IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 July 2023, accepted 21 July 2023, date of publication 25 July 2023, date of current version 31 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3298810

RESEARCH ARTICLE

Distributed Split Computing System in Cooperative Internet of Things (IoT)

SEUNG-YEON KIM[1] AND HANEUL KO[2], (Senior Member, IEEE)
[1]Department of Computer Convergence Software, Korea University, Sejong 30019, South Korea
[2]Department of Electronic Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, South Korea

Corresponding author: Haneul Ko (heko@khu.ac.kr)

This work was supported in part by the National Research Foundation (NRF) of Korea funded by the Korean Government [Ministry of Science, ICT and Future Planning (MSIP)] under Grant 2021R1I1A3042204, and in part by the Institute for Information & Communications Technology Planning & Evaluation (IITP) funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant 2021-0-00739.

ABSTRACT The split computing approach, where the head and tail models are respectively distributed between the IoT device and cloud, suffers from high network latency especially when the cloud is located far from the IoT device. To mitigate this problem, we introduce a distributed split computing system (DSCS) where an IoT device (called split computing requester) broadcasts a split computing request to its neighboring IoT devices. After receiving the request, the neighboring IoT devices (i.e., requestees) distributively determine whether or not to accept the split computing request by taking into account the unnecessary energy consumption and computation time. To minimize energy consumption while maintaining a specified probability of on-time computing completion, we develop a constrained stochastic game model. Then, a best-response dynamics-based algorithm is used to obtain the Nash equilibrium. The evaluation results demonstrate that the DSCS consumes can reduce more than 20% energy consumption compared to a probabilistic-based acceptance scheme, where the IoT devices accept a split computing request based on a predefined probability, while providing high on-time computing completion probability.

INDEX TERMS Split computing, stochastic game, energy consumption, inference time, distributed system.

I. INTRODUCTION

Deep neural networks (DNNs) are currently the most frequently used machine learning approach in intelligent mobile applications and have grown more popular owing to their accurate and reliable inference capability [1]. Meanwhile, despite the recent improvements in the computing capabilities of IoT devices, their performances fall far short of that of cloud computing. Thus, when conducting inference for the entire DNN model, a sufficiently low latency cannot be achieved. In addition, the battery capacities of IoT devices have severe limitations, especially for inference with high complexity. Therefore, there is increasing interest in the split computing approach [2], [3]. In this approach, the DNN is split into two subnetworks (i.e., head and tail models), and the head and tail models are distributed between the IoT device and cloud, respectively. The IoT device first conducts inference of the head model to obtain intermediate data (i.e., output of the head model). It then sends this intermediate data to the cloud. Using the intermediate data as the input, the cloud processes the tail model sequentially. However, this split computing approach suffers from high network latency between the IoT device and cloud, especially when the cloud is located far from the IoT device [4], [5].

To mitigate this problem, we introduce a distributed split computing system (DSCS); here, an IoT device simply determines whether to use the split computing approach and the splitting point by considering its available computing power as well as computing deadline. If the IoT device decides to use the split computing approach, it conducts inference of the head model based on the splitting point. Then, the IoT device (which is the split computing requester) broadcasts a split computing request that includes the splitting point, intermediate data, computing latency of the head model, and computing deadline to its neighboring IoT devices. After receiving the request, the neighboring IoT devices

The associate editor coordinating the review of this manuscript and approving it for publication was Eyuphan Bulut.

(i.e., requestees) distributively determine whether or not to accept the split computing request by taking into account the unnecessary energy consumption and computation time. Because the total number of IoT devices (i.e., requestees) accepting the split computing request affects the amount of energy consumed and the probability of completing the computations on time, each IoT device should consider the actions of its neighboring IoT devices. In this context, we formulate a constrained stochastic game model and utilize a best-response dynamics-based algorithm to obtain the multi-policy constrained Nash equilibrium with minimized energy consumption while maintaining desirable on-time computing completion probability. The evaluation results show that the DSCS consumes can reduce more than 20% energy consumption compared to a probabilistic-based acceptance scheme, where the IoT devices (i.e., requestees) accept a split computing request based on a predefined probability, while providing high on-time computing completion probability. Moreover, it is found that the best-response dynamics-based algorithm converges quickly to the Nash equilibrium within a few iterations.

The main contributions of this study are as follows: 1) the proposed system is a pioneering effort in which the actions of the split computing requestees are distributively decided to optimize the performance of the split computing system; 2) the optimal policy of the requestees regarding acceptance of the computing request can be obtained in a few iterations, indicating that the proposed algorithm can be implemented in actual systems without significant signaling cost; 3) we show and scrutinize the evaluation results under various conditions to provide guidance for constructing a DSCS.

The remainder of this manuscript is structured as follows. The related works are detailed in Section II, and the proposed DSCS is described in Section III. The stochastic game model development is detailed in Section IV. The evaluations are reviewed in Section V, and the final conclusions are summarized in Section VI.

II. RELATED WORK

Many reported studies have investigated the possibility of lowering task completion times in split computing environments [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18].

Kang et al. [6] created an automated two-step model splitting technique. In the first step, performance prediction models are created for each layer. In the second step, the splitting point is dynamically determined from the prediction models by considering the importance of performance metrics. Li et al. [7] suggested a model splitting framework that considers early exit and allows the inference task to be undertaken at an appropriate intermediary layer; they decided the exit and splitting points concurrently to maximize the accuracy of inference while ensuring that the task completion time remained below a specified threshold. Laskaridis et al. [8] presented a system that continually monitors the resources of the edge cloud and mobile device to decide the splitting point by considering application requirements. Krouka et al. [9] introduced a technique involving pruning and compression before splitting the DNN model to minimize energy consumption by the mobile device while assuring correctness of inference. Yan et al. [10] jointly optimized placing and splitting of the model to minimize energy consumption and reduce task completion time while accounting for the network dynamics. Eshratifar et al. [11] determined several optimal splitting points by converting the presented model into a well-known one to exploit existing algorithms. Zhou et al. [12] suggested a strategy to minimize task completion time by pruning the model and compressing the intermediate data. He et al. [13] exploited a queuing model for task completion time to formulate a joint optimization problem regarding the splitting point and resource allocation, which can be divided into subproblems; moreover, they designed a heuristic algorithm to solve the subproblems sequentially. Tang et al. [14] designed an algorithm that uses the structural characteristics of the model splitting problem to obtain its solution in polynomial time. Wang and Zhang [15] designed a split computing architecture that exploits the error-tolerant characteristics of the intermediate data to reduce the communication overhead; in this architecture, the controller decides if retransmission is needed depending on the error rate. Wang et al. [16] proposed a multiple-splitting-points decision system that determines several optimal splitting points in real time with low signaling overhead. Matsubara et al. [17] suggested a supervised compression method that discretizes the intermediate data to avoid high communication overhead. Ahn et al. [18] introduced a system in which the DNN model is partitioned and deployed between the IoT device and cloud to improve inference accuracy and reduce task completion time. In [19] and [20], to minimize system energy consumption, authors developed a distributed DNN computing system orchestrating cooperative inference among multiple IoT devices by considering available computing power and network condition of IoT devices. Zhang et al. [21] introduced a collaborative and adaptive inference system that can handle various types of DNN models and optimize the trade-off between the computation and synchronization. In [22] and [23], authors introduced a method dynamically detecting the best splitting point for a given DNN based on the communication channel state, batch size, and multiclass categorization. In [24], authors proposed a novel framework for the split computing, in which a round-robin schedule to select a device and Hungarian optimization algorithm to assign a layer to the device are exploited.

However, there are no existing works for optimizing the split computing performance from the perspective of the requestees in a distributed manner.

III. DISTRIBUTED SPLIT COMPUTING SYSTEM

Figure 1 shows the proposed DSCS in which an IoT device (i.e., requester) generates the computing task periodically. During the computing task, the IoT device checks its available computing power and task deadline. If the available power is

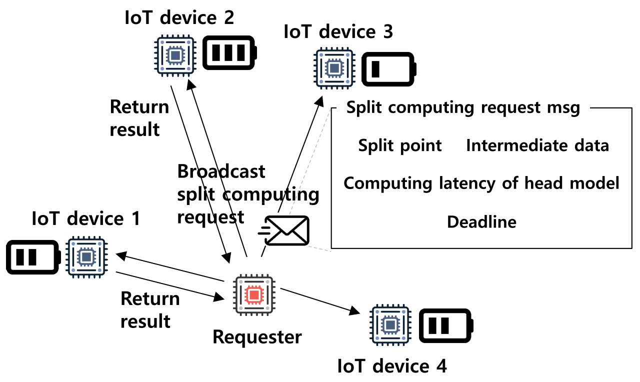


FIGURE 1. System model.

sufficient to complete the task within the deadline, the IoT device performs inference for the entire DNN model. Otherwise, the IoT device decides the splitting point according to its available computing power. Specifically, the entire DNN model is split at the mth splitting point, and m is obtained as $\lceil \alpha C \rceil$ where α is a scale factor and C is the available computing power of the IoT device. Note that the scaling factor can be set to $\frac{N_L}{C^{\max}}$ where N_L and $C^{\max}$ denote the total number of layers and the maximum computing power of the IoT device, respectively. After deciding the splitting point, the IoT device conducts inference for the head model with the mth splitting point to obtain the intermediate data (i.e., result of the head model). Then, the IoT device broadcasts the split computing request message to its neighboring IoT devices using LoRa [25][1]. This split computing request message includes the 1) splitting point, 2) intermediate data, 3) computing latency of the head model, and 4) task deadline.

After receiving the split computing request, the neighboring IoT devices (i.e., requestees) distributively determine whether to accept the split computing request by considering the unnecessary energy consumption and computation time. If the IoT device determines not to accept the request, it does nothing. Otherwise, it conducts inference of the tail model with the mth splitting point. At this point, the only neighboring IoT devices who can complete the task within its deadline accept the request. Therefore, even though the IoT device (i.e., requester) does not consider the computing power of the neighboring IoT devices (i.e., requestees), the on-time computing completion can be achieved. After completing inference, the IoT device returns the outcome of the tail model to the requester.

As the number of IoT devices accepting the request increases, the probability that at least one IoT device can complete the computations within the deadline (i.e., on-time computing completion probability) also increases. However, excessive duplicate acceptances increase the total amount of energy consumed. This means that each IoT device should consider the actions of others to achieve tradeoff between unnecessary energy consumption and on-time computing completion probability. Accordingly, we develop a constrained stochastic game model to minimize the energy consumed while ensuring that the on-time computing completion probability remains above a specified threshold; this model is explained in the following section.

[1]Note that IoT devices (LoRa class C devices) can transmit the packet without the synchronization [26].

TABLE 1. Summary of notations.

Notation	Description
$\mathbf{S}$	Global state space
$\mathbf{S_i}$	Local state space of IoT device i
$\mathbf{P_i}$	State space of IoT device i for the splitting point
$\mathbf{C_i}$	State space of IoT device i for the available computing power
$\mathbf{H_i}$	State space of IoT device i for the channel gain
$\mathbf{E_i}$	State space of IoT device i for the energy level
$\mathbf{A}$	Global action space
$\mathbf{A_i}$	Local action space of IoT device i
L	Computing completion time
L^H	Computing latency of the head model
L_i^D	Latency of transmitting the intermediate data from the requester to IoT device i
L_i^T	Computing latency of the tail model at IoT device i
T_i^R	Transmission rate between the requester to IoT device i
λ_i	Probability that IoT device i can complete the computing within the deadline
λ_{-i}	Probability that at least one IoT device except IoT device i can complete the computing within the deadline

IV. CONSTRAINED STOCHASTIC GAME

In this section, we present the development of a constrained stochastic game model [27], [28] to accomplish distributed implementation of the split computing service. In the game, N_I players (i.e., N_I requestees) exist, and they have five tuples: 1) local state space; 2) local action space; 3) transition probability; 4) cost function; 5) constraint function. Table 1 summarizes some important notations.

A. STATE SPACE

Let $\mathbf{S_i}$ be a finite local state space of player i (i.e., IoT device i or requestee i). Then, the global state space $\mathbf{S}$ can be represented as $\prod_i \mathbf{S_i}$, where $\prod$ is the Cartesian product. $\mathbf{S_i}$ can be defined as

$$\mathbf{S_i} = \mathbf{P_i} \times \mathbf{C_i} \times \mathbf{H_i} \times \mathbf{E_i}, \tag{1}$$

where $\mathbf{P_i}$ and $\mathbf{C_i}$ are the state spaces of IoT device i for the splitting point and available computing power, respectively. In addition, $\mathbf{H_i}$ and $\mathbf{E_i}$ denote the state spaces of IoT device i for the channel gain and energy level, respectively. To denote an element of each local state space of the IoT device i, we use italic letters (i.e., $P_i \in \mathbf{P_i}$, $C_i \in \mathbf{C_i}$, $H_i \in \mathbf{H_i}$ and $E_i \in \mathbf{E_i}$).

Without loss of generality, we assume that the entire DNN model consists of M layers. Then, $\mathbf{P_i}$ can be described as

$$\mathbf{P_i} = \{1, 2, \ldots, M\}, \tag{2}$$

where $P_i = m$ denotes the mth splitting point.

When $C^{\max}$ is the maximum computing power of the IoT device, $\mathbf{C_i}$ can be represented by

$$\mathbf{C_i} = \left\{u_C, 2u_C, \ldots, C^{\max}\right\}, \tag{3}$$

where C_i and u_C are the computing power and unit computing power, respectively.

Since the channel is quantized to Q levels [29], $\mathbf{H_i}$ can be represented by

$$\mathbf{H_i} = \left\{h_1, h_2, \ldots, h_Q\right\}, \tag{4}$$

where $H_i = h_q$ represents the qth quantized channel gain.

When $E^{\max}$ denotes the maximum energy level of the IoT device, $\mathbf{E_i}$ can be represented by

$$\mathbf{E_i} = \left\{0, 1, 2, \ldots, E^{\max}\right\}, \tag{5}$$

where E_i is the energy level of the IoT device.

B. ACTION SPACE

Let $\mathbf{A_i}$ be a finite local action space of player i. The global action space $\mathbf{A}$ is denoted by $\prod_i \mathbf{A_i}$. Here, the IoT device can accept or reject the computing request. Therefore, $\mathbf{A_i}$ can be represented by

$$\mathbf{A_i} = \{0, 1\}. \tag{6}$$

Here, for $A_i \in \mathbf{A_i}$, $A_i = 1$ denotes that IoT device i accepts the computing request and $A_i = 0$ means that IoT device i does not accept the computing request.

C. TRANSITION PROBABILITY

Let $P[S_i'|S_i, A_i]$ be a transition probability from the current state S_i to the next state when the IoT device i performs the chosen action A_i. The energy of the IoT device i decreases when it accepts the request. In addition, depending on the splitting point, the energy consumed by IoT device i varies. That is, the transition of E_i is influenced by P_i and A_i. Meanwhile, the other states change independently. Thus, the transition probability from the current state $S_i = [P_i, C_i, H_i, E_i]$ to next state $S_i' = [P_i', C_i', H_i', E_i']$ can be represented as (7), shown at the bottom of the next page.

According to the source condition, the harvesting energy volume changes. Therefore, it can be assumed that the harvested energy k during the decision epoch in the IoT device i follows a Poisson distribution with mean λ_E, $P_E(\lambda_E, k)$ [30]. That is, the energy level of IoT device i can increase by k with probability $P_E(\lambda_E, k)$ when it does not accept the computing request (i.e., $A_i = 0$) and its current energy level is less than the maximum energy level $E^{\max}$. Note that the IoT device i cannot harvest more energy if its current energy level is maximum. To summarize, the corresponding probabilities can be expressed as (8) and (9), shown at the bottom of the next page.

When the IoT device i accepts the split computing request (i.e., $A_i = 1$), it consumes $f_E(P_i)$ units of energy, where $f_E(P_i)$ is a function that returns the energy consumption for the tail model with splitting point P_i. If the IoT device i has insufficient energy for the tail model (i.e., $E_i < f_E(P_i)$)[2], then it does not consume any energy even though it accepts the split computing request. In addition, its energy can increase by k the probability $P_E(\lambda_E, k)$. Therefore, $P[E_i'|E_i \geq f_E(P_i), P_i, A_i = 1]$ and $P[E_i'|E_i < f_E(P_i), P_i, A_i = 1]$ can be represented as (10) and (11), shown at the bottom of the next page, respectively.

It is assumed that the available computing power of the requesting IoT device follows a discrete uniform distribution [31]. Then, since the requesting IoT device decides the splitting point according to its available computing power, the transition probability of P_i can be defined as

$$P[P_i'|P_i] = \begin{cases} \frac{1}{M}, & \text{if } P_i' \in \{1, 2, \ldots, M\} \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The available computing power of the IoT device i (i.e., C_i) is assumed to follow a Poisson distribution with mean λ_C, $P_C(\lambda_C, k)$, and $P[C_i'|C_i]$ can be defined as follows:

$$P[C_i'|C_i] = \begin{cases} P_C(\lambda_C, k), & \text{if } C_i' = k \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

In the channel gain with Q quantized levels [29], when the channel model follows Rayleigh fading with average channel gain ρ, $P[H_i'|H_i]$ is defined as follows [32]:

$$P[H_i'|H_i] = \begin{cases} F(h_{q+1}) - F(h_q), & \text{if } h_q \leq H_i' < h_{q+1} \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

where $F(x) = 1 - e^{-x/\rho}$.

D. COST FUNCTION

The energy consumption of the IoT device is exploited as the cost function $r(S_i, A_i)$, which is defined as

$$r(S_i, A_i) = A_i f_E(P_i). \tag{15}$$

E. CONSTRAINT FUNCTION

To provide high on-time computing completion probability, the constraint function $c(S_i, A_i)$ is defined. First, the computing completion time is derived as follows.

Because the size of output the DNN model output is generally much smaller than that of the intermediate data [33], the latency for result transmission from the IoT device i to split computing requester can be neglected. Then, the computing completion time can be calculated as the summation of the computational latency L^H of the head model, latency L^D for transmitting the intermediate data from split computing requester to IoT device i, and computing latency L^T of the tail model at IoT device i. Therefore, the computing completion time L can be represented by

$$L = L^H + L_i^D + L_i^T. \tag{16}$$

Note that the computing latency L^H of the head model is included in the split computing request message. Meanwhile,

[2]In this case, the IoT device i cannot conduct the tail model.

the transmission latency L^D of the intermediate data from the requester to IoT device i can be calculated as

$$L_i^D = \frac{f_D(P_i)}{T_i^R}, \tag{17}$$

where $f_D(P_i)$ is a function that returns the intermediate data size with the splitting point P_i. In addition, T_i^R denotes the effective transmission rate between the requester and IoT device i, which is obtained from the spreading factor chosen according to γ as [34] and [35]. Then γ is given by

$$\gamma = \frac{P_T |H_i|^2}{\sigma^2}, \tag{18}$$

where P_T and σ^2 represent the transmission power and noise power, respectively.

The computing latency L^T of the tail model for the IoT device i can be calculated as

$$L_i^T = \frac{f_F(P_i)}{C_i}, \tag{19}$$

where $f_F(P_i)$ is a function that returns floating-point operations (FLOPS) of the tail model for the splitting point P_i.

The probability λ_i that the IoT device i can complete computations within the deadline D can be calculated as

$$\lambda_i = P[A_i = 1] \cdot P\left[L^H + L_i^D + L_i^T < D\right], \tag{20}$$

where $P[A_i = 1]$ is the probability that IoT device i accepts the computing request with its current policy. Meanwhile, the probability λ_A that at least one IoT device can complete computing within the deadline is calculated as $\lambda_A = 1 - \prod_i (1 - \lambda_i)$, which can be considered as $c(S_i, A_i)$.

F. OPTIMIZATION FORMULATION

Let π_i and π be a stationary policy of the IoT device i and the stationary multi-policy of all players, respectively. Then, the long-term average energy consumption of the IoT device i can be defined as

$$\zeta_E(\pi) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E_\pi\left[r\left(S^t, A^t\right)\right], \tag{21}$$

where S^t and A^t are the global state and action at time t, respectively.

Meanwhile, IoT device i tries to maintain its on-time computing completion probability above a certain level. Thus, the constraint can be represented as

$$\zeta_C(\pi) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E_\pi\left[c\left(S^t, A^t\right)\right] \geq \theta_D, \tag{22}$$

where θ_D is the target on-time computing completion probability.

The multi-policy $\pi^* = \left(\pi_i^*, \pi_{-i}^*\right)$ is the constraint Nash equilibrium if $\zeta_C\left(\left(\pi_i^*, \pi_{-i}^*\right)\right) \leq \zeta_C\left(\left(\pi_i, \pi_{-i}^*\right)\right)$ for each IoT device among any π_i such that $\left(\pi_i, \pi_{-i}^*\right)$ is feasible [36]. To achieve the multi-policy constrained Nash equilibrium, the best response policy π_i^* of the IoT device i given any stationary policies of the other IoT devices π_{-i} can be defined as $\zeta_C\left(\left(\pi_i^*, \pi_{-i}\right)\right) \leq \zeta_C\left(\left(\pi_i, \pi_{-i}\right)\right)$.

Based on the LP problem, we can obtain the best response policy of IoT device i. Let $\phi_{i,\pi_{-i}}(S_i, A_i)$ denote the stationary probability for local state S_i and action A_i of the IoT device i when stationary policies of the other IoT devices π_{-i} are given. Then, the solution of the LP problem $\phi^*_{i,\pi_{-i}}(S_i, A_i)$ can be matched to the best response policy of the constrained stochastic game.

To minimize the average energy consumption of the IoT device i, the objective function can be expressed as

$$\min_{\phi(S,A)} \sum_S \sum_A \phi_{i,\pi_{-i}}(S_i, A_i)\, r(S_i, A_i). \tag{23}$$

Since the average on-time computing completion probability should be maintained above the target on-time computing

$$P[S_i'|S_i, A_i] = P[P_i'|P_i] \times P[C_i'|C_i] \times P[H_i'|H_i] \times P[E_i'|E_i, P_i, A_i]. \tag{7}$$

$$P[E_i'|E_i \neq E^{\max}, P_i, A_i = 0] = \begin{cases} P_E(\lambda_E, k), & \text{if } E_i' = E_i + k \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

$$P[E_i'|E_i = E^{\max}, P_i, A_i = 0] = \begin{cases} 1, & \text{if } E_i' = E_i \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

$$P[E_i'|E_i \geq f_E(P_i), P_i, A_i = 1] = \begin{cases} P_E(\lambda_E, k), & \text{if } E_i' = E_i + k - f_E(P_i) \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

$$P[E_i'|E_i < f_E(P_i), P_i, A_i = 1] = \begin{cases} P_E(\lambda_E, k), & \text{if } E_i' = E_i + k \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

completion probability θ_D, we have

$$\sum_S \sum_A \phi_{i,\pi_{-i}}(S_i, A_i)\, c(S_i, A_i) \geq \theta_D. \tag{24}$$

For the Chapman-Kolmogorov equation [37], we have

$$\sum_A \phi_{i,\pi_{-i}}\left(S_i', A_i\right) = \sum_S \sum_A \phi_{i,\pi_{-i}}(S_i, A_i)\, P[S_i' | S_i, A_i]. \tag{25}$$

The fundamental properties of the probability are constrained by

$$\sum_S \sum_A \phi_{i,\pi_{-i}}\left(S_i{'}, A_i\right) = 1 \tag{26}$$

and

$$\phi_{i,\pi_{-i}}\left(S_i{'}, A_i\right) \geq 0. \tag{27}$$

If a feasible solution exists for the LP problem, then the stationary best response policy of IoT device i is given by

$$\pi_i^*(S_i, A_i) = \frac{\phi^*_{i,\pi_{-i}}(S_i, A_i)}{\sum_{A_i' \in \mathbf{A_i}} \phi^*_{i,\pi_{-i}}\left(S_i, A_i'\right)}. \tag{28}$$

Algorithm 1 Best Response Dynamics-Based Algorithm

```
1: Generate the random policy $\pi_i$ for $\forall i$.
2: repeat
3:   for each IoT device $i$ do
4:     Calculate the probability $\lambda_i$
5:     Transmit the probability $\lambda_i$ to other IoT devices
6:     Calculate the probability $\lambda_A$
7:     Solve the LP problem to obtain the optimal policy $\pi_i^*$
8:   end for
9: until All policies of IoT devices are converged
```

To obtain the optimal policies of the IoT devices, we design a best response dynamics-based algorithm (see Algorithm 1). First, each IoT device generates its policy randomly (line 1 in Algorithm 1). Next, each IoT device i calculates the probability λ_i that it can complete computing within the deadline using (20) and transmits the probability to the other IoT devices (see lines 4-5 in Algorithm 1)[3]. After receiving the probability λ_i from all the IoT devices, each IoT device calculates λ_A that at least one the IoT device can complete computing within the deadline. Each IoT device then solves the LP problem to achieve the optimal policy π_i^* (line 7 in Algorithm 1). Once the policies of all the IoT devices converge, the algorithm is complete.

The LP problem is generally solved with low complexity [38]. For example, Vaidya's algorithm [38], which is one of the traditional methods of solving the LP problem, has a polynomial complexity of $\mathcal{O}(\Omega(\mathbf{S_i}) \cdot \Omega(\mathbf{A_i})^3)$, where $\mathcal{O}(\cdot)$ and $\Omega(\cdot)$ represent the big O notation and number of elements in the given set, respectively. Moreover, since the stationary policies can be attained from a small number of iterations (as shown in Section V-A), we conclude that Algorithm 1 has a manageable overhead and can hence be utilized practically.

[3]Note that $P[A_i = 1]$ can be calculated as $\sum_S \sum_A \phi_{i,\pi_{-i}}(S_i, A_i = 1)$.

TABLE 2. System parameters.

Parameter	E_{max}	C^{max}	M	λ_C	λ_E	γ_{th}
Value	10	10	10	5	5	-4 dB

V. EVALUATION RESULTS

To evaluate the performance of the proposed system, we built a simulation program using MATLAB. We simulated a cooperative IoT based system with $N_I = 4 \sim 8$ IoT devices. We assumed that the initial energies of all IoTs were fully charged, the target on-time computing completion probability was $\theta_D = 0.9$. For the effective transmission rate, we assumed that the transmission rate is zero when γ is under threshold γ_{th}, i.e., $\gamma < \gamma_{th}$. Additionally, $P_T = 1$ and $\sigma^2 = 1$ to observe the effects of the parameters. To obtain the realistic evaluation setup, we have measured the inference latency of VGG16 several times at Raspberry Pi 4B according to the splitting points. For example, the average inference latency using the whole model for all test data is 5.1 sec. Note that, since we measure the inference latency without any other background applications, the measured inference latency can be considered as the inference latency when the IoT device has the maximum computing power. In addition, it is assumed that the inference latency is inversely proportional to the available computing power. Meanwhile, it is assumed that the energy consumption of IoT device (i.e., requestee) is proportional to floating point operations (FLOPS) in each layer. FLOPS in each layer can be measured by python-papi package [39]. The energy consumption for the layer having the smallest FLOPS is normalized as 1. The other default parameter settings are as shown in Table 2.

For the performance evaluation, we compare the proposed system to five schemes: 1) ALWAYS, where the IoT devices always accept the computing request; 2) CE-BASED [29], where the IoT devices perform channel estimation with imperfect channel-state information; 3) RAND, where the IoT devices randomly accept the computing request; 4) P-BASED, where the IoT devices accept a computing request with probability p_A that is set to 0.7; and 5) CoEdge [19], where the IoT devices compute the evenly splitted the workload. For the fair comparison, IoT devices in all comparison schemes operate in the same channel. The average energy consumption and average on-time computing completion probability are used as the performance metrics.

A. CONVERGENCE TO NASH EQUILIBRIUM

Figure 2 describes the process of convergence to the Nash equilibrium. Herein, the initial probabilities of accepting the computing requests of IoT devices 1, 2, 3, and 4 are set to 0.9, 0.5, 0.1, and 0.8, respectively. In addition, the computing power of IoT device 1 is set to 4, whereas those of IoT devices 2, 3, and 4 are all set to 5. As shown in Figure 2, the

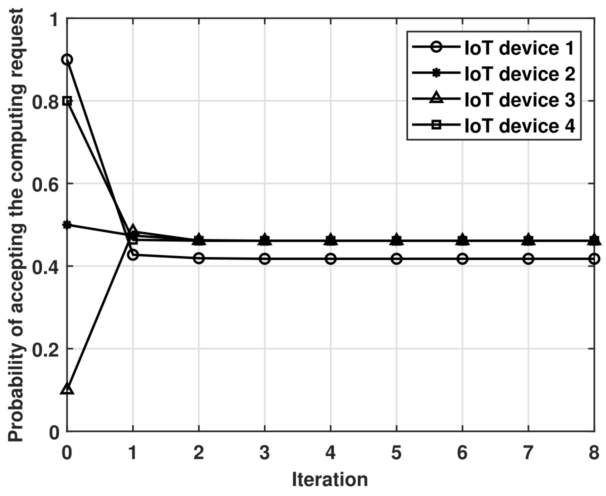
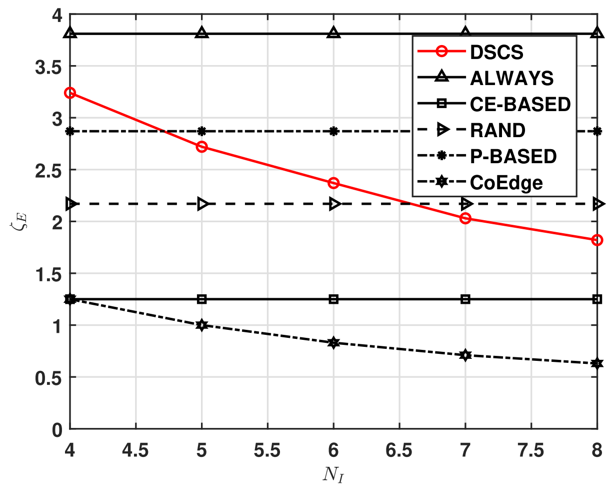
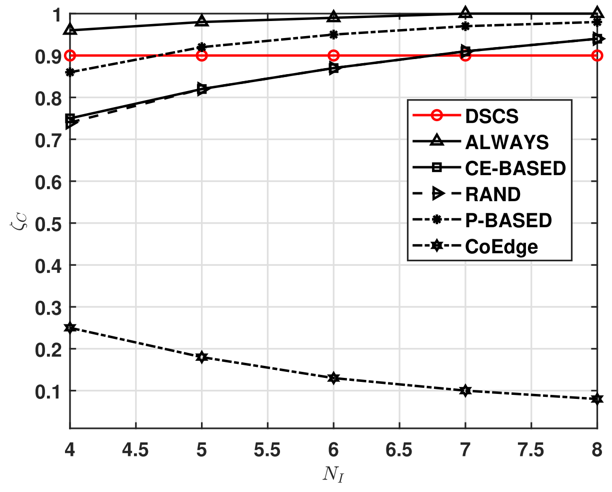


FIGURE 2. Converging process to Nash equilibrium.



(a) Average energy consumption ζ_E.



(b) Average on-time computing completion probability ζ_C.

FIGURE 3. Effect of the number of IoT devices N_I.

policies of IoT devices converge to the Nash equilibrium only after 3 iterations. This indicates that each IoT device needs to transmit its completion probability λ_i to the other devices only 2 times (see line 5 in Algorithm 1)[4].

From Figure 2, it is evident that the action of an IoT device is based on the other IoT devices. For example, in this result, IoT device 1 has a lower computing power than the other devices; hence, its acceptance of the computing request is not helpful for increasing the on-time computing completion probability. In other words, the acceptance of the computing request by IoT device 1 can be considered as unnecessary energy consumption. In this situation, IoT device 1 does not probably accept the request (i.e., it has low probability of accepting the computing request). Considering this fact, the other IoT may devices accept the computing request with a greater probability.

B. EFFECT OF N_I

Figures 3(a) and (b) show the effects of the number of IoT devices N_I on the average energy consumption ζ_E and average on-time computing completion probability ζ_C, respectively. From the figures, it is shown that the DSCS can minimize the average energy consumed while maintaining desirable average on-time computing completion probability (i.e., 0.9). This is because the IoT devices in the DSCS decide the action regarding acceptance of the computing request by considering their operating environments. For example, if an IoT device has high available computing power and channel gain, it accepts the computing request because it can probably complete the computation within the deadline, which also entails avoiding unnecessary energy consumption. On the contrary, if an IoT device has low available computing power and channel gain, it does not accept the computing request to avoid unnecessary energy consumption[5].

From Figure 3(a), it is observed that the average energy consumption of the DSCS decreases with increasing numbers of IoT devices. This is because the IoT devices in the DSCS consider their neighboring devices when deciding whether to accept the computing request. Specifically, in a situation where numerous IoT devices exist, from the perspective of a specific device, it is expected that the task can be completed with high probability within the deadline even though that device does not accept the request. Therefore, each IoT device accepts the computing request with a lower probability to reduce the amount of energy consumed. Meanwhile, since the other comparison schemes (except CoEdge) do not consider the number of IoT devices N_I, their average energy consumption are constant regardless of N_I. Note that, all IoT devices in

[4]Since the cooperative decision can be converged with only 2 iterations as shown in Figure 2, the time required the cooperative decision can be neglected.

[5]Note that, if an IoT device has low available computing power and channel gain, it does not complete computing within the deadline. This implies that, if that IoT device accepts the computing request, it just wastes its energy unnecessarily.

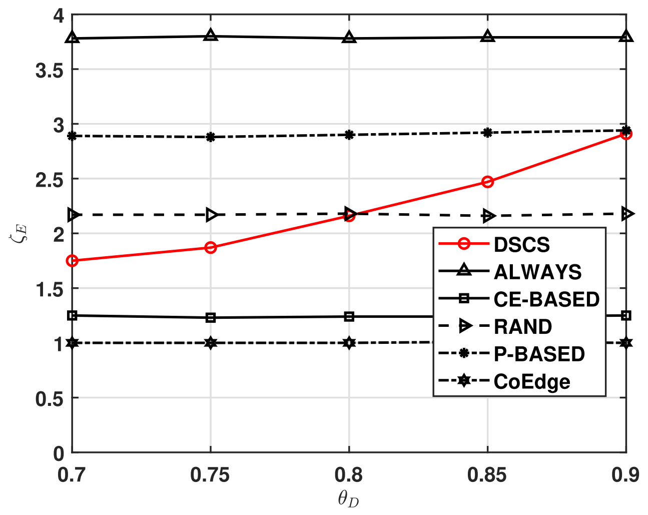


FIGURE 4. Effect of the target on-time computing completion probability θ_D on the average energy consumption ζ_E.

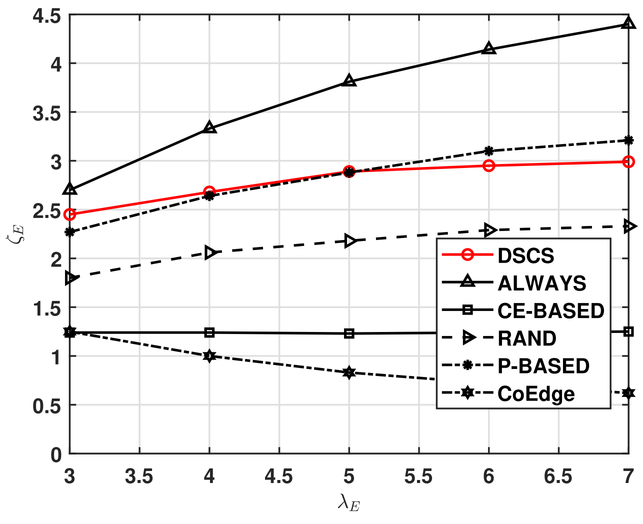
CoEdge compute the evenly splitted the workload, its average energy consumption decreases as N_I increases.

C. EFFECT OF θ_D

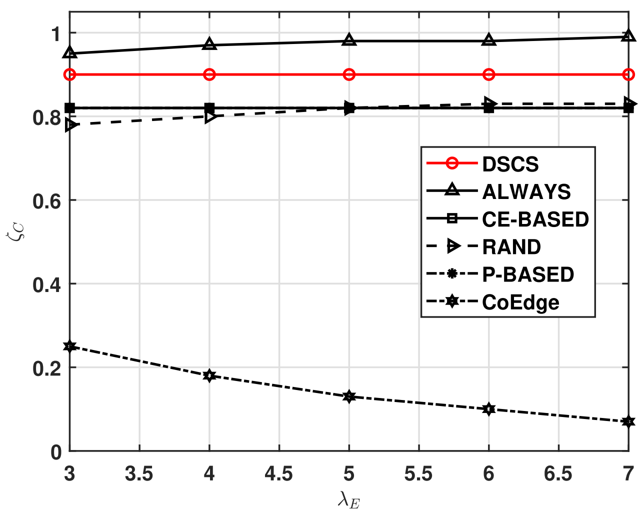
Figure 4 shows the effect of the target on-time computing completion probability θ_D on the average energy consumption ζ_E. From the figure, it is seen that ζ_E of the DSCS increases as θ_D increases, whereas ζ_E of the other schemes are constant regardless of θ_D. This can be explained as follows. As the number of IoT devices accepting the computing request increases, the probability that at least one of them can complete the computation within the deadline also increases. The IoT devices in the DSCS recognize this fact and accept a computing request aggressively when a higher θ_D is given. However, the other schemes compared herein do not consider the target on-time computing completion probability, so they do not change their policies.

D. EFFECT OF λ_E

Figures 5(a) and (b) demonstrate the effects of the average energy units harvested λ_E on the average energy consumed ζ_E and average on-time computing completion probabilities ζ_C, respectively. Interestingly, from Figure 5(a), it is seen that the average energy consumption ζ_E of all schemes except CE-BASED increase logarithmically with increase in λ_E. This can be explained as follows. If an IoT device does not have sufficient energy E ($< f_E(P)$), it cannot infer the tail model and does not consume any energy. The probability of occurrence of this situation decreases as λ_E increases. Thus, with the increase in λ_E, the average on-time computing completion probabilities ζ_C of all the comparison schemes also increase, as shown in Figure 5(b). However, the DSCS does not consume the additional energy to increase the average on-time computing completion probability in excess above a certain level (i.e., 0.9 herein); thus, its average on-time computing completion probability remains unchanged irrespective of λ_E, as shown in Figure 5(b).



(a) Average energy consumption ζ_E.



(b) Average on-time computing completion probability ζ_C.

FIGURE 5. Effect of λ_E.

E. EFFECT OF ρ

Figure 6 demonstrates the effect of the average channel gain ρ on the average energy consumption ζ_E. As observed in the figure, ζ_E of the DSCS decreases as ρ increases. This is because the intermediate data can be delivered with a lower latency when ρ is higher, which indicates a higher probability of completing the computation within the deadline. Each IoT device in the DSCS recognizes this condition and does not accept the computing request so as to reduce the amount of energy consumed. However, since the other schemes except CE-BASED do not recognize this condition, their average amounts of energy consumed remain unchanged irrespective of ρ.

Meanwhile, in CE-BASED, the decision to accept the computing request is based on the channel condition (i.e., the IoT devices accept the request when the channel gain is high). Thus, its average energy consumption ζ_E increases as the average channel gain increases.

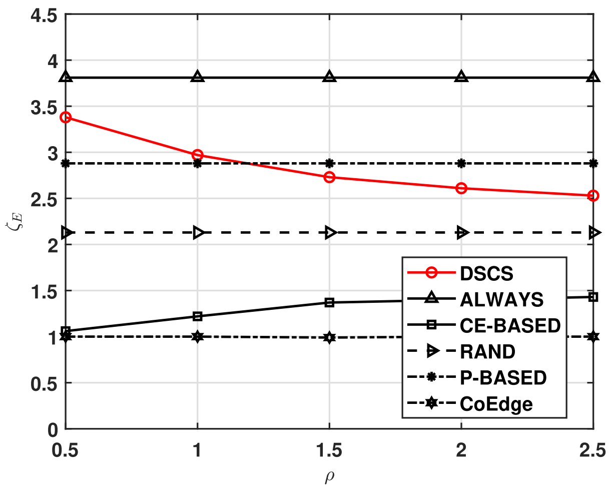


FIGURE 6. Effect of the average channel gain ρ on the average energy consumption ζ_E.

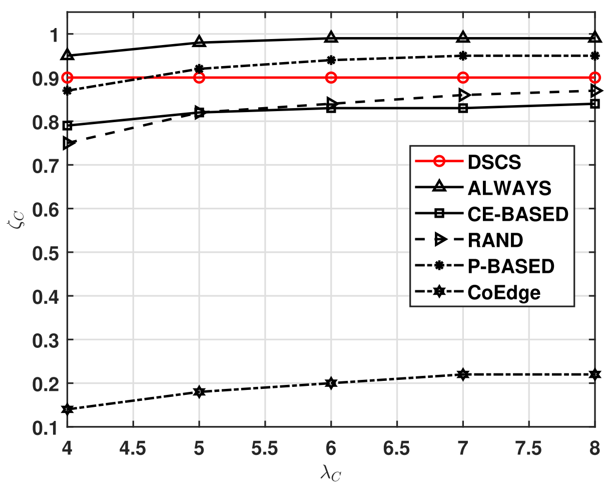


FIGURE 7. Effect of the average computing power λ_C on the average on-time computing completion probability ζ_C.

F. EFFECT OF λ_C

Figure 7 demonstrates the effect of the average computing power λ_C on the average on-time computing completion probability ζ_C. Intuitively, when an IoT device has higher computing power, it can complete inference of the given tail model within a shorter duration. Therefore, as shown in Figure 7, the average on-time computing completion probabilities ζ_C of the other comparison schemes increase with increase in average computing power λ_C. However, in the DSCS, if the IoT devices have more computing power, they reduce their acceptance probabilities for the computing request to reduce the amount of energy consumed. Thus, the average on-time computing completion probability of the DSCS is maintained at a specific level (i.e., 0.9).

G. EFFECT OF D

Figure 8 demonstrates the effect of the deadline D on the average energy consumption ζ_E. When the deadline is farther, even though the IoT devices accept the computing request with a lower probability, the desired on-time computing completion probability can be achieved. In the DSCS, the IoT devices are aware of this fact and can thus reduce energy consumption when D is large, as demonstrated in Figure 8. However, the other comparison schemes do not alter their policies according to the deadline D, so their amounts of energy consumed remain constant.

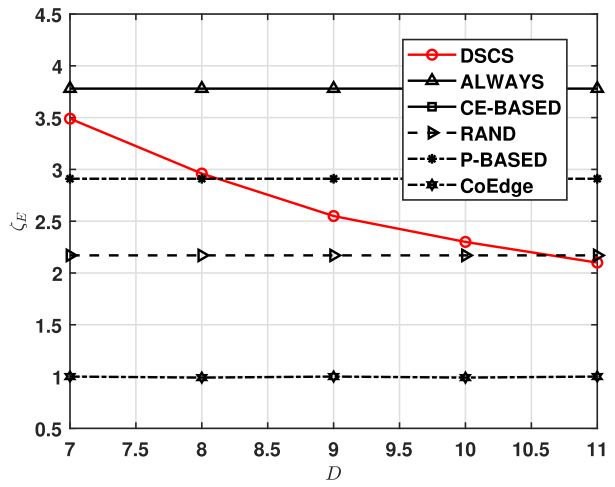


FIGURE 8. Effect of the deadline D on the average energy consumption ζ_E.

VI. CONCLUSION

In this work, we introduce a distributed split computing system (DSCS) wherein the IoT devices (i.e., requestees) distributively determine acceptance of a split computing request from a specific IoT device by considering the unnecessary energy consumption and computation completion time. For performance optimization, a constrained stochastic game model is developed, and a multipolicy constrained Nash equilibrium is attained using a best-response dynamics-based algorithm. The evaluation results show that the DSCS significantly reduces the amounts of energy consumed by the IoT devices while providing high on-time computing completion probability. Moreover, it is noted that the IoT devices in the DSCS adaptively adjust their actions by considering their neighbors' actions and operating environments. In the future, we plan to expand the proposed system to also account for multitask learning.

REFERENCES

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[2] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–30, May 2023.

[3] Y. Matsubara and M. Levorato, "Split computing for complex object detectors: Challenges and preliminary results," in *Proc. 4th Int. Workshop Embedded Mobile Deep Learn.*, Sep. 2020, pp. 7–12.

[4] H. Ko, J. Lee, and S. Pack, "Spatial and temporal computation offloading decision algorithm in edge cloud-enabled heterogeneous networks," *IEEE Access*, vol. 6, pp. 18920–18932, 2018.

[5] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[6] Y. Kang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. ASPLOS*, Apr. 2017, pp. 1–15.
[7] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, Aug. 2018, pp. 31–36.
[8] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "SPINN: Synergistic progressive inference of neural networks over device and cloud," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2020, pp. 1–15.
[9] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Energy-efficient model compression and splitting for collaborative inference over time-varying channels," in *Proc. IEEE 32nd Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2021, pp. 1173–1178.
[10] J. Yan, S. Bi, and Y.-J.-A. Zhang, "Optimal model placement and online model splitting for device-edge co-inference," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8354–8367, Oct. 2022.
[11] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 565–576, Feb. 2021.
[12] H. Zhou, "BBNet: A novel convolutional neural network structure in edge-cloud collaborative inference," *Sensors*, vol. 21, no. 13, pp. 1–16, Jun. 2021.
[13] W. He, S. Guo, S. Guo, X. Qiu, and F. Qi, "Joint DNN partition deployment and resource allocation for delay-sensitive deep learning inference in IoT," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9241–9254, Oct. 2020.
[14] X. Tang, X. Chen, L. Zeng, S. Yu, and L. Chen, "Joint multiuser DNN partitioning and computational resource allocation for collaborative edge intelligence," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9511–9522, Jun. 2021.
[15] S. Wang and X. Zhang, "NeuroMessenger: Towards error tolerant distributed machine learning over edge networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 2058–2067.
[16] S. Wang, X. Zhang, H. Uchiyama, and H. Matsuda, "HiveMind: Towards cellular native machine learning model splitting," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 626–640, Feb. 2022.
[17] Y. Matsubara, R. Yang, M. Levorato, and S. Mandt, "SC2 benchmark: Supervised compression for split computing," 2022, *arXiv:2203.08875*.
[18] H. Ahn, M. Lee, C.-H. Hong, and B. Varghese, "ScissionLite: Accelerating distributed deep neural networks using transfer layer," 2021, *arXiv:2105.02019*.
[19] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2021.
[20] E. Samikwa, A. D. Maio, and T. Braun, "Adaptive early exit of computation for energy-efficient and low-latency machine learning over IoT networks," in *Proc. IEEE 19th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2022, pp. 200–206.
[21] S. Zhang, S. Zhang, Z. Qian, J. Wu, Y. Jin, and S. Lu, "DeepSlicing: Collaborative and adaptive CNN inference with low latency," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 9, pp. 2175–2187, Sep. 2021.
[22] A. Bakhtiarnia, N. Miloševic, Q. Zhang, D. Bajovic, and A. Iosifidis, "Dynamic split computing for efficient deep EDGE intelligence," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.
[23] F. Cunico, L. Capogrosso, F. Setti, D. Carra, F. Fummi, and M. Cristani, "I-SPLIT: Deep network interpretability for split computing," in *Proc. 26th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2022, pp. 2575–2581.
[24] T. A. Khoa, D.-V. Nguyen, M.-S. Dao, and K. Zettsu, "SplitDyn: Federated split neural network for distributed edge AI applications," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2022, pp. 6066–6073.
[25] *LoRa Alliance, LoRaWAN L2 1.0.4 Specification*. Accessed: Jul. 26, 2023. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-104-specification-package/
[26] A. Lavric and V. Popa, "Internet of Things and LoRa low-power wide-area networks: A survey," in *Proc. Int. Symp. Signals, Circuits Syst. (ISSCS)*, Jul. 2017, pp. 1–5.
[27] E. Altman, K. Avrachenkov, N. Bonneau, M. Debbah, R. El-Azouzi, and D. S. Menasche, "Constrained cost-coupled stochastic games with independent state processes," *Oper. Res. Lett.*, vol. 36, no. 2, pp. 160–164, Mar. 2008.
[28] H. Ko and S. Pack, "Function-aware resource management framework for serverless edge computing," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1310–1319, Jan. 2023.
[29] R. Xie, Q. Tang, C. Liang, F. R. Yu, and T. Huang, "Dynamic computation offloading in IoT fog systems with imperfect channel-state information: A POMDP approach," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 345–356, Jan. 2021.
[30] X. Wang, J. Gong, C. Hu, S. Zhou, and Z. Niu, "Optimal power allocation on discrete energy harvesting model," *EURASIP J. Wireless Commun. Netw.*, vol. 2015, no. 1, pp. 1–14, Dec. 2015.
[31] C. Liu, W.-X. Meng, C. Li, and M. Ma, "Active computing toward 5G Internet of Things," *IEEE Wireless Commun.*, vol. 29, no. 2, pp. 124–130, Apr. 2022.
[32] Y. Yuan, W. Yi, W. Choi, and L. Kong, "Dynamic quantizer design for target tracking for wireless sensor network with imperfect channels," *IEEE Trans. Wireless Commun.*, vol. 22, no. 3, pp. 1695–1711, Mar. 2023.
[33] *Study on Traffic Characteristics and Performance Requirements for AI/ML Model Transfer (Release 18)*, Standard 3GPP TR 22.874, Dec. 2021.
[34] D. Mugerwa, Y. Nam, H. Choi, Y. Shin, and E. Lee, "Implicit overhearing node-based multi-hop communication scheme in IoT LoRa networks," *Sensors*, vol. 23, no. 8, p. 3874, Apr. 2023.
[35] E. F. Silva, L. M. Figueiredo, L. A. de Oliveira, L. J. Chaves, A. L. de Oliveira, D. Rosário, and E. Cerqueira, "Adaptive parameters for LoRa-based networks physical-layer," *Sensors*, vol. 23, no. 10, p. 4597, May 2023.
[36] E. Solan, "Three-player absorbing games," *Math. Oper. Res.*, vol. 24, no. 3, p. 669 698. Aug. 1999.
[37] J. Karush, "On the Chapman–Kolmogorov equation," *Ann. Math. Statist.*, vol. 32, no. 4, pp. 1333–1337, 1961.
[38] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*. New York, NY, USA: Springer-Verlag, 1997.
[39] *PyPAPI*. Accessed: Jul. 26, 2023. [Online]. Available: https://flozz.github.io/pypapi/

SEUNG-YEON KIM received the B.S., M.S., and Ph.D. degrees in electronics and information engineering from Korea University, Sejong, South Korea, in 2005, 2007, and 2012, respectively. From 2012 to 2013, he was a Research Professor with the Institute of Natural Science, Korea University, where he was a Research Professor with the Department of Computer and Information Science, from 2013 to 2015. Currently, he is an Associate Professor with the Department of Computer Convergence Software, Korea University. His research interests include 6G networks, network automation, mobile cloud computing, federated learning, and the Internet of Things.

HANEUL KO (Senior Member, IEEE) received the B.S. and Ph.D. degrees from the School of Electrical Engineering, Korea University, Seoul, South Korea, in 2011 and 2016, respectively. He is currently an Assistant Professor with the Department of Electronic Engineering, Kyung Hee University, Yongin-si, South Korea. From 2019 to 2022, he was an Assistant Professor with the Department of Computer and Information Science, Korea University. From 2017 to 2018, he was a Postdoctoral Fellow with the University of British Columbia, Vancouver, BC, Canada. From 2016 to 2017, he was a Postdoctoral Fellow in mobile network and communications with Korea University. His research interests include 5G/6G networks, network automation, mobile cloud computing, SDN/NFV, and future internet. He was the recipient of the Minister of Education Award, in 2019, and IEEE ComSoc APB Outstanding Young Researcher Award, in 2022.