Risk-Aware Cloud-Edge Computing Framework for Delay-Sensitive Industrial IoTs

Yi Zhang, *Member, IEEE,* and Hung-Yu Wei, *Senior Member, IEEE*

***Abstract*—The industrial internet of things (IIoT) has been widely deployed to provide autonomous inspection on current production status and quality of products for modern manufacturing. However, the IIoT sensors generally are short of computing capabilities and therefore could not offer acceptable latency for computation-intensive inspection tasks. Besides, the mission-critical industrial applications are extremely sensitive to inspection failure, which may lead to serious manufacturing problems or accidents. In this paper, we propose a risk-aware cloud-edge computing framework for the delay-sensitive inspections of autonomous manufacturing. Due to the uncertainty of 802.11ax, we utilize the conditional value-at-risk (CVaR) to measure the inspection risk basing on the distribution of channel access delay. We develop a branch-and-check (BNC) approach to optimally and efficiently deploy the decomposable inspection tasks with the minimum operation cost and acceptable latency. The extensive simulations guide the operational use for future IIoT and the results show that the proposed system can save a large amount of unnecessary operation cost by enabling the processor sharing strategy.**

***Index Terms*—Industrial IoT, edge computing, delay-sensitive, conditional value-at-risk**

I. Introduction

With the evolution of Industry 4.0, factory automation becomes one of the most important concepts to improve production efficiency and the economic benefits [1]. To meet the trend of modern autonomous manufacturing, more and more sensors deployed within the smart factory form an *industrial internet of things (IIoT)* to collect data, which is utilized to give recommendations and feedback on current production status and quality of products [2]. Note that various industrial applications require reliable and efficient inspections, such as package inspection, automated surface defect detection, predictive maintenance, and so on. However, the IIoT sensor generally is empowered with limited computing capabilities so that it could not perform advanced computer vision for autonomous inspections, especially when the powerful deep learning-based techniques are applied. Moreover, autonomous inspection in industrial scenarios is usually time-sensitive, which means that even a little inspection failure will lead to serious manufacturing problems or accidents.

In recent years, *edge computing* has emerged as a promising solution to process the computation-intensive tasks offloaded by IIoT [3]. Compared with the traditional cloud computing, edge computing provides sufficient computing capabilities closer to the IIoT devices and therefore avoids the long-distance transmission delay to the cloud [4, 5]. In addition, the mission-critical industrial applications are extremely sensitive to the failure caused by the overtime of inspection tasks. Note that an offloaded task may be under the risk due to the uncertainty of wireless environments, such as varying wireless channel conditions [6], the unknown channel estimation errors [7], the contention-based communication protocol [8], and so on. Therefore, it is necessary to focus on the inspection risk of the IIoT system to ensure the reliability of the edge computing-based offloading services.

The 5G NR on unlicensed bands (NR-U) has been regarded as a candidate communication technology for the future smart factory [9], however, the standardization is still underway. Nevertheless, the IEEE 802.11ax (WiFi 6) formally published in 2019 has delivered significant performance improvements in indoor communications and therefore it is recognized as a complementary technology to the industrial applications. Different from the upcoming 5G NR-U, the 802.11ax can be quickly deployed without any interaction with the network operator. To the best of our knowledge, relatively little work discusses the impact of the 802.11-based protocols on the delay-sensitive industrial applications.

Based on the above discussions, we propose a risk-aware cloud-edge computing framework for the delay-sensitive inspections of autonomous manufacturing executed by IIoT-based vision sensors. The edge computing cluster provides local processing for latency reduction and the remote cloud center provides fast computation service. Specifically, we adopt 802.11ax as the wireless communication protocol within the smart factory to provide connectivity to the sensors. We assume that the delay-sensitive inspection tasks are decomposable so that each of them can be partitioned into two sub-tasks and processed sequentially. Besides, the processor sharing strategy is allowed to the local processors at the edge to cut down the buying of unnecessary processors. By leveraging the power of the cloud-edge computing architecture, the owner of the factory aims to optimally deploy the inspection tasks over a long duration of running with the minimum total operation cost and acceptable latency. Our main contributions are as follows:

- We propose a risk-aware cloud-edge computing framework for the delay-sensitive inspections of autonomous manufacturing. To tackle the uncertainty of the channel access delay caused by the 802.11ax, we model its delay distribution in terms of mean and variance. We utilize the conditional value-at-risk (CVaR), which has been extensively used in

Corresponding author: Hung-Yu Wei

Yi Zhang is with the Department of Information and Communication Engineering, Xiamen University, China (email: yizhang@xmu.edu.cn)

Hung-Yu Wei is with the Graduate Institute of Communication Engineering, Graduate Institute of Electrical Engineering, and Department of Electrical Engineering, National Taiwan University, Taiwan (email: hywei@cc.ee.ntu.edu.tw)

portfolio optimization for effective risk management [10], to measure the inspection risk to guarantee the reliability of the offloading services.

- We formulate a cost minimization problem for optimal offloading service deployment subject to the latency requirements and reliability threshold. We develop a branch-and-check (BNC) approach to solve this mixed-integer nonlinear programming problem (MINLP) optimally and efficiently. Compared with the widely used branch-and-bound (BNB) method, the proposed BNC approach greatly reduces the number of feasibility checks to the worst-case CVaR constraint and therefore saves a large amount of computation time.
- Through extensive simulations, we investigate how the number of associated sensors and random access resource units (RA-RUs) to each access point (AP) determines the distribution of channel access delay and further impacts the optimal offloading service deployments. We also discuss how the operation cost and the offloading service deployments are affected by the operation duration, the reliability threshold, and the number of inspection sensors, which is valuable for operational use in industrial applications. The results show that the proposed approach takes full advantage of the cloud-edge computing architecture and the processor sharing strategy to save the unnecessary operation cost.

The rest of this paper is organized as follows. We summarize related works in Section II. The proposed cloud-edge computing framework is presented in Section III and a cost minimization problem is formulated in Section IV. To make the problem solvable, we develop an effective BNC approach in Section V. We provide extensive evaluation results in Section VI and, finally, Section VII concludes this paper.

II. Related work

The computation offloading of edge computing has been widely studied to process computation-intensive IIoT tasks. A dynamic computation offloading scheme [11], which aims to minimize energy consumption in edge computing scenarios, is proposed for IIoT by jointly optimizing the offloading ratio, transmission power, local CPU computation speed, and transmission time. The decomposable intelligence is proposed on a cloud-edge IoT framework [12] to support joint latency- and accuracy- aware live video analytics. Specifically, it enables pipeline sharing to reduce unnecessary resource costs at the edge. Considering the rapid growth of IIoT devices in mobile edge computing (MEC), a multi-agent deep reinforcement learning scheme [13] is provided to increase the success rate of multi-channel access and meanwhile reduce the computation delay of task offloading.

In the research field of risk-aware edge computing, some research works try to overcome the uncertainty of wireless environments. To minimize the total offloading delays of a heterogeneous edge computing architecture, a power allocation problem [6] is formulated to achieve reliable communication, which is constrained by the worst-case channel condition. The error of channel estimation is taken into consideration in [7] and an energy-efficient offloading strategy is provided to ensure the robustness of edge computing service subject to the latency requirement. To maximize the long-term rewards of the 802.11p-based vehicular fog computing system, an optimal task offloading scheme [8] is proposed by jointly considering the transmission delay, the computing delay, and the available resources. So far, the uncertainty of the 802.11-based protocol is rarely discussed for autonomous manufacturing and our proposed risk-aware framework can guarantee the reliability of the offloading services.

Due to the service demand uncertainty, a two-stage risk-averse optimization, including edge resource rental and workload assignment, is developed to maximize the deployment profits of the service provider [14]. To migrate the risk of overload, several confidence sets are constructed under the worse-case demand distribution by exploiting historical service demand traces. The service risk probability (SRP) of virtual machine (VM) fails is defined and considered in a distributed cloud/edge computing system [15]. The existence of the optimal SRP is theoretically proved for minimum energy cost. In a vehicular edge computing network, entropic risk measure is leveraged to formulated a risk minimization problem and a distributed no-regret learning approach is proposed for task fetching and offloading [16]. A reliable-aware computation offloading is proposed to maximize the number of requests in a UAV-enable edge computing system [17], where the reliability requirement is modeled as the condition that the achievable offloading rate should be greater than the requests. However, it assumes that the task could be arbitrarily partitioned into multiple subtasks, which may not be applicable for all real-world tasks. The decomposable task applied in our model is more practical and more general, especially for those inspection tasks in IIoT environments. Note that the shared nature of computation and communication resources of a multi-MEC servers system may introduce high uncertainly in terms of limited computation capability and wireless interference, a con-cooperative game is formulated among users with risk-seeking or loss-aversion behaviors [18]. Specifically, Prospect Theory (PT) is adopted to formulate the utility function for data offloading decisions.

III. System Model

A. System Architecture

The proposed cloud-edge architecture for autonomous manufacturing inspection is shown in Fig. 1. We consider a smart factory, where a set of IIoT-based vision sensors $\mathcal{N} = \{1, \ldots, |\mathcal{N}|\}$, are deployed around the manufacturing lines and have enough energy to capture the visual information of the productions. Through specific recognition processes, the analytic results could be utilized to find the possible defects of products and feedback on the status of the productions. We assume a set of *computation-intensive and time-sensitive inspection tasks* $\mathcal{F} = \{1, \ldots, |\mathcal{F}|\}$ are processed in this factory, such as printed circuit board (PCB) defect detection, chip quality control, and liquid crystal display (LCD) surface defect inspection. Each sensor plays one type of inspection tasks from $\mathcal{F}$ and it is given in advance. Those sensors with the specific task $f \in \mathcal{F}$ form a set $\mathcal{N}_f$. Specifically,

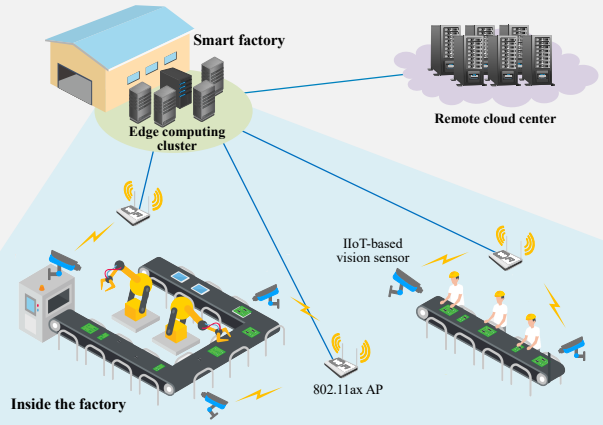


Fig. 1. Cloud-edge architecture for autonomous manufacturing inspections

some manufacturing requirements are predefined to ensure the *production efficiency* of manufacturing lines and *outgoing quality* of goods. The production efficiency is reflected by the required frame sampling rate of each inspection task operated in different manufacturing lines in order to meet production capacities and the outgoing quality is sensitive to the inspection failure caused by violating diverse delay and accuracy tolerances of inspection tasks. It should be noted that the accuracy requirement has been reflected by the required computational resource of different types of inspection tasks and therefore the accuracy control is beyond the scope of this paper. We define that each sensor $i \in \mathcal{N}$ should be configured at particular frame sampling rate λ_i and latency requirement L_i^{req}. We assume that the latency requirement is less than the interval of two contiguous frames, which means that the current frame should be dropped when the next frame is generated. As the owner of the smart factory, the goal is to improve the economic benefits in term of the operation cost with those predefined manufacturing requirements above.

Since the sensors are generally short of computing capabilities, an edge computing cluster consisting of a controller and multiple processors is deployed and managed by the smart factory at the local place to provide *offloading services*. Let $\mathcal{K}_f$ represent the set of processors allocated to execute the inspection task of type $f \in \mathcal{F}$. Alternatively, an inspection task can also be processed at the remote cloud center, which is more powerful than the edge cluster in computing capability and therefore can provide faster computation service. Besides, offloading to the cloud means the smart factory can cut down a part of operation cost of buying local processors. We assume that all inspection tasks are *decomposable*, that is, any inspection task in $\mathcal{F}$ can be partitioned into two sub-tasks and then executed sequentially [12]. For example, a decomposable task for PCB defect detection finds all the components from the PCB board and further identifies the defects of each component. It should be noted that all decomposable tasks are known in advance by the factory operator and fixed during the whole operation duration. By taking full advantage of the cloud-edge computing architecture, there are three feasible schemes $\Phi = \{\text{eo}, \text{co}, \text{ce}\}$ for the offloading service placement: *1) Edge-Only (EO)*: the task is completely executed at the edge; *2) Cloud-Only (CO)*: the task is completely executed at the cloud; *3) Cloud-Edge (CE)*: the decomposable task is executed at the edge and the cloud separately. We introduce a binary variable $x_{ik}^{\phi} \in \{0, 1\}$ to indicate the **service placement decision** for sensor $i \in \mathcal{N}_f$ with $k \in \mathcal{K}_f$ and $\phi \in \Phi$. When $x_{ik}^{\phi} = 1$, the task of sensor i is executed at processor k with scheme ϕ. To enable wireless connectivity to the sensors, multiple 802.11ax APs are placed inside the factory as shown in Fig. 1. We assume that the basic service set (BSS) coloring technology has been utilized to decrease co-channel interference among multiple APs and the sensor-AP association problem is beyond the scope of this paper. The key notations of this paper are listed in Table I.

TABLE I
LIST OF KEY NOTATION

Notation	Definition
$\mathcal{N}$	Set of IIoT-based vision sensors, $\forall i \in \mathcal{N}$
$\mathcal{F}$	Set of inspection tasks, $\forall f \in \mathcal{F}$
$\mathcal{N}_f$	Set of vision sensors with task f
$\mathcal{K}_f$	Set of processors allocated to execute task f
λ_i	Frame sampling rate of sensor i
L_i^{req}	Latency requirement of sensor i
Φ	Set of service placement schemes, $\forall \phi \in \Phi$, where $\Phi = \{\text{eo}, \text{co}, \text{ce}\}$
N	# of associated sensors of a specific AP
N_{r}	# of RA-RUs for UORA contention process
m	Maximum back-off stage
R	Retransmission limit
W	Minimum size of OCW
W_j	Size of OCW for back-off state j
τ	Prob. that a sensor transmits in any RA-RUs
p	Prob. that a packet encounters a collision
T_{c}	Time duration of a TF cycle taken in a collision
T_{P}	Longest time in the TF cycle consumed for a packet transmission
$E[\text{slot}]$	Average length of a slot time
L_i	Service latency of sensor i, which consists of channel access delay $D_i^{(\text{a})}$, transmission delay $D_i^{(\text{t})}$, and computing delay $D_i^{(\text{c})}$
$E[D^{(\text{a})}]$	Mean of channel access delay $D_i^{(\text{a})}$
$Var[D^{(\text{a})}]$	Variance of channel access delay $D_i^{(\text{a})}$
B	Total available bandwidth
r_i	Data rate from sensor i to its connecting AP
r_{cld}	Link capacity between the edge and the cloud
t^{prog}	Propagation delay between the edge and the cloud
b_f	Data size of a uploaded frame with task f
b_f^{ce}	Data size of scheme CE delivered from the edge to the cloud
$t_{f\phi}^{\text{edge}}$	Processing time of a frame at the edge
$t_{f\phi}^{\text{cld}}$	Processing time of a frame at the cloud
x_{ik}^{ϕ}	Service placement decision of whether the task of sensor i is executed at processor k with scheme ϕ
y_k^{ϕ}	Processor deployment decision of whether a local processor k is bought and deployed by scheme ϕ
$1 - \epsilon$	Reliability threshold
C^{depl}	Total deployment cost
C_i^{run}	Per frame running cost of sensor i
c	Unit price of a processor
$e_{f\phi}^{\text{edge}}$	Power consumption at the edge when operating inspection task of type f with scheme ϕ
γ	Coefficient converting power consumption to cost
π^{cld}	service fee per unit time of the cloud

B. Channel Access Model

The IEEE 802.11ax standard provides prominent features for uplink transmission compared with legacy 802.11 stan-

$$\tau = \frac{2(1-2p)(1-p^{R+1})}{\frac{W}{N_\mathrm{r}}(1-(2p)^{m+1})(1-p)+(1-2p)(1-p^{R+1})+\frac{W}{N_\mathrm{r}}2^m p^{m+1}(1-2p)(1-p^{R-m})}, \quad \forall R > m \tag{2}$$

dards [19]: *1)* it enables Orthogonal Frequency Division Multiple Access (OFDMA) based concurrent multi-user (MU) transmission through multiple orthogonal sub-channels called *random access resource units (RA-RUs)*; *2)* AP periodically announces random access information, e.g., the number of RA-RUs, as well as time synchronization for uplink transmission in *Trigger Frame (TF)*; *3)* it supports trigger-based *Uplink OFDMA Random Access (UORA)* as contention mechanism.

In the proposed system, we follow the fundamental assumptions of Bianchi's two-dimensional Markov model [20], that is, ideal channel conditions (i.e., no hidden terminals and capture) and saturation throughput (i.e., always nonempty transmission queue of each sensor). For a specific AP, the number of associated sensors is denoted by N and total N_r RA-RUs are allocated for UORA contention process. We have the size of OFDMA contention window (OCW) for back-off state j as

$$W_j = \begin{cases} 2^j W & \forall 0 \le j \le m \\ 2^m W & \forall m \le j \le R \end{cases}, \tag{1}$$

where W is the minimum size of OCW, m is the maximum back-off stage, and R is the retransmission limit. We know that a RA-RU is randomly chosen to transmit when the OFDMA back-off counter (OBO) decreases to zero. Combining the derivations from [21–23], the probability that a sensor transmits in any N_r RA-RUs can be calculated by (2) with $R \ge m$. The probability that a transmitted packet encounters a collision is presented as follows,

$$p = 1 - \left(1 - \frac{\tau}{N_\mathrm{r}}\right)^{N-1}. \tag{3}$$

For given values of N, N_r, W, m and R, equations (2)(3) can be solved using numerical methods and an unique solution to the probabilities τ and p will be generated.

With the known τ and p, we can calculate the probability that at least one sensor transmits by

$$P_\mathrm{tr} = 1 - \left(1 - \frac{\tau}{N_\mathrm{r}}\right)^{N}. \tag{4}$$

Therefore, the probability that all RA-RUs are in idle will be

$$P_\mathrm{idle} = \left(1 - P_\mathrm{tr}\right)^{N_\mathrm{r}}. \tag{5}$$

Basing on the definition of TF cycle, the time duration of a TF cycle taken in idle condition (i.e., all RA-RUs are in idle) is represented as

$$T_\mathrm{idle} = T_\mathrm{H} + (T_\mathrm{TF} + \mathrm{SIFS} + T_\delta), \tag{6}$$

where T_H, T_TF, SIFS and T_δ are the time taken to transmit header frame, TF frame, Short Interframe Space (SIFS) frame and propagation delay, respectively. Similarly, the time duration of a TF cycle taken in busy condition (i.e., at least one RA-RU is occupied) consisting of either collision T_c or successful transmission T_s is written as

$$\begin{aligned} T_\mathrm{c} = T_\mathrm{s} = & T_\mathrm{H} + (T_\mathrm{TF} + \mathrm{SIFS} + T_\delta) + (T_\mathrm{P} + \mathrm{SIFS} + T_\delta) \\ & + (T_\mathrm{ACK} + \mathrm{SIFS} + T_\delta), \end{aligned} \tag{7}$$

where T_ACK are the time taken to transmit ACK frame. Specifically, T_P is the longest time consumed by the N sensors. In this way, average length of a slot time can be calculated by

$$E[\mathrm{slot}] = P_\mathrm{idle} T_\mathrm{idle} + (1 - P_\mathrm{idle}) T_\mathrm{s}. \tag{8}$$

IV. Problem Formulation

As the owner of the proposed smart factory, the goal is to minimize the total operation cost over a long duration of running subject to the predefined manufacturing requirements reflected by frame sampling rates and delay requirements of inspection tasks. As we know that those decomposable inspection tasks are deployed and served by the cloud-edge computing architecture. However, the latency of offloading service is indeterminate due to the UORA contention mechanism of 802.11ax. To guarantee the required outgoing quality, we introduce CVaR as the measurement of the inspection risk. A cost minimization problem is formulated to save the total operation cost and meanwhile fit the latency requirements.

A. Service Latency Analysis

For sensor i, we have a comprehensive consideration on the components of its service latency, which is written as

$$L_i = D_i^{(\mathrm{a})} + D_i^{(\mathrm{t})} + D_i^{(\mathrm{c})}. \tag{9}$$

Specifically, the channel access delay $D_i^{(\mathrm{a})}$ is the sum of durations of collisions and back-offs. The transmission delay $D_i^{(\mathrm{t})}$ is the time interval of a complete data transmission from sensor i to its connecting AP after successfully occupying the channel plus the round-trip data forwarding between the edge and the cloud. The computing delay $D_i^{(\mathrm{c})}$ is the total processing time of offloading services in the cloud-edge architecture.

1) Channel Access Delay: Due to the random access process of 802.11ax, i.e., several back-offs and retransmissions, we cannot calculate the exact delay of 802.11ax channel access. Alternatively, we measure its delay distribution in terms of mean $E[D^{(\mathrm{a})}]$ and variance $Var[D^{(\mathrm{a})}]$ by modifying the mathematical models of [24]. The key approximation of our model is that we assume an image frame from any inspection task can be completely delivered in one transmission.

We define S_j as the sum of delays that a packet has experienced from back-off stage 0 to stage j ($0 \le j \le R$) and finally accesses the channel without any collision. Similarly, U_j is defined as the sum of delays with unsuccessful channel access at stage j. Then we have the average delay at back-off stage j as follows:

$$E[S_j] = jT_\mathrm{c} + E[\mathrm{slot}] \sum_{j'=0}^{j} \Big(\frac{1}{W_{j'}} \sum_{w=0}^{W_{j'}-1} \lceil \frac{w}{N_\mathrm{r}} \rceil \Big), \tag{10}$$

$$E[U_j] = (j+1)T_\mathrm{c} + E[\mathrm{slot}] \sum_{j'=0}^{j} \Big(\frac{1}{W_{j'}} \sum_{w=0}^{W_{j'}-1} \lceil \frac{w}{N_\mathrm{r}} \rceil \Big), \tag{11}$$

where $T_{\rm c}$ is the time duration of a TF cycle taken in a collision condition, $E[\text{slot}]$ is the average length of a slot time, and $W_{j'}$ is the size of OFDMA contention window (OCW) for stage j'. It should be noted that the OFDMA back-off counter (OBO) is decremented by $N_{\rm r}$ in 802.11ax and the packet is transmitted in a randomly selected RA-RU when the OBO decreases to zero. In this way, the average number of back-offs at stage j' will be $(\sum_{w=0}^{W_{j'}-1}\lceil w/N_{\rm r}\rceil)/W_{j'}$, where w is a selected back-off value. It also shows that $E[U_j]$ has more delay than $E[S_j]$, i.e., $T_{\rm c}$, because of collision. Since the UORA contention behavior follows a truncated geometric distribution [25], the probability of successful channel access at stage j is calculated by

$$Q_j = \frac{(1-p)p^j}{1-p^{R+1}}, \quad \forall 0 \le j \le R, \tag{12}$$

where p is the collision probability. Using (10)(12), we measure the mean of channel access delay by

$$E[D^{(\rm a)}] = \sum_{j=0}^{R} E[S_j Q_j]. \tag{13}$$

Moreover, we define $S_{j,w}$ as the sum of delays from stage 0 to stage j that w is selected as the back-off value at stage j and meanwhile the channel is successfully accessed. Accordingly, the average of $S_{j,w}$ is expressed as

$$E[S_{j,w}] = \lceil \frac{w}{N_{\rm r}} \rceil \cdot E[\text{slot}] + E[U_{j-1}]. \tag{14}$$

Since the value w at stage j is uniformly selected in the range $[0, W_j - 1]$, the probability to select value w will be $1/W_j$. Therefore, we present the mean of the squared delay as

$$E[(D^{(\rm a)})^2] = \sum_{j=0}^{R}\Big(\sum_{w=0}^{W_j-1}(E[S_{j,w}])^2\frac{Q_j}{W_j}\Big). \tag{15}$$

Finally, we calculate the variance of channel access delay by

$$Var[D^{(\rm a)}] = E[(D^{(\rm a)})^2] - (E[D^{(\rm a)}])^2. \tag{16}$$

So far, for each sensor i, its channel access delay in terms of mean μ_i and variance Σ_i can be calculated by substituting the network configurations of its connecting AP, i.e., N and $N_{\rm r}$, into (13)(16). Specifically, we denote the average delay $E[S_j]$ at back-off stage R by Θ_i for each sensor i, which is the maximum channel access delay with $j = R$.

2) Transmission Delay: According to the Shannon bound, the data rate from sensor i to its connecting AP is

$$r_i = \frac{B}{N_{\rm r}}\log_2\Big(1+\frac{\rho g_i}{\sigma^2}\Big), \tag{17}$$

where ρ is the transmission power, g_i is the channel gain between sensor i and the AP, and σ^2 is the additive white Gaussian noise at the AP side. Note that total B available bandwidth is equally divided into $N_{\rm r}$ RA-RUs and therefore each RA-RU has $B/N_{\rm r}$ bandwidth for transmissions. Let b_f denote the data size of a compressed frame uploaded from each sensor with a specific inspection task $f \in \mathcal{F}$. Besides, we introduce a binary indicator $q_{i\psi}$, that is, $q_{i\psi} = 1$ if sensor i is connecting to AP ψ. Note that $T_{\rm P}$ is the longest time in the TF cycle consumed for a packet transmission, we present its value in AP ψ as

$$T_{\rm P}(\psi) = \max_{f\in\mathcal{F}, i\in\mathcal{N}_f}\{\frac{q_{i\psi}b_f}{r_i}\}, \tag{18}$$

where b_f/r_i is the wireless transmission delay of sensor i if it executes an inspection task of type f.

We assume that a fixed link capacity $r_{\rm cld}$ with a propagation delay $t^{\rm prog}$ is provided between the edge cluster and the remote cloud center for each sensor if need be. Let $b_f^{\rm ce}$ denote the data size of scheme CE delivered from the edge to the cloud. In this way, the transmission delay of sensor i with the inspection task f can be calculated by

$$D_i^{(\rm t)} = \frac{b_f}{r_i} + \sum_{k\in\mathcal{K}_f} x_{ik}^{\rm co}\cdot\frac{b_f}{r_{\rm cld}} + \sum_{k\in\mathcal{K}_f} x_{ik}^{\rm ce}\cdot\frac{b_f^{\rm ce}}{r_{\rm cld}} + 2(1-\sum_{k\in\mathcal{K}_f} x_{ik}^{\rm eo})\cdot t^{\rm prog}, \quad \forall i\in\mathcal{N}_f. \tag{19}$$

Specifically, we ignore the transmission delay of those inspection results responded by the cloud because of the small size but reserve the calculation of its propagation delay.

3) Computing Delay: We introduce a binary variable $y_k^\phi \in \{0,1\}$ as the **processor deployment decision**, where $y_k^\phi = 1$ represents that a local processor k is bought at the beginning of deployment and deployed by executing scheme ϕ. Specifically, a processor is allowed to deploy one type of inspection tasks with single scheme at a time. Imaging that a local processor may be powerful enough to execute a single inspection task, therefore, *it is potential to cut down the operation cost of buying unnecessary processors by allowing several sensors to share the same processor*. However, more sensors sharing the same processor leads to the increase of computing delay due to the queueing process. Different from the edge computing cluster, we assume that the remote cloud center has ability to process a frame immediately once it has arrived. Therefore, the computing delay of sensor i is

$$D_i^{(\rm c)} = \sum_{k\in\mathcal{K}_f}\sum_{\phi\in\Phi} x_{ik}^{\phi}\Big(\frac{\sum_{i'\in\mathcal{N}_f} x_{i'k}^{\phi}\lambda_{i'}}{\lambda_i}\cdot t_{f\phi}^{\rm edge} + t_{f\phi}^{\rm cld}\Big), \tag{20}$$

where $t_{f\phi}^{\rm edge}$ and $t_{f\phi}^{\rm cld}$ are the processing time of a frame at the edge and the cloud, respectively.

B. CVaR Measurement

Recall that the smart factory operates multiple inspection tasks during the autonomous production to guarantee the target outgoing quality. However, every time an inspection failure will cause a missing alarm of possible defects of products and therefore inevitably bring down the outgoing quality. Because of the uncertainty of the channel access delay, the system is exposed to the inspection risk of violating diverse latency requirements. Therefore, risk control is necessary to achieve reliable inspections for those delay-sensitive tasks. As an effective risk assessment measure, the CVaR has significant advantages over the value-at-risk (VaR) for quantifying the amount of tail risk in the investment portfolio. In the proposed system, we exploit the CVaR to measure the inspection risk.

We assume that the factory announces $1-\epsilon$ as its reliability threshold to guide the offloading service deployments for those

inspection tasks, where $\epsilon \in (0,1)$ is the maximum tolerance to inspection failure. The CVaR at level ϵ (ϵ-CVaR) [26] can be written as

$$\mathbb{P}\text{-CVaR}_\epsilon(\mathcal{L}) = \inf_{\beta\in\mathbb{R}} \left\{\beta + \frac{1}{\epsilon}\mathbb{E}_{\mathbb{P}}\Big[(\mathcal{L}-\beta)^+\Big]\right\},^1 \tag{21}$$

where $\mathcal{L}$ is a measurable loss function, β is an auxiliary variable, and $\mathbb{P}$ is the probability distribution. It can be observed that the ϵ-CVaR is a convex function respect to $\mathcal{L}$. We let the loss function $\mathcal{L}$ be $L_i - L_i^{\text{req}}$ for each sensor i, that is, the service latency minus the latency requirement. To ensure $1-\epsilon$ system reliability, the following *worse-case CVaR constraint* should be satisfied [27]:

$$\sup_{\mathbb{P}\in\mathcal{P}_i} \mathbb{P}\text{-CVaR}_\epsilon(L_i - L_i^{\text{req}}) \le 0, \quad \forall i \in \mathcal{N}, \tag{22}$$

$$\text{where } \mathcal{P}_i = \left\{\mathbb{P} : \mathbb{E}_{\mathbb{P}}[D_i^{(\text{a})}] = \mu_i, \mathbb{E}_{\mathbb{P}}[(D_i^{(\text{a})}-\mu_i)^2] = \Sigma_i\right\}.$$

Furthermore, we find out that the constraint in (22) is intractable to be solved when substituting (21). Fortunately, the worst-case CVaR can be converted into a semi-definite programming (SDP) and therefore calculated efficiently. We perform SDP reformulation [27] to (22) and obtain the following feasible set for each sensor i:

$$\begin{cases} \mathbf{M}_i \succcurlyeq \mathbf{0} \\ \beta_i + \frac{1}{\epsilon}\langle \mathbf{\Omega}_i, \mathbf{M}_i\rangle \le 0 \\ \mathbf{M}_i - \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & \Gamma_i - \beta_i \end{bmatrix} \succcurlyeq \mathbf{0} \end{cases},^2 \tag{23}$$

where Γ_i and $\mathbf{\Omega}_i$ are calculated by

$$\Gamma_i = D_i^{(\text{t})} + D_i^{(\text{c})} - L_i^{\text{req}}, \tag{24}$$

$$\mathbf{\Omega}_i = \begin{bmatrix} \Sigma_i + \mu_i^2 & \mu_i \\ \mu_i & 1 \end{bmatrix}. \tag{25}$$

C. System Objective

We characterize the operation cost before formally formulating the optimization problem. The operation cost is comprised of two parts, i.e., the *deployment cost* and the *running cost*. The deployment cost is the fixed expenses of buying local processors at the beginning of deployment. Note that those processors are operated at the edge to provide offloading services for EO and CE schemes, the deployment cost is written as

$$C^{\text{depl}} = c \cdot \sum_{k\in\mathcal{K}_f} \sum_{\phi\neq\text{co}} y_k^\phi, \tag{26}$$

where c is the unit price of a processor. The running cost is the cost spent over the time, including the energy consumption at the edge and the service fee paid to the cloud. Specifically, compared with the energy consumption of the computing services, the running cost of data communications can be ignored. Therefore, the per frame running cost of each sensor $i \in \mathcal{N}_f$ is

$$C_i^{\text{run}} = \sum_{k\in\mathcal{K}_f} \sum_{\phi\in\Phi} x_{ik}^\phi \Big(\gamma e_{f\phi}^{\text{edge}} t_{f\phi}^{\text{edge}} + \pi^{\text{cld}} t_{f\phi}^{\text{cld}}\Big), \tag{27}$$

[1] $x^+ = \max\{x, 0\}$. $\mathbb{R}$ denotes the set of all real numbers. $\mathbb{E}_{\mathbb{P}}(\cdot)$ denotes the expectation with respect to $\mathbb{P}$.

[2] $\mathbf{X} \succcurlyeq \mathbf{Y}$ implies that $\mathbf{X}-\mathbf{Y}$ is positive semidefinite. $\langle \mathbf{X}, \mathbf{Y}\rangle = \text{Tr}(\mathbf{X}, \mathbf{Y})$ is the trace scalar product.

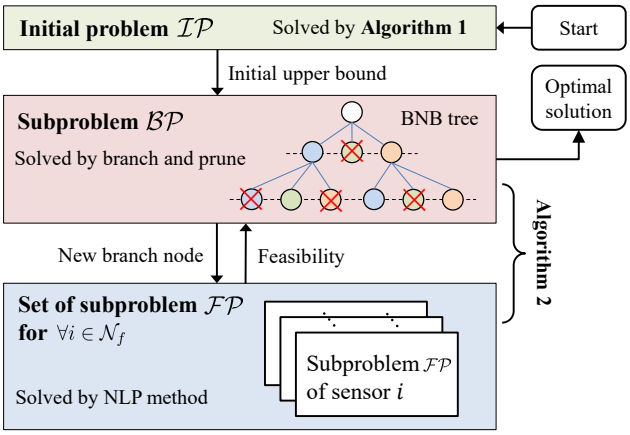


Fig. 2. Flowchart of the proposed BNC approach.

where $e_{f\phi}^{\text{edge}}$ is the power consumption at the edge when operating inspection task of type f with scheme ϕ, γ is the coefficient converting power consumption to cost, and π^{cld} is the service fee per unit time announced by the cloud. The equations (26)(27) indicate that some operation cost can be saved if the processor sharing strategy is adopted.

We define Δ as the duration of running, which is constrained by either the rate of depreciation of local processors or a particular purpose for autonomous manufacturing. Given the value of Δ, the **cost minimization problem** for those inspection tasks of type $f \in \mathcal{F}$ is formulated as follows:

$$\begin{aligned} \min_{x_{ik}^\phi, y_k^\phi, \beta_i, \mathbf{M}_i} \quad & C^{\text{depl}} + \Delta \sum_{i\in\mathcal{N}_f} \lambda_i C_i^{\text{run}} \\ \text{s.t. C1}: & \sum_{k\in\mathcal{K}_f}\sum_{\phi\in\Phi} x_{ik}^\phi = 1, \quad \forall i \in \mathcal{N}_f \\ \text{C2}: & \Big(\sum_{i\in\mathcal{N}_f} x_{ik}^\phi \lambda_i\Big) \cdot t_{f\phi}^{\text{edge}} \le 1, \quad \forall k \in \mathcal{K}_f, \phi \\ \text{C3}: & \sum_{\phi\in\Phi} y_k^\phi \le 1, \quad \forall k \in \mathcal{K}_f \\ \text{C4}: & y_k^\phi \ge x_{ik}^\phi, \quad \forall i \in \mathcal{N}_f, k \in \mathcal{K}_f, \phi \\ \text{C5}: & y_k^\phi \le \sum_{i\in\mathcal{N}_f} x_{ik}^\phi, \quad \forall k \in \mathcal{K}_f, \phi \\ \text{C6}: & \Gamma_i + \Theta_i < 0, \quad \forall i \in \mathcal{N}_f \\ \text{C7}: & \mathbf{M}_i \succcurlyeq \mathbf{0}, \quad \forall i \in \mathcal{N}_f \\ \text{C8}: & \beta_i + \frac{1}{\epsilon}\langle \mathbf{\Omega}_i, \mathbf{M}_i\rangle \le 0, \quad \forall i \in \mathcal{N}_f \\ \text{C9}: & \mathbf{M}_i - \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & \Gamma_i - \beta_i \end{bmatrix} \succcurlyeq \mathbf{0}, \quad \forall i \in \mathcal{N}_f \\ & x_{ik}^\phi \in \{0,1\},\ y_k^\phi \in \{0,1\} \quad \forall i \in \mathcal{N}_f, k \in \mathcal{K}_f, \phi. \end{aligned} \tag{28}$$

The constraint C1 indicates that a inspection task could not be deployed on multiple processors at the same time, and the constraint C2 guarantees the stability of service at the edge. The processor deployment is regulated by the constraints C3-C5. The constraints C6-C9 guarantee the system reliability subject to the predefined tolerance threshold. Specifically, the constraint C6 ensures that the remaining time duration allocated for channel access should not be longer than the maximum channel access delay basing on the configurations

Algorithm 1: Exhaustive Search To Solve $\mathcal{IP}$

Initialization: $[x_i^\phi] = 0$, $[\Phi_i] = \emptyset$, $[\hat{\lambda}_{i\phi}^{\mathrm{F}}] = 0$, $[\hat{\lambda}_{i\phi}^{\mathrm{IF}}] = +\infty$.
for $i \in \mathcal{N}_f$ **do**
  Assign $C_i^* = +\infty$, $\phi^* = \mathrm{eo}$.
  for $\phi \in \Phi$ **do**
    if $\mathcal{IP}$ *with* $x_i^\phi = 1$ *is feasible* **then**
      Update $\Phi_i = \Phi_i \cup \{\phi\}$, $\hat{\lambda}_{i\phi}^{\mathrm{F}} = \lambda_i$.
      Calculate $C_i = \begin{cases} \Delta\lambda_i(\gamma e_{f\phi}^{\mathrm{edge}} t_{f\phi}^{\mathrm{edge}} + \pi^{\mathrm{cld}} t_{f\phi}^{\mathrm{cld}}), & \phi = \mathrm{co} \\ c + \Delta\lambda_i(\gamma e_{f\phi}^{\mathrm{edge}} t_{f\phi}^{\mathrm{edge}} + \pi^{\mathrm{cld}} t_{f\phi}^{\mathrm{cld}}), & \phi \neq \mathrm{co} \end{cases}$.
      if $C_i < C_i^*$ **then**
        Record $C_i^* = C_i$, $\phi^* = \phi$.
      end
    else
      $\hat{\lambda}_{i\phi}^{\mathrm{IF}} = \lambda_i$.
    end
  end
  if $\Phi_i \neq \emptyset$ **then**
    Assign $x_i^{\phi^*} = 1$.
  end
end
Output: initial solution $[x_i^\phi]$, global information $[\Phi_i^{\mathrm{c}}]$, $[\hat{\lambda}_{i\phi}^{\mathrm{F}}]$, and $[\hat{\lambda}_{i\phi}^{\mathrm{IF}}]$.

of m and R. We observe that *1)* x_{ik}^ϕ and y_k^ϕ are binary decision variables, *2)* β_i and $\mathbf{M}_i$ are continuous variables, and *3)* the constraints $\mathrm{C7}, \mathrm{C8}, \mathrm{C9}$ are in SDP formulations. Therefore, the optimization problem in (28) is a mixed-integer nonlinear programming problem (MINLP), which is complicated and very difficult to solve [28].

V. Branch-and-Check Approach

In this section, we propose a *branch-and-check (BNC)* approach executed at the local edge computing controller to solve the optimization problem (28) optimally and efficiently, as shown in Fig. 2. In the proposed BNC approach, we separate the original problem (28) into two subproblems, including a 0-1 integer programming problem and a set of feasibility problems. The 0-1 integer programming problem is written as

$$\begin{aligned} \mathcal{BP}: \quad & \min_{x_{ik}^\phi, y_k^\phi} \quad C^{\mathrm{depl}} + \Delta \sum_{i \in \mathcal{N}_f} \lambda_i C_i^{\mathrm{run}} \\ & \text{s.t. } \mathrm{C1}, \mathrm{C2}, \mathrm{C3}, \mathrm{C4}, \mathrm{C5}, \mathrm{C6}, \\ & x_{ik}^\phi \in \{0,1\},\ y_k^\phi \in \{0,1\} \quad \forall i \in \mathcal{N}_f, k \in \mathcal{K}_f, \phi, \end{aligned}$$

which gets rid of continuous variables. For each sensor $i \in \mathcal{N}_f$, the feasibility problem is formulated by the worse-case CVaR constraints C7-C9:

$$\begin{aligned} \mathcal{FP}: \quad & \min_{\beta_i, \mathbf{M}_i} \quad 0 \\ & \text{s.t. } \mathrm{C7}, \mathrm{C8}, \mathrm{C9}, \end{aligned}$$

It can be observed that the continuous variables do not appear in the objective function.

The proposed BNC approach is based on a branch-and-bound (BNB) tree. That is to say, its optimality can be guaranteed. Compared with the widely used BNB method, the proposed BNC approach greatly reduces the state space search. To reduce the solution space and accelerate the search process, a list of **global information** needs to be maintained:

1) C^*: The minimum operation cost found so far, which is the upper bound of the system objective.
2) $[\Phi_i^{\mathrm{c}}]$: Binary variables to indicate the candidate service placement schemes of sensor $i \in \mathcal{N}_f$.
3) $[\hat{\lambda}_{i\phi}^{\mathrm{F}}]$, $[\hat{\lambda}_{i\phi}^{\mathrm{IF}}]$: The maximum feasible and minimum infeasible frame rates of sensor i with scheme ϕ found so far.

A. Initial Upper Bound

The processor sharing strategy can effectively cut down the operation cost, therefore, serving each sensor by a dedicated processor provides the upper bound of the problem (28). Without taking processor sharing into account, the offloading service deployment of each sensor $i \in \mathcal{N}_f$ can be solved separately by replacing x_{ik}^ϕ with x_i^ϕ:

$$\begin{aligned} \mathcal{IP}: \min_{\beta_i, \mathbf{M}_i, x_i^\phi} \ & c \sum_{\phi \neq \mathrm{co}} x_i^\phi + \Delta\lambda_i \sum_{\phi \in \Phi} x_i^\phi \left(\gamma e_{f\phi}^{\mathrm{edge}} t_{f\phi}^{\mathrm{edge}} + \pi^{\mathrm{cld}} t_{f\phi}^{\mathrm{cld}}\right) \\ \text{s.t. } & \mathrm{C1}': \sum_{\phi \in \Phi} x_i^\phi = 1, \\ & \mathrm{C2}': x_i^\phi \lambda_i \cdot t_{f\phi}^{\mathrm{edge}} \leq 1, \quad \forall\phi \\ & \mathrm{C6}': \Gamma_i' + \Theta_i < 0, \\ & \mathrm{C7}': \mathbf{M}_i \succcurlyeq \mathbf{0}, \\ & \mathrm{C8}': \beta_i + \frac{1}{\epsilon}\langle \mathbf{\Omega}_i, \mathbf{M}_i \rangle \leq 0, \\ & \mathrm{C9}': \mathbf{M}_i - \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & \Gamma_i' - \beta_i \end{bmatrix} \succcurlyeq \mathbf{0}, \\ & x_i^\phi \in \{0,1\}, \quad \forall\phi \end{aligned}$$

$$\text{where } \Gamma_i' = \left[\frac{b_f}{r_i} + x_i^{\mathrm{co}} \frac{b_f}{r_{\mathrm{cld}}} + x_i^{\mathrm{ce}} \frac{b_f^{\mathrm{ce}}}{r_{\mathrm{cld}}} + 2(1 - x_i^{\mathrm{eo}}) \cdot t^{\mathrm{prog}}\right] + \sum_{\phi \in \Phi} x_i^\phi (t_{f\phi}^{\mathrm{edge}} + t_{f\phi}^{\mathrm{cld}}) - L_i^{\mathrm{req}}.$$

Since the service placement scheme ϕ has only three choices, we traverse all choices for each sensor i and then choose a feasible scheme ϕ with least operation cost as the initial upper bound. The overall description of the exhaustive search to solve $\mathcal{IP}$ is shown in Algorithm 1. We can observe that Algorithm 1 also determines the candidate service placement schemes $[\Phi_i^{\mathrm{c}}]$.

It should be noted that the latency requirements of some sensors may not be satisfied with any service placement schemes. Therefore, those sensors should be removed for the rest of the search process, that is,

$$\mathcal{N}_f = \mathcal{N}_f \setminus \{i : \Phi_i = \emptyset, \forall i \in \mathcal{N}_f\}. \tag{29}$$

In order to successfully deploy all sensors to the manufacturing lines, the factory operator may try to decrease frame sampling rate or relax latency requirement of those infeasible sensors. Besides, the number of RA-RUs or associated sensors to a AP can be modified to reduce the risk of channel access delay.

B. Solution Using BNC

The proposed BNC approach works on the solution space of binary variables (x_{ik}^ϕ, y_k^ϕ) by adopting the BNB tree. According to the previous discussions, the initial upper bound ensures that the sensors in the rest of the search process at

least have one available scheme. Therefore, when a new branch node is reached, there are two available branching strategies, that is, deploy the current sensor to a vacant processor or a processor occupied by other sensors. We make $|\mathcal{K}_f| = |\mathcal{N}_f|$ so that there always has a vacant processor to select.

The overall description of the proposed BNC approach is shown in Algorithm 2. During the search process, the approach checks whether or not the current solution satisfies the constraints of $\mathcal{BP}$ as well as the set of feasibility problems $\mathcal{FP}$. Let $C(x_{ik}^{\phi}, y_k^{\phi})$ be the total operation cost achieved by the solution $(x_{ik}^{\phi}, y_k^{\phi})$. We can observe that a branch should be pruned once the current operation cost of the branching node is larger or equal to the upper bound, i.e., $C(x_{ik}^{\phi}, y_k^{\phi}) \geq C^*$. We update C^* by $C(x_{ik}^{\phi}, y_k^{\phi})$ when a leaf node, where all sensors have been successfully deployed, is feasible and $C(x_{ik}^{\phi}, y_k^{\phi}) < C^*$. Furthermore, the total frame rates arriving to processor k is calculated by $\lambda = \sum_{i' \in \mathcal{N}_f} x_{i'k}^{\phi} \lambda_{i'}$. The inequality $\lambda \leq \lfloor 1/t_{f\phi}^{\text{edge}} \rfloor$ is used to take the place of constraint C2, where the latter term is the service capability of a processor. The approach maintains the global information $\hat{\lambda}_{i\phi}^{\text{F}}$ and $\hat{\lambda}_{i\phi}^{\text{IF}}$ to reduce the check times of feasibility problems $\mathcal{FP}$. There are three cases for sensor i with scheme ϕ: *i)* if $\lambda \leq \hat{\lambda}_{i\phi}^{\text{F}}$ then $\mathcal{FP}$ is feasible; *ii)* if $\lambda \geq \hat{\lambda}_{i\phi}^{\text{IF}}$ then $\mathcal{FP}$ is infeasible; *iii)* if $\lambda > \hat{\lambda}_{i\phi}^{\text{F}}$ and $\lambda < \hat{\lambda}_{i\phi}^{\text{IF}}$, the feasibility of $\mathcal{FP}$ needs to be checked again. If $\mathcal{FP}$ turns out to be feasible then $\hat{\lambda}_{i\phi}^{\text{F}}$ is updated by λ; otherwise, $\hat{\lambda}_{i\phi}^{\text{IF}}$ is updated by λ.

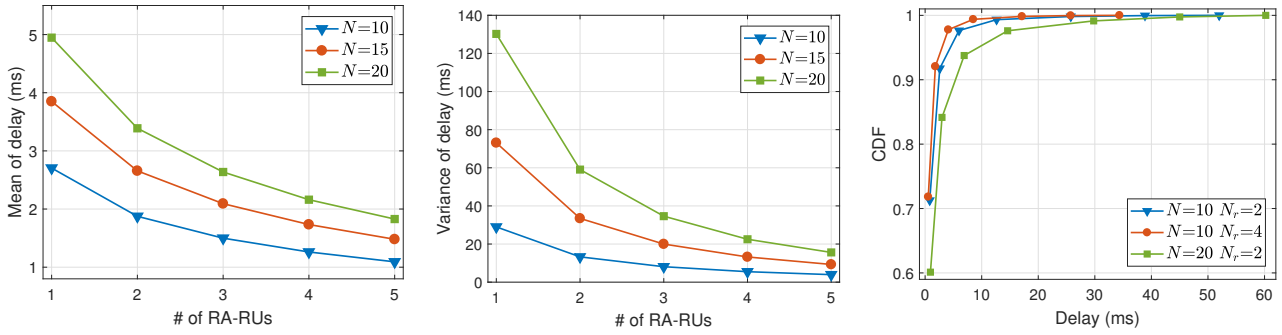
Since the problem $\mathcal{FP}$ is in SDP formulation, the feasibility check to $\mathcal{FP}$ has polynomial worst-case complexity. To perform an exhaustive search to solve $\mathcal{IP}$, Algorithm 1 requires a total $N|\Phi|$ times of feasibility checks. It should be noted that the NP-hard property of the MINLP problem (28) will not be changed by any equivalent transformation. That is to say, the computational complexity of the proposed BNC approach, i.e., Algorithm 2, is still dominated by the solution space of the BNB tree. Therefore, the proposed BNC approach in the worse case needs an exponential number of feasibility checks. Nevertheless, the advantage over the Benders decomposition method [29], which tackles the integer programming subproblem at every iteration, is that the proposed BNC approach solves the 0-1 integer programming problem $\mathcal{BP}$ only once. Besides, the proposed BNC approach ensures that the cost minimization problem in (28) can be solved optimally and efficiently. In the beginning, the problem $\mathcal{IP}$ generates a high-quality initial upper bound and therefore can reject more branch nodes. In addition, the global information helps to reduce the search space in two aspects: *i)* the records of candidate service placement schemes $[\Phi_i^{\text{c}}]$ avoid invalid search process and therefore can greatly prune the BNB tree; *ii)* most of the time-consuming feasibility check to $\mathcal{FP}$ can be saved with the assistance of $\hat{\lambda}_{i\phi}^{\text{F}}$ and $\hat{\lambda}_{i\phi}^{\text{IF}}$. We will demonstrate through simulations that the proposed BNC approach can save a huge number of feasibility checks and therefore accelerate the convergence to the optimal solution.

Algorithm 2: Branch-and-Check (B&C) Approach

Solve $\mathcal{IP}$ by Algorithm 1 and obtain the initial solution $[x_i^{\phi}]$, the global information $[\Phi_i^{\text{c}}]$, $[\hat{\lambda}_{i\phi}^{\text{F}}]$, and $[\hat{\lambda}_{i\phi}^{\text{IF}}]$.
Assign $\hat{x}_{ik}^{\phi} = \hat{y}_k^{\phi} = x_i^{\phi}$ for $\forall k = i$.
Remove infeasible sensors from $\mathcal{N}_f$ by (29) and meanwhile make $|\mathcal{K}_f| = |\mathcal{N}_f|$.
Calculate the upper bound $C^* = C(x_{ik}^{\phi}, y_k^{\phi})$.
Create a queue and store the root node as $(0, [\hat{x}_{ik}^{\phi}], [\hat{y}_k^{\phi}])$.
while *the queue is not empty* **do**
  Take a node $(i, [x_{ik}^{\phi}], [y_k^{\phi}])$ off the queue.
  if i *is not the last element of* $\mathcal{N}_f$ **then**
    Update i to the next element of $\mathcal{N}_f$.
    for $\phi \in \Phi_i$ **do**
      1) Deploy to a vacant processor:
      Set $[\tilde{x}_{ik}^{\phi}] = [x_{ik}^{\phi}]$, $[\tilde{y}_k^{\phi}] = [y_k^{\phi}]$.
      Set $\tilde{x}_{ik}^{\phi} = 1$ and $\tilde{y}_k^{\phi} = 1$ with any one $k \in \{k : \tilde{y}_k^{\phi} = 0, k \in \mathcal{K}_f\}$.
      Store $(i, [\tilde{x}_{ik}^{\phi}], [\tilde{y}_k^{\phi}])$ to the queue if $C(\tilde{x}_{ik}^{\phi}, \tilde{y}_k^{\phi}) < C^*$.
      2) Deploy to an occupied processor:
      for $k \in \{k : \tilde{y}_k^{\phi} = 1, k \in \mathcal{K}_f, \phi \neq \text{co}\}$ **do**
        Set $[\tilde{x}_{ik}^{\phi}] = [x_{ik}^{\phi}]$, $[\tilde{y}_k^{\phi}] = [y_k^{\phi}]$, $\tilde{x}_{ik}^{\phi} = 1$.
        Calculate $\lambda = \sum_{i' \in \mathcal{N}_f} x_{i'k}^{\phi} \lambda_{i'}$.
        if $C(\tilde{x}_{ik}^{\phi}, \tilde{y}_k^{\phi}) < C^*$ *and* $\lambda \leq \lfloor 1/t_{f\phi}^{edge} \rfloor$ **then**
          Set $flag = feasible$.
          for $i' \in \{i' : x_{i'k}^{\phi} = 1, i' \in \mathcal{N}_f\}$ **do**
            if $\Gamma'_{i'} + \Theta_{i'} < 0$ **then**
              if $\lambda > \hat{\lambda}_{i'\phi}^{F}$ *and* $\lambda < \hat{\lambda}_{i'\phi}^{IF}$ **then**
                if $\mathcal{FP}$ *is feasible to* i' **then**
                  Set $\hat{\lambda}_{i'\phi}^{\text{F}} = \lambda$.
                else
                  Set $\hat{\lambda}_{i'\phi}^{\text{IF}} = \lambda$ and $flag = infeasible$.
                end
              else if $\lambda \geq \hat{\lambda}_{i'\phi}^{IF}$ **then**
                Set $flag = infeasible$.
              end
            else
              Set $flag = infeasible$.
            end
          end
          Store $(i, [\tilde{x}_{ik}^{\phi}], [\tilde{y}_k^{\phi}])$ to the queue if $flag$ is *feasible*.
        end
      end
    end
  else
    if $C(x_{ik}^{\phi}, y_k^{\phi}) < C^*$ **then**
      Update $C^* = C(x_{ik}^{\phi}, y_k^{\phi})$.
      Update $[\hat{x}_{ik}^{\phi}] = [x_{ik}^{\phi}]$, $[\hat{y}_k^{\phi}] = [y_k^{\phi}]$.
    end
  end
end
Output: optimal value C^*, optimal solution $[\hat{x}_{ik}^{\phi}]$ and $[\hat{y}_k^{\phi}]$.

VI. Simulation Results

In this section, we evaluate the proposed framework through extensive simulations for those sensors executing the inspection tasks of type f. Within the $40\text{m}\times40\text{m}$ smart factory, four 802.11.ax APs are located in a grid topology. Total $|\mathcal{N}_f|$ vision sensors are randomly and uniformly deployed and each sensor connects to its nearest AP by default. The key simulation parameters are listed in Table II. Specifically, the unit price c and the power consumption of a processor are according to the actual information of GeForce GTX 1060. The service fee of the cloud π^{cld} comes from the table in [31] published



(a) Mean of delay (b) Variance of delay (c) CDF

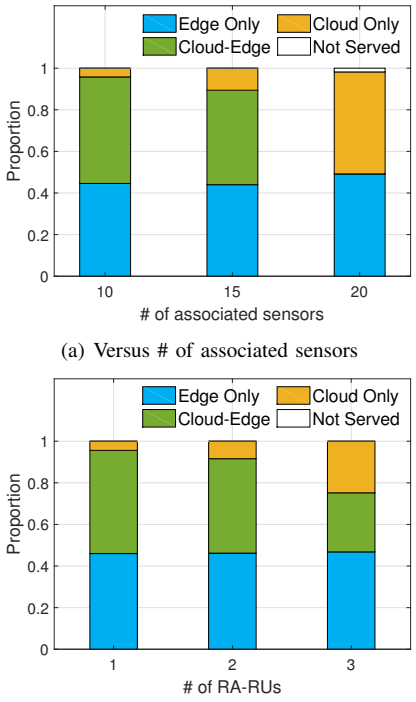
Fig. 3. Probability distribution of channel access delay.

TABLE II
LIST OF KEY SIMULATION PARAMETERS

Parameter	Value
Frequency band, B, ρ	5GHz, 20MHz, 16dBm
Pathloss model [30]	$PL(d) = 40.05$ $+20\log_{10}(\min\{d, 10\}) + (d > 10)35\log_{10}(d/10)$
Shadowing standard deviation	5dB
Control bit rate	11.8Mbps
Trigger frame (TF)	140bytes at 11.8Mbps
Packet header (H)	16(PHY) + 28(MAC)bytes
ACK	14 bytes at 11.8Mbps
SIFS, T_δ, t^{prog}	16us, 3us, 10ms
W, m, R	32, 5, 6
N, N_{r}, $\lvert\mathcal{N}_f\rvert$	15, 2, 10
π^{cld}, c, γ	\$1.2, \$799, \$2.3874
Δ, $1-\epsilon$	2year, 0.95
b_f, b_f^{ce}, r_{cld}	30KB, 5KB, 50Mbps
Processor's power consumption	120W

by Google Cloud and the coefficient converting power consumption to cost γ is announced by Taiwan Power Company in 2020. Other parameters in Table II, if not mentioned, follow the similar settings in [22, 23]. Moreover, we define the latency requirement of each sensor i as the reciprocal of its frame sampling rate, i.e., $L_i^{\text{req}} = 1/\lambda_i$. Since the cloud center generally is more powerful than the edge cluster, we assume that the processing time at the cloud is three times faster than the local processor at the edge according to our empirical investigation, which is reasonable but not absolute. We compare five service placement approaches for the decomposable inspection tasks, including *Edge-Only*, *Cloud-Only*, *Dedicated*, *Proposed*, and *R&R* [32]. Similar to the *Proposed* approach, the *Dedicated* approach takes full advantage of the cloud-edge computing architecture to enable all three feasible service placement schemes. However, it serves each task by a dedicated processor while the *Proposed* approach allows processor sharing. The *R&R* approach in [32], which utilizes binary variable relaxation and recovery to tackle the difficulty of the MINLP problem, is applied for performance comparison with the proposed BNC approach.

We first present the probability distribution of channel access delay in Fig. 3. We observe that more number of associated sensors to an AP increases both the mean and variance of channel access delay because a transmitted packet has a higher probability to encounter a collision. On the contrary, more RA-RUs decreases the collision probability and therefore helps to

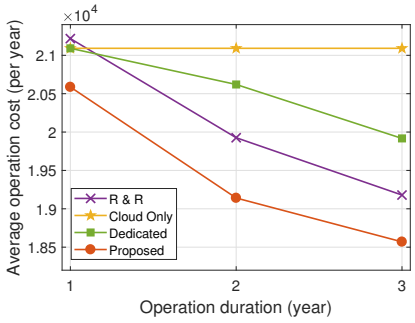


(a) Versus # of associated sensors

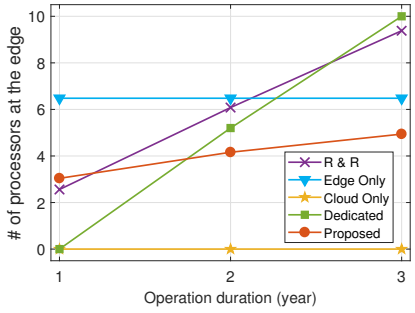
(b) Versus # of RA-RUs

Fig. 4. Service placement of *Proposed* versus network configurations.

win the UORA contention process. The cumulative distribution function (CDF) in Fig. 3(c) shows that the system can achieve less than 20ms channel access delay in most cases but still has a small probability to suffer from high delay. That is to say, the risk of inspection failure is inevitable under strict latency requirements. By adopting the *Proposed* approach, the service placement versus different network configurations is illustrated in Fig. 4, where the proportion represents the number sensors of each service placement scheme over the total number. With the increase of associated sensors to each AP in Fig. 4(a), the CO scheme is more attractive than the CE scheme because the cloud has more powerful computing capabilities and therefore the CO scheme can greatly reduce the total service latency. Besides, some sensors under $N = 20$ even fail to be served due to the violation of the latency requirement and the reliability threshold. Basing on the discussions of Fig. 3, we expect that the increase of RA-RUs will achieve similar outcomes to the

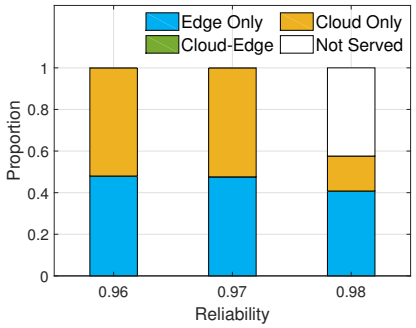


(a) Average operation cost per year

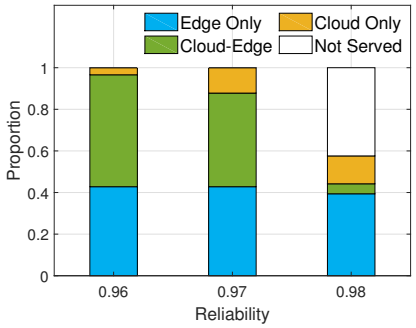


(b) # of processors at the edge

Fig. 5. Performance versus operation duration Δ.



(a) Service placement of *Dedicated*



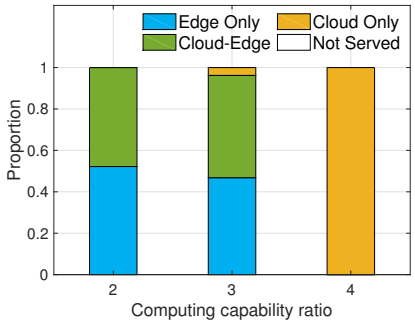
(b) Service placement of *Proposed*

Fig. 6. Performance versus reliability threshold $1-\epsilon$.

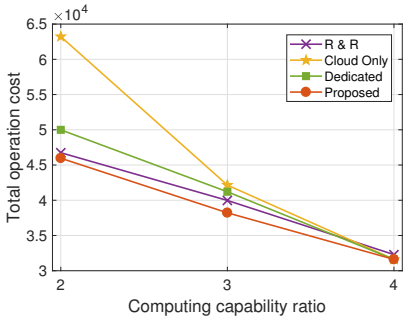
decease of associated sensors. However, Fig. 4(b) displays an opposite result because more RA-RUs also means less bandwidth allocated to each RA-RU and therefore results in a longer transmission delay. In other words, there is a tradeoff between the channel access delay and transmission delay when a different number of RA-RUs is announced.

The performance versus operation duration is demonstrated in Fig. 5, where the average operational cost is calculated by the total operation cost over the operation duration in years. Specifically, the *Edge-Only* approach is not illustrated in Fig. 5(a) because it only can serve about half of sensors while the other approaches successfully serve all sensors. We observe that the average operation cost decreases with the increase of duration in Fig. 5(a). Compared with the *Dedicated* approach, the *Proposed* approach deploys fewer processors at the edge so that it can save a large amount of operation cost. The *R&R* approach requires more operation cost than the *Proposed* approach because the former applies a rounding-off method to recover the binary variables while the latter can achieve the optimal offloading service deployment. Fig. 5(b) indicates that the EO scheme will be preferred with a long duration of running because the extra cost to the processor deployment plus the energy consumption will be less than the service fee paid to the cloud. Fig. 6 exhibits some evaluation results under the reliability threshold, which determines the tolerance of inspection failure. Some sensors will encounter unsatisfied inspection risk under the strict reliability threshold and therefore they are rejected to serve. Compared with the *Dedicated* approach, the *Proposed* approach exploits the advantage of processor sharing and the CE scheme becomes the best scheme to save operation cost when the inspection risk is not the matter.

Note that we have assumed that the processing time at the cloud is three times faster than the local processor at the edge for the previous simulations. However, this assumption is not compatible with all cases in reality. Therefore, we demonstrate the performance versus the ratio of the computing capability of the cloud over the local processor at the edge, as shown in Fig. 7. Fig. 7(a) shows that the CO scheme becomes more attractive but the EO scheme gradually losses the advantage with the increase of computing capability of the cloud. The reason is that the CO scheme provides faster offloading services so that the round-trip of propagation delay becomes negligible. In Fig. 7(b), the *Edge-Only* approach is not illustrated due to the same reason as Fig. 5(a). We observe that the higher computing capability of the cloud will save the total operation cost because the CO scheme cuts down the deployment cost of buying local processors compared with the EO and CE schemes. Finally, we find out that the total operation cost increases steadily against the number of sensors in Fig. 8(a) because more sensors are successfully served. The *R&R* approach outperforms other approaches except for the *Proposed* approach. Moreover, the operation cost reduction between the *Dedicated* and *Proposed* approaches becomes larger with the increase of sensors. Note that the practical computation time of the proposed BNC approach depends on not only the computing power of the processor but also which programming language is used. Altenatively, we demonstrate the number of feasible checks between the proposed BNC approach and the benchmark BNB method to verify the efficiency since the cost of solving a feasibility check is in polynomial time complexity. In Fig. 8(b), it should be noted that the left and right y-axes denote the scale of the required number of feasible checks between the BNB and BNC approaches, respectively. We can observe that the proposed BNC greatly reduces the number of

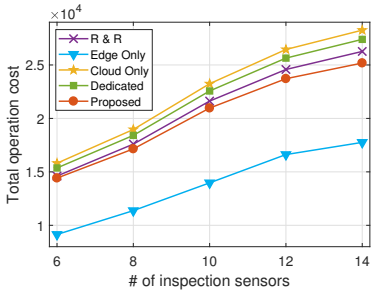


(a) Service placement of *Proposed*

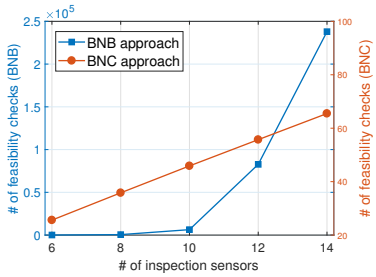


(b) Total operation cost

Fig. 7. Performance versus computing capability ratio.



(a) Total operation cost



(b) # of feasibility checks

Fig. 8. Performance versus # of inspection sensors.

feasibility checks of $\mathcal{FP}$ compared with the benchmark BNB method and therefore can save a large amount of computation time. Specifically, the average number of feasible checks is approximately proportional to the number of inspection sensors, which means that the proposed BNC approach on average can achieve polynomial computation time.

VII. Conclusions

We proposed a risk-aware cloud-edge computing framework for the delay-sensitive inspections of autonomous manufacturing. We utilize CVaR to measure the inspection risk and formulate a cost minimization problem to optimally deploy the decomposable inspection tasks. The results show that there is a tradeoff between the channel access delay and transmission delay when a different number of RA-RUs is announced. Besides, the EO scheme is preferred with a long duration of running and the CE scheme is the best scheme to save operation cost when the inspection risk is not the matter. The processor sharing strategy can save a large amount of operation costs. The proposed framework is valuable for future IIoT with next-generation wireless communication and cloud-edge computing infrastructure.

Note that the proposed framework provides a one-shot solution for optimal offloading service deployment by considering the long-term operation cost, it is difficult to cope with the dynamic changes of the environments, such as varying wireless channels, indeterministic arrivals of computation tasks, and so on. Therefore, one of the future directions is to propose some online solution against system dynamics basing on the current proposed solution, which is utilized as an initial resource allocation and offloading service deployment. The power control for channel access is another future direction, which can not only reduce the energy consumption of data communications and but also decrease co-channel interference and thus reduce the channel access delay. Furthermore, a promising research topic is a coexistence between 802.11ax and the upcoming 5G NR-U communication protocols for IIoT.

Acknowledgment

Hung-Yu Wei is grateful for the funding support from Ministry of Science and Technology (MOST) of Taiwan under grant 109-2221-E-002-148-MY2, 109-2218-E-002-018- and 109-2221-E-002-085-.

References

[1] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4665–4673, 2018.

[2] Z. Bi, L. Da Xu, and C. Wang, "Internet of things for enterprise systems of modern manufacturing," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1537–1546, 2014.

[3] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4674–4682, 2018.

[4] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Netw.*, vol. 31, no. 1, pp. 52–58, 2016.

[5] Y. Zhang, C.-Y. Wang, and H.-Y. Wei, "Parking reservation auction for parked vehicle assistance in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3126–3139, 2019.

[6] J. Wang, K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Trans. Mobile Comput.*, 2019.

[7] Z. Wu, B. Li, Z. Fei, Z. Zheng, and Z. Han, "Energy-efficient robust computation offloading for fog-iot systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4417–4425, 2020.

[8] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay-sensitive task offloading in the 802.11 p-based vehicular fog computing systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 773–785, 2019.

[9] G. Hampel, C. Li, and J. Li, "5g ultra-reliable low-latency communications in factory automation leveraging licensed and unlicensed bands," *IEEE Wireless Commun.*, vol. 57, no. 5, pp. 117–123, 2019.

[10] S. Uryasev, "Conditional value-at-risk: Optimization algorithms and applications," in *Proceedings of the IEEE/IAFE/INFORMS 2000 Conference on Computational Intelligence for Financial Engineering (CIFEr)(Cat. No. 00TH8520)*. IEEE, 2000, pp. 49–57.

[11] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-optimal dynamic computation offloading for industrial iot in fog computing," *IEEE Trans. on Green Commun. Netw.*, 2019.

[12] Y. Zhang, J.-H. Liu, C.-Y. Wang, and H.-Y. Wei, "Decomposable intelligence on cloud-edge iot framework for live video analytics," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8860–8873, 2020.

[13] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multi-agent deep reinforcement learning for joint multi-channel access and task offloading of mobile edge computing in industry 4.0," *IEEE Internet Things J.*, 2020.

[14] L. Li, D. Shi, R. Hou, X. Li, J. Wang, H. Li, and M. Pan, "Data-driven optimization for cooperative edge service provisioning with demand uncertainty," *IEEE Internet Things J.*, 2020.

[15] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. C. Leung, "Energy-efficient workload allocation and computation resource configuration in distributed cloud/edge computing systems with stochastic workloads," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1118–1132, 2020.

[16] S. Batewela, C.-F. Liu, M. Bennis, H. A. Suraweera, and C. S. Hong, "Risk-sensitive task fetching and offloading for vehicular edge computing," *IEEE Commun. Lett.*, vol. 24, no. 3, pp. 617–621, 2019.

[17] E. El Haber, H. A. Alameddine, C. Assi, and S. Sharafeddine, "A reliability-aware computation offloading solution via uav-mounted cloudlets," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*. IEEE, 2019, pp. 1–6.

[18] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1405–1418, 2020.

[19] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on ieee 802.11 ax high efficiency wlans," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 197–216, 2018.

[20] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, 2000.

[21] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, "Performance of reliable transport protocol over ieee 802.11 wireless lan: analysis and enhancement," in *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2. IEEE, 2002, pp. 599–607.

[22] G. Naik, S. Bhattarai, and J.-M. Park, "Performance analysis of uplink multi-user ofdma in ieee 802.11 ax," in *2018 International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[23] S. Bhattarai, G. Naik, and J.-M. J. Park, "Uplink resource allocation in ieee 802.11 ax," in *2019 International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[24] P. Raptis, V. Vitsas, and K. Paparrizos, "Packet delay metrics for ieee 802.11 distributed coordination function," *Mobile Networks and Applications*, vol. 14, no. 6, p. 772, 2009.

[25] T. Sakurai and H. L. Vu, "Mac access delay of ieee 802.11 dcf," *IEEE Trans. Wireless Commun.*, vol. 6, no. 5, pp. 1702–1710, 2007.

[26] R. T. Rockafellar, S. Uryasev *et al.*, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[27] S. Zymler, D. Kuhn, and B. Rustem, "Distributionally robust joint chance constraints with second-order moment information," *Mathematical Programming*, vol. 137, no. 1-2, pp. 167–198, 2013.

[28] C. A. Floudas, *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.

[29] A. M. Geoffrion, "Generalized benders decomposition," *Journal of optimization theory and applications*, vol. 10, no. 4, pp. 237–260, 1972.

[30] IEEE 802.11 Task Group AX, "TGax Simulation Scenarios," https://mentor.ieee.org/802.11/dcn/14/11-14-0980-16-00ax-simulation-scenarios.docx.

[31] G. Cloud, "Gpus pricing." [Online]. Available: https://cloud.google.com/compute/gpus-pricing

[32] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, 2018.

Yi Zhang received the B.S. degree in software engineering from Software College, Xiamen University in 2014. He received the M.S. and the Ph.D. degree from Graduate Institute of Communication Engineering, National Taiwan University in 2016. He has been an assistant engineer in Quanzhou Institute of Equipment Manufacturing, Haixi Institutes, Chinese Academy of Sciences, during 2016-2017. He is currently an Assistant Professor with Department of Information and Communication Engineering, Xiamen University. His research interests include mobile and wireless networking, fog/edge computing, and game theoretical models for communications networks.

Hung-Yu Wei is a Professor in Department of Electrical Engineering and Graduate Institute of Communications Engineering, National Taiwan University. Currently, he serves as Associate Chair in Department of Electrical Engineering. He received the B.S. degree in electrical engineering from National Taiwan University in 1999. He received the M.S. and the Ph.D. degree in electrical engineering from Columbia University in 2001 and 2005 respectively. He was a summer intern at Telcordia Applied Research in 2000 and 2001. He was with NEC Labs America from 2003 to 2005. He joined Department of Electrical Engineering at the National Taiwan University in July 2005. His research interests include next-generation wireless broadband networks, IoT, vehicular networking, fog/edge computing, cross-layer design for wireless multimedia, and game theoretical models for communications networks.

Dr. Wei received NTU Excellent Teaching Award in 2008 and 2018. He also received "Recruiting Outstanding Young Scholar Award" from the Foundation for the Advancement of Outstanding Scholarship in 2006, K. T. Li Young Researcher Award from ACM Taipei/Taiwan Chapter and The Institute of Information and Computing Machinery in 2012, Excellent Young Engineer Award from the Chinese Institute of Electrical Engineering in 2014, Wu Ta You Memorial Award from MOST in 2015, and Outstanding Research Award from MOST in 2020. He has been actively participating in NGMN, IEEE 802.16, 3GPP, IEEE P1934, and IEEE P1935 standardization. He serves as Vice Chair of IEEE P1934 Working Group to standardize fog computing and networking architecture. He serves as Secretary for IEEE Fog/Edge Industry Community. He also serves as an Associate Editor for IEEE IoT journal. He is an IEEE certified Wireless Communications Professional. He was the Chair of IEEE VTS Taipei Chapter during 2016-2017. He is currently the Chair of IEEE P1935 working group for edge/fog management and orchestration standard.