in each time slot and the resource contention update of fogs having tasks in queues. In particular, POST can reach the generalized NE (GNE) fixed-point, thus achieving near-optimal performance in terms of average task execution delay. However, there is a lack of investigation regarding the impact of the queuing status of fog nodes, scheduling of subtasks, and task arrival rate on system performance.

To efficiently deal with the dynamic nature of fog computing environment, distributed algorithms using matching theory are developed in the literature [27]. In this class of algorithms, the task offloading problem is translated into the matching game between the task set and HN set. The objective is to achieve the stable matching outcome which matches tasks with HNs appropriately. To deal with dynamic task division, a recent DISCO algorithm introduced in [28] proposes a matching based distributed computation scheme for the fog computing network in which the theory of matching with group firstly is applied. Accordingly, a task divided into multiple subtasks is preferred to be processed by a group of HNs to minimize its execution delay [29].

The work [30] introduced an online algorithm for task offloading in the IoT-fog cloud systems, which applies parallel communication and computation at multiple computing nodes (i.e., fog and cloud servers) to minimize latency. Considering the scenarios with varying task arrival rates, the queuing delays of fog nodes are analyzed to optimize the number of tasks offloaded to other computing nodes to achieve the objective. However, the task transmission and computation at the computing nodes are performed according to the first come and first serve (FCFS) scheduling policy rather than the optimal task scheduling policy. In addition, task division is not considered to balance the workload of heterogeneous fog computing nodes. Recently, a recursive computation offloading algorithm (RCO) was developed in [31] to jointly perform task offloading and task scheduling for fog-radio access networks (Fog-RANs). Considering the execution of tasks residing in a mobile device (MD), the tasks can be offloaded by edge or cloud tiers. The optimization of task scheduling is associated with the RCO to contribute to achieving the delay reduction objective. However, queuing delay and task division have not yet been investigated thoroughly. In [32], the authors proposed an optimization framework to offload the splittable tasks to multiple HNs. The tasks can be divided into subtasks dynamically based on the fog resource states and task properties. The optimization incorporates the optimal scheduling to allocate the subtasks to HNs appropriately such as the overall delay of task execution is minimized.

The advanced techniques such as reinforcement learning (RL) are deployed in many studies to achieve the efficient computational offloading operations in the fog computing environment [33], [34]. By using the online learning methods, the proposed RL-based task offloading solutions are able to cope with the dynamics and complex nature of fog computing, thus offering the efficient offloading policies.

III. System Model and Problem Formulation

A. System Model

This study considered an FCS, as illustrated in Fig. 1, which comprises $N+1$ fog devices denoted as F_0, F_1, $\cdots$, and F_N. F_0 serves as a fog controller (e.g., gateway) that can process and distribute tasks to its neighbors, while the other fogs are HNs, which can connect to F_0 directly and have available computation resources to process the tasks assigned and scheduled by F_0. In addition, the FCS can be connected to a cloud for further purposes such as computation, storage, and caching. However, this study aimed to investigate the performance of the FCS when all the tasks are processed only by fog devices without the need of cloud servers. In this context, F_0 involves two main tasks: data transmission (i.e., transmitting the input data of tasks to the intended offloadees) and data processing. The main role of HNs is to process the tasks in queues according to the FCFS policy.

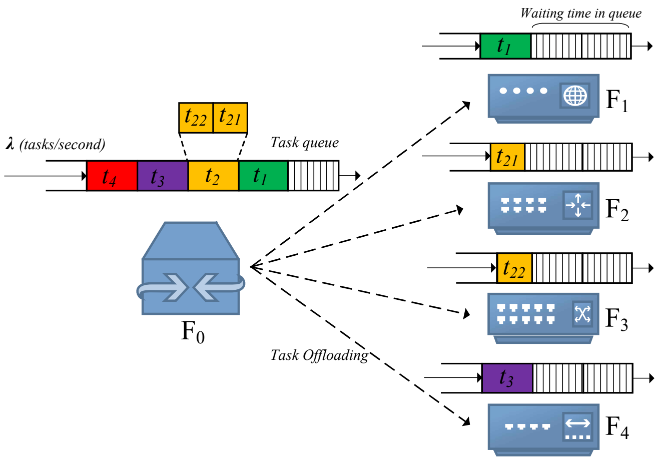

Fig. 1. The proposed DCTO model for the FCS, in which some tasks (e.g., task t_2) can be divided into multiple subtasks (e.g., t_{21}, and t_{22}) to reduce the overall task execution delay through parallel computing.

TABLE I
Resource table of neighbors of fog node F_0.

Fog node	Fog specification & resource status			
	M (MB)	f (GHz)	γ (cycles/bit)	W (s)
F_1	128	5	500	1.2
F_2	256	10	600	2.15
F_3	128	15	750	0.55
F_4	256	10	1000	2.39

This FCS model can be deployed in many practical IoT systems such as smart cities, smart factory, and smart warehouse management systems. Generally, these systems include multiple domains for providing IoT services and applications and each FCS is designed to serve a specific set of services in a single domain such as smart parking services, smart inventory monitoring in the warehouses.

To make an efficient offloading policy, F_0 is based on a table of neighbor resources that contains updated information about the available resources of HNs. We assumed that fog devices of the FCS use a common protocol to communicate and update their resource status periodically [35]. We also assumed that during the offloading decision, the resource states of the HNs are unchanged. Table I shows an example of neighbor resource table stored by F_0, which records the resource states of neighbors with respect to the memory capacity of queue buffer (M), CPU frequency (f), CPU processing density (γ), and expected waiting time in the queue (W).

TABLE II
SUMAMRY OF KEY NOTATIONS.

Notation	Description
t_i	Task i
a_i	Size of T_i (bits)
t_{ik}	Subtask divided from t_i
a_{ik}	Size of t_{ik} (bits)
T_i^a	Arrival time of t_i at F_0
T_i^t	Start time to transmit t_i to an offloadee by F_0
T_i^p	Start time to process t_i by offloadee
T_i^f	Finish time of processing t_i by offloadee
D_i, D_{ik}	Total delays for executing t_i, and t_{ik}
W_i	Expected waiting time in queue of F_i
M_i	Memory capacity of queue buffer of F_i
λ	Arrival rate of tasks at F_0 queue (tasks/s)
F_i	Fog node i
r_{ik}	Data transmission rate from F_i to F_k
f_i	CPU frequency of F_i (cycles/s)
γ_i	CPU processing density of F_i (cycles/bit)

B. Task Model

Computation tasks arriving at the queue of F_0 follow an exponential distribution with an average rate of λ tasks per second (tasks/s). At a given time, there is a set $\mathcal{T} = \{t_1, t_2, \cdots, t_K\}$, including K computational tasks, which reside in the queue of F_0. A single task t_i can be processed by either F_0 or F_k ($k = 1, \cdots, N$). In addition, a task t_i with a size a_i can be divided into $m = N + 1$ subtasks (i.e., t_{ik}, $k = 0, \cdots, N$), which can be processed in parallel by different fog nodes. As illustrated in Fig. 1, two subtasks, t_{21} and t_{22} divided from t_2 can be computed simultaneously by F_2 and F_3, respectively. When $m = 1$, the entire task is processed by a single fog. We define a_{ik} as the data size of subtask t_{ik}; thus we have

$$\sum_{k=0}^{N} a_{ik} = a_i, \forall t_i \in T. \tag{1}$$

With this definition, $a_{ik}^n > 0$ if t_{ik} is computed by F_k, and $\alpha_{ik}^n = 0$ if F_k is not involved in processing t_i.

C. Problem Formulation

This paper aims to develop an online approach for F_0 to offload each individual task whenever it arrives at the queue of F_0 so that the execution delay of the task is minimized. Based on the resource table, the algorithm also considers dividing the task dynamically, thereby exploiting the parallel computation of subtasks to achieve the delay reduction objective. In addition, we assume that the data transmission and data processing are not performed simultaneously by F_0, thus the algorithm requires an efficient scheduling mechanism for subtasks. The next section formulates an optimization problem that combines task offloading and subtask scheduling to minimize the execution delay of an individual task.

For the sake of clarity, the key notations used in this paper are summarized Table II.

IV. Optimization Problem for Minimization of Task Execution Delay

The overall delay for executing task t_i (D_i) is the interval from the arrival time of task (T_i^a) at the queue of F_0 until the end of the processing time (T_i^f) at an offloadee. As illustrated in Fig. 2, $D_i = T_i^f - T_i^a$.

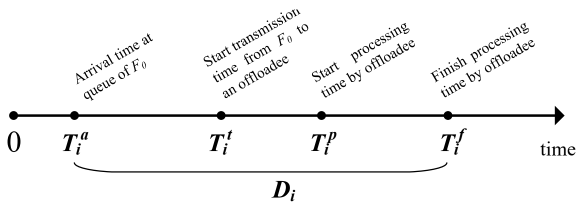
Arrival time at queue of F_0
Start transmission time from F_0 to an offloadee
Start processing time by offloadee
Finish processing time by offloadee
0
T_i^a
T_i^t
T_i^p
T_i^f
time
D_i

Fig. 2. Timeline to calculate the task execution delay.

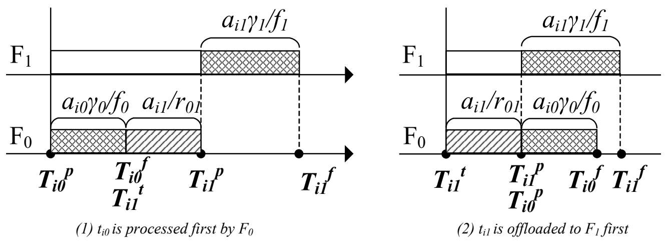

Fig. 3. T_{ij}^p depends on the relationship between W_n and transmission delay (a_{ij}/r_{0n}).

Suppose that there is no data dependency between the subtasks of t_i; therefore, the subtasks can be computed in parallel by different HNs. We also assume that the results of processing the subtasks do mot need to be aggregated because the sizes of results are small and their aggregation time can be negligible. Therefore, $D_i = \max\{D_{ik}\}$, $k = 0, \cdots, N$. Notably, D_{ik} = T_{ik}^f - T_i^a. In this sense, besides subtask offloading (i.e., which fog node processes which subtask), subtask scheduling (i.e., which subtask is transmitted or processed first) has a significant impact on the total task execution delay.

The priority of tasks is not considered during processing; thus, they will be processed according to the FCFS policy in the queue of F_0. In addition, to minimize the execution delay of individual tasks, scheduling is employed for the substasks of a task. Because data transmission and data processing are not performed simultaneously by F_0, an optimal schedule is required to minimize the delay.

Lemma 4.1 specifies an efficient schedule for exploiting the parallel computation of subtasks to reduce the delay.

Lemma 4.1: When the subtasks of a task are processed by both F_0 and HNs, data transmission is performed before data computation to obtain the benefit of parallel computation .

Proof For the illustrative simplicity, we suppose that task t_i is divided into two subtasks, t_{i0} and t_{i1}, which are processed by F_0 and F_1, respectively.

We consider the first scenario, as illustrated in Fig. 3, in which the the queue of F_1 is empty at the time of the offloading decision (i.e., $W_1 = 0$). We consider Case 1, in which t_{i0} is processed first by F_0. The total delay to execute t_{i0} is:

$$D_{i0} = \frac{a_{i0}\gamma_0}{f_0}. \tag{2}$$

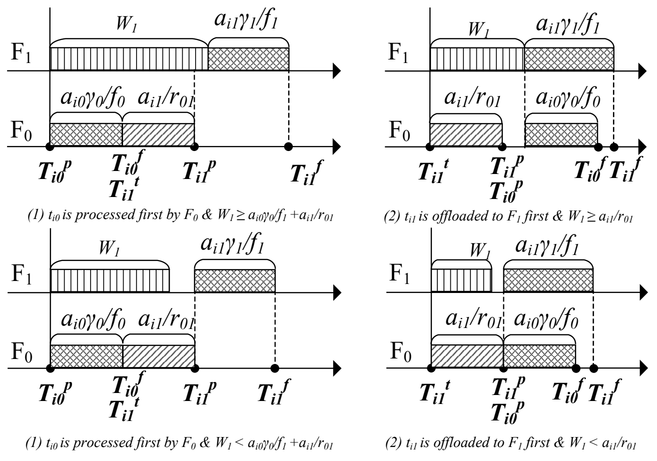
F1
F0
W1
ai1γ1/f1
ai0γ0/f0
ai1/r01
Ti0p
Ti0f
Ti1t
Ti1p
Ti1f
(1) ti0 is processed first by F0 & W1 ≥ ai0γ0/f1 + ai1/r01
(2) ti1 is offloaded to F1 first & W1 ≥ ai1/r01
(1) ti0 is processed first by F0 & W1 < ai0γ0/f1 + ai1/r01
(2) ti1 is offloaded to F1 first & W1 < ai1/r01

Fig. 4. T^p_{ij} depends on the relationship between W_n and transmission delay (a_{ij}/r_{0n}).

Because data processing and data transmission are not handled simultaneously by F_0, the total delay to execute t_{i1} including D_{i0}, as the waiting time is calculated by $D_{i1} = D_{i0} + a_{i1}/r_{01} + a_{i1}\gamma_1/f_1$. Because $D_i = \max\{D_{i0}, D_{i1}\}$, $D_i = a_{i0}\gamma_0/f_0 + a_{i1}/r_{01} + a_{i1}\gamma_1/f_1$. When t_{1i} is processed first in Case 2, the total delay to execute the subtasks is calculated as follows:

$$D^*_{i1} = \frac{a_{i1}}{r_{01}} + \frac{a_{i1}\gamma_1}{f_1}. \tag{3}$$

$$D^*_{i0} = \frac{a_{i1}}{r_{01}} + \frac{a_{i0}\gamma_0}{f_0}. \tag{4}$$

The total delay to finish t_1 in Case 2 is $D^*_i = \max\{D^*_{i0}, D^*_{i1}\}$. Obviously, $D^*_i < D_i$; thus, the lemma is proved.

We now examine the second scenario where the queue of F_1 is not empty (i.e., $W_1 > 0$), as shown in Fig. 4.

In Case 1, $D_{i0} = a_{i0}\gamma_0/f_0$ because t_{i0} is executed first by F_0. Moreover, the delay to finish t_{i1} is achieved by:

$$D_{i1} = \begin{cases} W_1 + \frac{a_{i1}\gamma_1}{f_1}, & \text{if } W_1 \geq D_{i0} + \frac{a_{i1}}{r_{01}}, \\ D_{i0} + \frac{a_{i1}}{r_{01}} + \frac{a_{i1}\gamma_1}{f_1}, & \text{otherwise.} \end{cases} \tag{5}$$

Obviously, D_i is minimized when $W_1 < D_{i0} + a_{i1}/r_{01}$; thus, $D_i = a_{i0}\gamma_0/f_0 + a_{i1}/r_{01} + a_{i1}\gamma_1/f_1$.

In Case 2 of the second scenario (i.e., t_{i1} is transmitted first to F_1), the delay to finish t_{i0} is $D^*_{i0} = a_{i1}/r_{01} + a_{i0}\gamma_0/f_0$. The execution delay of t_{i1} is as follows:

$$\begin{cases} D^*_{i1} = \frac{a_{i1}}{r_{01}} + \frac{a_{i1}\gamma_1}{f_1}, & \text{if } W_1 < \frac{a_{i1}}{r_{01}}, \\ \overline{D}^*_{i1} = W_1 + \frac{a_{i1}\gamma_1}{f_1}, & \text{if } W_1 \geq \frac{a_{i1}}{r_{01}}. \end{cases} \tag{6}$$

Recall that the delay to finish t_i is obtained by $D^*_i = \max\{D^*_{i0}, D^*_{i1}\}$ or $D^*_i = \max\{D^*_{i0}, \overline{D}^*_{i1}\}$. Therefore, if $D^*_i = D^*_{i0}$ or $D^*_i = D^*_{i1}$, then $D^*_i < D_i$. This means that Case 2 is beneficial for delay reduction compared with Case 1. If $D^*_i = \overline{D}^*_{i1}$, we also have $D^*_i < D_i$ because the condition $W_1 < a_{i0}\gamma_0/f_0 + a_{i1}/r_{01}$ still holds (see (5)).

Consequently, DCTO has the potential to reduce the delay only if all subtasks scheduled to be offloaded to HNs are transmitted first from F_0.

We now consider a more general scenario in which the task t_i is divided into three subtasks t_{i0}, t_{i1}, t_{i2}. We suppose that these subtasks are processed by F_0 locally and offloaded by F_1 and F_2, respectively. Based on the aforementioned proof, for any two subtask t_{i0} and t_{i1} which are processed locally by F_0 and offloaded by F_1, the benefit of delay reduction is enabled when t_{i1} is transmitted to F_1 before t_{i0} is processed. Now we assume that for the three subtasks, this benefit can be achieved when t_{i2} is transmitted lastly to F_2 after processing t_{i0} is completed. In this case, the total delay D_{i2} to finish t_{i2} is data transmission delay of t_{i1}, data processing delay of t_{i0}, data transmission delay of t_{i2}, and data processing delay of t_{i2} at F_2. Obviously, $D_{i2} \geq \max\{D_{i0}, D_{i1}\}$. Therefore, the assumption of processing t_{i2} lastly is infeasible to get the benefit of delay reduction.

Based on the lemma, we define o^k_{ij} to represent the scheduling order between subtask t_{ik} and t_{jk} for wireless transmission. In particular, if t_{ik} is scheduled on a wireless link before t_{jk}, we have $o^k_{ij} = 1$; otherwise, $o^k_{ij} = 0$. Because t_{ik} is transmitted either before or after t_{jk}, we have:

$$o^k_{ij} + o^k_{ji} = 1, \forall i, j = 0, ..., N \ \& \ i \neq j. \tag{7}$$

We define T^t_{ij} as the start time for transmitting t_{ij}. There is no overlap between the wireless transmission times of the two subtasks t_{ij} and t_{ik}; therefore, their start transmission times must satisfy the following:

$$T^t_{ij} - \sum_{k=0}^{N} (T^t_{ik} + \frac{a_{ik}}{r_{0k}}) \geq -Lo^i_{jk}, \tag{8}$$

where L is a large possible constant, and $r_{00} = \infty$ indicates that t_{ik} is processed by F_0. Notably, the propagation delay is neglected because it is much smaller than the transmission delay (a_{ik}/r_{0n}).

Additionally, the start processing time T^p_{in} of t_{in} must not be smaller than the arrival time at an offloadee as well as the finish time of the last task in the queue of the offloadee (see Fig. 5). Therefore,

$$T^p_{in} \geq T^t_{in} + \sum_{n=1}^{N} (\theta_n W_n + (1 - \theta_n)\frac{a_{in}}{r_{0n}}), \tag{9}$$

where

$$\theta_n = \begin{cases} 1, & \text{if } W_n \geq \frac{a_{in}}{r_{0n}}, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Because all the tasks in the queue of F_n $(n = 1, \cdots, N)$ are transmitted only from F_0, the expected waiting time in the queue of F_n can be achieved by:

$$W_n = \gamma_n \sum_i a_i/f_n, \tag{11}$$

where $\sum_i a_i$ is the total data size of tasks resided in the queue of F_n.

In addition, the buffer sizes of queues of fog devices are limited, thus

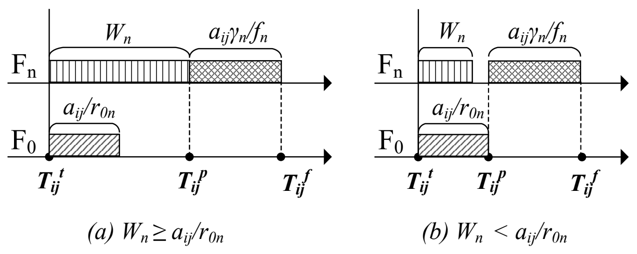

Fig. 5. T_{ij}^p depends on the relationship between W_n and transmission delay (a_{ij}/r_{0n}).

$$W_n\frac{f_n}{\gamma_n} + \sum_{i=0}^{N} a_{in} \leq M_n. \tag{12}$$

The end time T_{ij}^f to finish t_{ij} is calculated as follow:

$$T_{in}^f = T_{in}^p + \sum_{n=0}^{N} \frac{a_{in}\gamma_n}{f_n}. \tag{13}$$

Recall that $D_{ij} = T_{ij}^f - T_i^a$, therefore $D_i = \max_j\{D_{ij}|j = 0, \cdots, N\} = \max_j\{T_{ij}^f\} - T_{ij}^a$. The optimization problem **P** is modeled as follow to minimize D_i:

$$\textbf{P:} \quad \min_{a_i,\alpha_i,T_i^t,T_i^p,o_i} \quad D_i \qquad \text{s. t.} \quad (1), (7), (9), (12), (8), \tag{14}$$

where a_i=$[a_{i0}, a_{i1}, \cdots, a_{iN}]$, $T_i^t = [T_{i0}^t, \cdots, T_{iN}^t]$, $T_i^p = [T_{i0}^p, \cdots, T_{iN}^p]$, and $o_i = [o_{01}^i, \cdots, o_{0N}^i, \cdots, o_{N0}^i, \cdots, o_{NN-1}^i]$.

All the constraints introduced in the problem are linear except the equation 9. However, it can be relaxed to be linear by using the Big-M method [36]. Then, the problem **P** can be solved by mixed-integer linear programming (MILP), thus requiring a low computation complexity as compared to the algorithms in the related works. The solutions to the problem **P** indicates dynamic approaches that specify which tasks and how many subtasks should be divided, which subtask is processed by which fog devices, and also the transmission and processing order of subtasks. In the next section, many simulation scenarios are conducted to examine the performance of proposed offloading schemes under impact of many factors.

V. Simulation and Performance Evaluation

A. Simulation Environment Setup

The event-driven framework supported SimPy library in Python is used to design process-based discrete-event simulation scenarios for the studied FCS, which investigate the performance of DCTO as well as the comparative study with the related algorithms. In addition, we use CVX solver [37] to solve the optimization problem. Table III summarizes the important parameters and values for the simulation scenario, where U(x,y) indicates the uniform distribution on interval [x, y].

All the simulation results are averaged over 100 simulation rounds and each round lasts 100 seconds according to the clock of CPU. We compare the proposed DCTO approach with RCO, and POST. Recall that RCO considers to schedule and offload multiple tasks at each time of decision-making, and task division is not employed. In addition, the queuing states of fog devices and task request rate are not taken into account in the RCO algorithm. Meanwhile, POST uses the game theory to find the GNE point for mapping task/subtasks to the fog devices [26]. The impact of task request rate, task scheduling, and queuing states is not also considered in POST.

TABLE III
Parameters for Simulation.

Parameters	Values
N	$\{4, 5, 6, 7\}$
a_i	U(5,10) (MB)
λ	$\{0.5, 1.0, 1.5, 2.0, 2.5\}$ (tasks/s)
M	Randomly in $\{128, 256\}$ (MB)
γ	U(500,1000) (cycles/bit)
f	U(5,15) (GHz)
r_{0i}	U(100,150) (Mb/s)

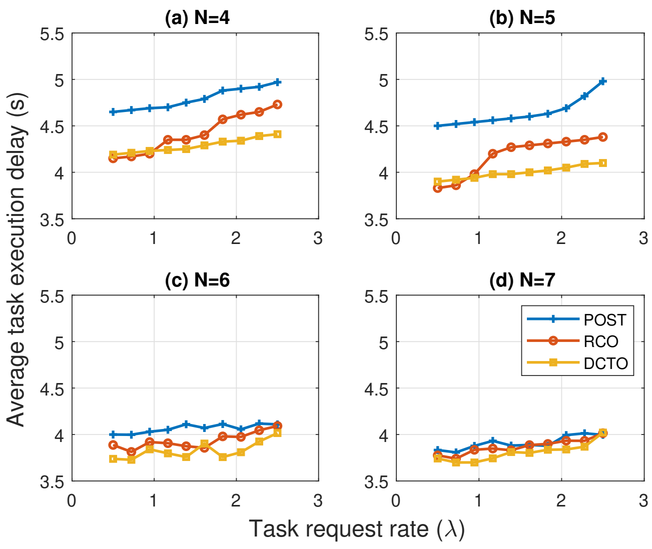

Fig. 6. The average execution delay offered by the comparative offloading schemes as a function of λ and N.

B. Evaluation and Analysis

We conduct the first simulation scenario, in which all the queues of fog nodes are empty initially. Fig. 6 depicts the average delay achieved by POST, RCO, and DCTO when λ and N are varied.

DCTO always outperforms RCO and POST because it can dynamically adjust the task offloading decision according to the states of available fog resources. When the queues are empty at the beginning of simulation, there would be no beneficial to reduce the delay through task division at the low task arrival rate (i.e., $\lambda = \{0.5, 1\}$). Therefore, DCTO and RCO can obtain the similar performance and be better than POST due to the optimal scheduling of tasks. When the task rate increases, the delay is no longer impacted by the queuing states of fog devices. The task division and the associated parallel computation of subtasks are beneficial to cope with the situation. In addition, the increase of number of HNs

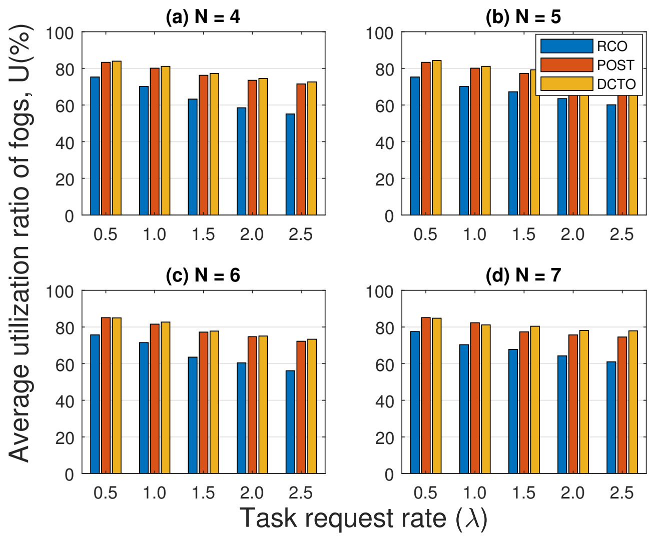

Fig. 7. Average utilization ratio of HNs under the impact of λ.

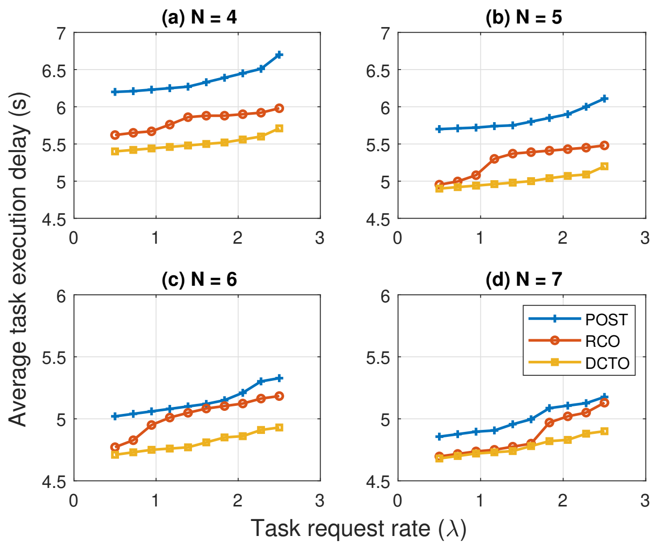

Fig. 8. The average task execution delay under the impact of λ, N, and waiting times in the queues of fog nodes.

helps in slowing the increase of average delay because more available resources are utilized to process the tasks/subtasks. In particularly, DCTO which incorporates the fog resource awareness and optimal tasks/subtasks scheduling is able to minimize the average execution delay. Notably, as the number of HNs increases ($N = 6, 7$), DCTO is able to exploit efficiently the available resource of fogs to perform the parallel computation, thus enabling a gradual delay increase even when the task arrival rate increases.

Fig. 7 supports such the claim through the average utilization ratio (U) of HNs. As shown in this figure, the task division employed in the POST and DCTO schemes can increase the utilization ratio of fog computing devices since the limited resource fogs can process the subtasks. As the rate of request is low the task division mechanism is not significantly beneficial.

Although the both DCTO and POST uses the task division and parallel commutation of subtasks to improve the fog resource utilization, the benefit of delay reduction is not the same. This is because, DCTO includes the optimal subtask scheduling based on the resource states of HN to minimize the delay. In addition, POST takes more running time for iteration to reach the near-optimal task-resource mapping.

In the simulation scenario 2, we analyze the impact of queuing state (i.e., waiting time in queue) on the average delay. Initially, all the queues of fog nodes have a number p of tasks and p is randomly selected from a set $\{1, 2, 3, 4\}$. In addition, the size of task is randomly selected from $\{3, 4, 5\}$ (MB). Notably, the expected waiting times (W) in the queues of HNs can be derived as (11). Fig. 8 shows the average execution task delay obtained by DCTO, RCO, and POST.

Besides the improved performance of DCTO as compared to RCO and POST, the simulation result shows that parallel computation and scheduling employed simultaneously in DCTO can decrease the impact of queuing states of fogs effectively. Although, the task division and parallel subtask computation are used in POST, the performance efficiency is not maximized. That is because POST only tries to obtain the optimal mapping between tasks and subtasks with the available fog resources through time slots. This mechanism is negatively influenced by waiting times in the queues of fog devices. In other words, POST without the usage of scheduling and resource allocation is unable to exploit the fog resources maximally since some fogs with queued tasks are not allocated to perform the task offloading. Meanwhile, DCTO divides the tasks adaptively aiming to schedule and allocate the fog resources for offloading the subtasks effectively. RCO also is unable to maximize the fog resources since the task division is not employed. For example, some fogs with less available computation and storage resources may not be allocated to process single tasks in the RCO algorithms. Meanwhile, they absolutely process subtasks as specified in DCTO if scheduled efficiently.

To examine the impact of task size, we conduct the third simulation scenario, in which all the tasks have the same sizes (i.e., $a_i = a$). Two types of tasks are considered including medium tasks ($a = 5$ MB) and heavy tasks ($a = 10$ MB).

Simulation results as shown in Fig. 9 indicate that when all the tasks are medium ($a = 5$ MB), the performance of DCTO is not significantly better than RCO since the division of medium size tasks is not beneficial. In addition, DCTO is just to minimize the execution delay of each single task while RCO can schedule multiple tasks, just achieving a better average task reduction when the task arrival rate is low. However, as the task arrival rate is high, there exist longer queuing delays in some powerful fog devices, thus the task division employed in DCTO is essential to balance the workload as well as reduce the delay through the parallel computation. As a result, the performance of DCTO is slightly improved. As expected, the heavy tasks (i.e., $a = 10$ MB) have significant impacts on the fog computing environment since some fogs with limited resources individually are unable to process the whole data of single task. Therefore, the system performance is affected negatively by workload imbalance of heterogeneous

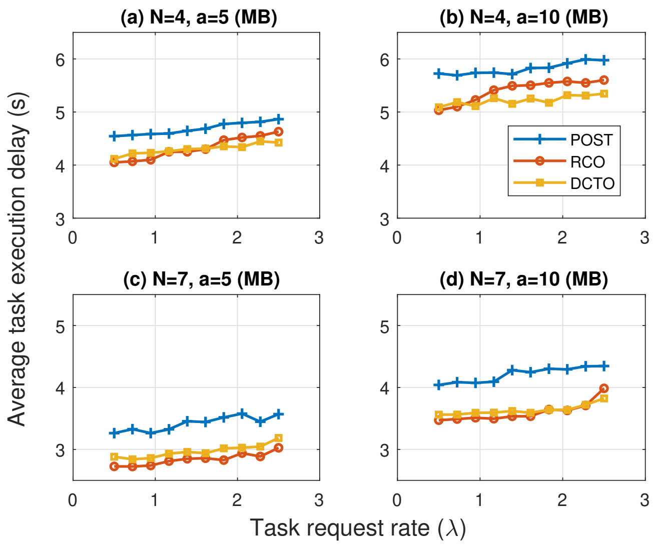

Fig. 9. The average task execution delay under the impact of task sizes and number of HNs.

fog devices. Although optimal resource allocation is employed in RCO, the delay is higher than DCTO, which exploits task division and parallel computation to decrease the task execution delay effectively.

VI. Conclusions

This paper proposes DCTO, a dynamic collaborative task offloading approach for FCS to reduce the delay of task execution. DCTO is used by the fog controller of FCS to derive the offloading solution dynamically based on the state of available resources of HNs. Accordingly, by applying task division technique and the associated parallel computation, DCTO allows a task to be processed by multiple fog computing devices to further reduce the overall task execution delay. The optimal scheduling of subtask transmission and subtask processing is also integrated in the DCTO policy to contribute to the delay reduction objective. The intensive simulation results show that DCTO is an effective offloading solution to achieve the reduced execution delay for fog computing environment, in which the resources of fog computing devices are heterogeneous and complicated. Especially, DCTO outperforms the existing related offloading schemes in the scenarios that the rate of task requests is high and task sizes are large.

VII. Future Works and Discussions

On the basis of the proposed dynamic offloading mechanism, there are many possible directions to develop and expand DCTO algorithms as future research works. Firstly, the stronger capabilities of fog computing nodes including F_0 and HNs should be considered that allow to offload multiple tasks for each time of offloading decision making in. In this case, the scheduling model is required to be redesigned to jointly schedule the order of tasks and also subtasks of different tasks. In addition, the fog nodes can perform data transmission and data processing simultaneously, thus the tasks/subtasks scheduling should be adjusted accordingly. Another direction for future research is to investigate DCTO in the large-scale systems. In the large-scale model, more domains can co-exist, resulting in more fog computing systems needed to be deployed. In such case, the collaboration between systems in different domains can be considered to enhance the flexibility of overall IoT systems as well as enhance the fog resource utilization.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014.

[2] Y. Saleem, N. Crespi, M. H. Rehmani, and R. Copeland, "Internet of things-aided smart grid: Technologies, architectures, applications, prototypes, and future research directions," *IEEE Access*, vol. 7, pp. 62962–63003, 2019.

[3] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial IoT data scheduling based on hierarchical fog computing: A key for enabling smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4590–4602, 2018.

[4] H. Tran-Dang, N. Krommenacker, P. Charpentier, and D.-S. Kim, "The Internet of things for logistics: Perspectives, application review, and challenges," *IETE Tech. Review*, pp. 1–29.

[5] H. Tran-Dang, N. Krommenacker, P. Charpentier, and D. Kim, "Toward the Internet of things for physical Internet: Perspectives and challenges," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4711–4736, 2020.

[6] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 112–121, 2014.

[7] H. Tran-Dang and D. Kim, "An information framework for Internet of things services in physical internet," *IEEE Access*, vol. 6, pp. 43967–43977, 2018.

[8] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. NCM*, 2009.

[9] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Comput. Syst.*, vol. 56, pp. 684–700, 2016.

[10] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. ACM MCC*, 2012.

[11] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 46–59, 2018.

[12] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of things realize its potential," *Comput.*, vol. 49, no. 8, pp. 112–116, 2016.

[13] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Generation Comput. Syst.*, vol. 87, pp. 278–289, 2018.

[14] H. Tran-Dang and D.-S. Kim, "Frato: Fog resource based adaptive task offloading for delay-minimizing iot service provisioning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2491–2508, 2021.

[15] T. Mattson, B. Sanders, and B. Massingill, *Patterns for Parallel Programming*, 1st ed. Addison-Wesley Professional, 2004.

[16] Y.-S. Jiang and W.-M. Chen, "Task scheduling in grid computing environments," in *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2014, pp. 23–32.

[17] A. Elgazar, K. Harras, M. Aazam, and A. Mtibaa, "Towards intelligent edge storage management: Determining and predicting mobile file popularity," in *Proc. IEEE MobileCloud*, 2018.

[18] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, 2018.

[19] J. Yao and N. Ansari, "Fog resource provisioning in reliability-aware IoT networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8262–8269, 2019.

[20] A. Yousefpour *et al.*, "Fogplan: A lightweight qos-aware dynamic fog service provisioning framework," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5080–5096, 2019.

[21] Y. Yang *et al.*, "Pomt: Paired offloading of multiple tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8658–8669, 2019.

[22] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, 2018.

[23] G. Zhang *et al.*, "Femto: Fair and energy-minimized task offloading for fog-enabled iot networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4388–4400, 2019.

[24] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "Dats: Dispersive stable task scheduling in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3423–3436, 2019.

[25] M. Mukherjee *et al.*, "Latency-driven parallel task data offloading in fog computing networks for industrial applications," vol. 16, no. 9, pp. 6050–6058, 2020.

[26] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "Post: Parallel offloading of splittable tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3170–3183, 2020.

[27] H. Tran-Dang and D.-S. Kim, "A survey on matching theory for distributed computation offloading in iot-fog-cloud systems: Perspectives and open issues," *IEEE Access*, vol. 10, pp. 118353–118369.

[28] H. Tran-Dang and D.-S. Kim, "Disco: Distributed computation offloading framework for fog computing networks," *J. Commun. Netw.*, pp. 1–11, 2023.

[29] H. Tran-Dang and D.-S. Kim, "A many-to-one matching based task offloading (mato) scheme for fog computing-enabled iot systems," in *Proc. ATC*, 2022.

[30] G. Lee, W. Saad, and M. Bennis, "An online optimization framework for distributed fog network formation with minimal latency," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2244–2258, 2019.

[31] K. Guo, M. Sheng, T. Q. S. Quek, and Z. Qiu, "Task offloading and scheduling in fog RAN: A parallel communication and computation perspective," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 215–218, 2020.

[32] H. Tran-Dang and D.-S. Kim, "Dynamic task offloading approach for task delay reduction in the IoT-enabled fog computing systems," in *Proc. IEEE INDIN*, 2022.

[33] H. Tran-Dang and D.-S. Kim, "Reinforcement learning for computational offloading in fog-based IoT systems: Applications, and open research issues," in *Proc. RIVF*, 2022.

[34] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues," *J. Commun. Netw.*, vol. 24, no. 1, pp. 83–98.

[35] M. Al-khafajiy *et al.*, "Improving fog computing performance via fog-2-fog collaboration," *Future Generation Comput. Syst.*, vol. 100, pp. 266–280, 2019.

[36] I. Griva, S. Nash, and A. Sofer, *Linear and Nonlinear Optimization: Second Edition*, ser. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics. [Online]. Available: https://books.google.co.kr/books?id=u63u_iNcnRkC

[37] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx.

Hoa Tran-Dang (M'17) received the B.E. degree in Electrical and Electronics Engineering from Hanoi University of Science and Technology (HUST), Vietnam and the M.S. degree in Electronics Engineering from Kumoh National Institute of Technology (KIT), South of Korea in 2010 and 2012, respectively. He pursued the Ph.D. degree with University of Lorraine, France during 2013-2017. He currently works in NSL laboratory, department of ICT Convergence Engineering at Kumoh National Institute of Technology, South of Korea as a Research Professor. His research interests include Internet of things (IoT), edge/fog computing, machine learning, artificial intelligence, distributed intelligence, physical internet, and resource management.

Dong-Seong Kim (S'98-M'03-SM'14) received his Ph.D. degree in Electrical and Computer Engineering from the Seoul National University, Seoul, Korea, in 2003. From 1994 to 2003, he worked as a Full-Time Researcher in ERC-ACI at Seoul National University, Seoul, Korea. From March 2003 to February 2005, he worked as a Postdoctoral Researcher at the Wireless Network Laboratory in the School of Electrical and Computer Engineering at Cornell University, NY. From 2007 to 2009, he was a Vistiting Professor with Department of Computer Science, University of California, Davis, CA. He is currently a Director of kit Convergence Research Institute and ICT Convergence Research Center (ITRC program) supported by Korean government at Kumoh National Institute of Technology. He is IEEE and ACM Senior Member. His current main research interests are real-time IoT, industrial wireless control network, networked embedded system and fieldbus.