Resource Offload Consolidation Based on Deep-Reinforcement Learning Approach in Cyber-Physical Systems

Mekala, M. S. ; Jolfaei, Alireza ; Srivastava, Gautam ; Zheng, Xi ; Anvari-Moghaddam, Amjad; Viswanathan, P.

Published in:
IEEE Transactions on Emerging Topics in Computational Intelligence

DOI (link to publication from Publisher):
10.1109/TETCI.2020.3044082

Publication date:
2022

Document Version
Accepted author manuscript, peer reviewed version

Citation for published version (APA):
Mekala, M. S., Jolfaei, A., Srivastava, G., Zheng, X., Anvari-Moghaddam, A., & Viswanathan, P. (2022). Resource Offload Consolidation Based on Deep-Reinforcement Learning Approach in Cyber-Physical Systems. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *6*(2), 245-254. Advance online publication. https://doi.org/10.1109/TETCI.2020.3044082

Resource Offload Consolidation Based on Deep-Reinforcement Learning Approach in Cyber-Physical Systems

M. S. Mekala, Alireza Jolfaei, *Senior Member, IEEE*, Gautam Srivastava, *Senior Member, IEEE*, Xi Zheng, Amjad Anvari-Moghaddam, *Senior Member, IEEE*, and P. Viswanathan

***Abstract*—In cyber-physical systems, it is advantageous to leverage cloud with edge resources to distribute the workload for processing and computing user data at the point of generation. Services offered by cloud are not flexible enough against variations in the size of underlying data, which leads to increased latency, violation of deadline and higher cost. On the other hand, resolving above-mentioned issues with edge devices with limited resources is also challenging. In this work, a novel reinforcement learning algorithm, Capacity-Cost Ratio-Reinforcement Learning (CCR-RL), is proposed which considers both resource utilization and cost for the target cyber-physical systems. In CCR-RL, the task offloading decision is made considering data arrival rate, edge device computation power, and underlying transmission capacity. Then, a deep learning model is created to allocate resources based on the underlying communication and computation rate. Moreover, new algorithms are proposed to regulate the allocation of communication and computation resources for the workload among edge devices and edge servers. The simulation results demonstrate that the proposed method can achieve a minimal latency and a reduced processing cost compared to the state-of-the-art schemes.**

***Index Terms*—Artificial intelligence, edge computing, game theory, deep-reinforcement learning, resource provision, measurement systems.**

I. Introduction

TRADITIONAL computation methods are not suitable for edge devices due to trade-offs among *low leased-cost, latency, and high bandwidth* [1]. This paper aims at formulating the latency problem of cloud-edge frameworks, where the latency problem is resolved with a computational offloading mechanism at the device-level to share the resources based on a top-down approach. Edge-cloud systems can enhance the computing resources flexibility of edge devices by scaling up (and down) the resource provision capacity based on workload fluctuations [2]. Throughout the paper, we consider that, C-level refers to cloud level computing, I-level refers intermediate level or region-level ($\mathbb{R}$), which includes Access Points (APs)/ Base Stations (BSs), and L-level refers to low-level or zone-level ($\mathbb{Z}$) which includes Edge Devices (EDs).

Resource flexibility aims to reduce the resource leasing cost by regulating the resource usage rate. The underlying objective is to accomplish optimal resource scheduling. The existing resource scheduling models ignore the impact of the leasing costs since they have no cost measurement mechanisms in place while making the computation offloading decisions [3], [4]. The performance hiccups happen while optimizing the configured resources, which cause resource wastage and a higher leasing cost. The existing approaches often make wrong decisions because they do not balance the resource fragmentation time and deadline time during resource sharing and task offloading to regulate workload fluctuations. For example, in [5], a price estimation method is designed to assess the resource requirement ratio based on performance tracking. A Reinforcement Learning (RL) technique is employed to restrict edge device makespan and Average Waiting Time (AWT) based on edge device resource usage rate [6]. Edge-device computing enables two types of computation offloading mechanisms *partial offloading and Binary Offloading (BO)* [7]. The BO processes are mainly targeted at low level edge devices or middle level edge servers. In *partial offloading*, workloads are segmented and executed at device-level and offloading a part of segmented workload towards the middle or top-level for execution which is unable to execute at device-level. Here, joint communication and computation mechanisms are essential to accomplish the tasks with less latency and less resource wastage.

Recently, Machine Learning (ML) and Artificial Intelligence (AI) frameworks have mainly exploited to deliver effective automated resource sharing. The Q-learning method, as a form of Reinforcement Learning, is used to streamline the offloading policy for computation resources. However,the resource scheduling issue remains as a non-deterministic polynomial (NP)-hard problem [8]. Several frameworks employ Deep-Reinforcement

Manuscript received July 11, 2020; revised October 2, 2020 and November 16, 2020; accepted December 1, 2020. This work was supported by KLEF through FEB Lab, India under Grant KLEF 0279-5949-555-CS5. *(Corresponding author: M. S. Mekala.)*

M. S. Mekala is with the Department of Computer Science, and Engineering, Koneru Lakshmaiah Education Foundation, Vijayawada 522502, AP, India (e-mail: dr.msmekala@gmail.com).

Alireza Jolfaei and Xi Zheng are with the Department of Computing, Macqaurie University, Sydney, NSW 2113, Australia (e-mail: alireza.jolfaei@mq.edu.au; james.zheng@mq.edu.au).

Gautam Srivastava is with the Department of Mathematics, and Computer Science, Brandon University, Brandon, Manitoba R7A 6A9, Canada (e-mail: srivastavag@brandonu.ca).

Amjad Anvari-Moghaddam is with the Department of Energy Technology, Aalborg University, 9220 Aalborg, Denmark (e-mail: aam@et.aau.dk).

P. Viswanathan is with the SCOPE, Vellore Insititute of Technology, Vellore 632014, India (e-mail: viswatry2003@gmail.com).

Digital Object Identifier 10.1109/TETCI.2020.3044082

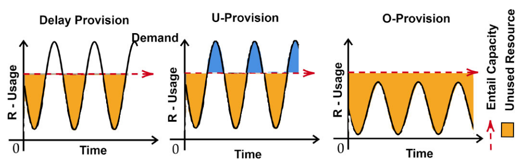


Fig. 1. Resource trade-off analysis.

Learning (DRL) methods for Industrial Internet of Things (IIoT) systems.

Our literature review reveals the following challenges in the Cyber-Physical Systems which utilize edge computing, which we named as Edge Cyber-Physical Orchestration (ECPO):

1) Workload of devices usually are distributed according to their computation and data transmission rate capabilities. However,estimating computation and data transmission rate is challenging [9].
2) Workload allocation and resource scheduling are regulated based on data producing rate. Dealing with data spikes is challenging [10].
3) In cyber-physical orchestration, the service latency caused by task schedulers, has an imminent impact on end-users experience [11].

   Fig. 1 shows a trade-off between cost and latency for reliable service delivery. The figure shows resource allocation under the scenarios of experiencing latency, under-provision, and over-provision

To overcome above-mentioned difficulties, we design and develop a novel reinforcement learning approach to streamline the resource computation offloading and communication issues over cyber-physical orchestration. In summary, the novel contributions are listed below.

1) Implementing an adaptive computation offloading and resource scheduling algorithm to reduce the latency of the Orchestration.
2) Developing the CCR-RL method based on deep-reinforcement learning to identify optimal integration of edge servers with edge devices which accommodates computation service based on resource rate, and deadline violation rate, with machine learning techniques.
3) Developing a DRL algorithm based on the R-retaliation method to reduce the deadline violation rate by scheduling resources during computation offloading at each edge device.

The paper is structured with the following sections. Section II briefly explains research gaps and problems in existing approaches. Section III walks through the proposed system and its mathematical models and also describes our proposed algorithm in detail. In Section V, we evaluate the proposed solution, and Section VI concludes the work.

II. Related Work

In [6], an optimized local computing scheme has designed to manipulate arrival tasks for depreciating latency constraints. In [12], an energy-efficient Mobile Edge Computing (MEC) framework enables an offloading method to execute tasks by assuring low latency constraints to decrease energy usage. Various traditional algorithms have given focus to determining task allocation policy with a time constraint. Consequently, the recent Heterogeneous Earliest Finish Time (HEFT) approach has a gratifying performance; as it decreases the search time. However, it exhibits insufficient versatility towards complex Dynamic Acyclic Graph (DAG) workflows; it merely works to a specific kind of DAG [13], [14]. However, it is appropriate for small-scale charts but not for complicated problems.

The Critical-Path-On-a-Processor (CPOP) approach is a similar approach to HEFT. It evaluates task rank value, and it concedes the sum of two ranks values for sorting the tasks in ascending order and the high-value job defined as a critical task [15], [16]. However, in the mentioned work fluctuations of dynamic resource allocation have not been considered, though the critical task has been assigned to the suitable machine to accomplish low execution time. The proposed solution, however, it is not feasible for multi-objective complex issues.

In [17], a Q-learning based offloading mechanism has been used to determine the difficulty of constant data interchange for executing the tasks, but the system performance has improved slightly. An RL-based work-flow allocation model has been introduced in [18] to determine the DAG task allocation issue. The absence of resource weight retaliation mechanism causes the wrong allocation among devices and exhibits unusual performance latency.

In [19], an RL method has been proposed to determine the computation offloading difficulty by conceding device heterogeneity weight to allocate a task to a device through a DAG to achieve less execution time. Likewise, in [20], a Centralized Learning Distributed Scheduling (CLDS) method has been introduced, which is quite similar to the systems mentioned above.

In [21], [22], a Deep Q-learning Network (DQN) has been introduced based on destination device Q-value to reduce haphazard allocation, which speeds up the training process during large-scale issues. As per the framework, the state value factor individually interprets at each layer, but in the last layer, it migrates to a single element to perform better [23]. The Joint Policy Gradient Method (JCORA) has been used in [24] to determine user-centric difficulties, where various factors asynchronously have been trained to increase circumstance instance and performance latency. In [25], RL-based offloading scheme has been implemented based on computation and energy constraint models to accelerate offload learning. In [26], the author has conceded a multi-X (X refers to a user or a server or cloud) scenario with radio resource computation to optimize vitality usage to achieve a low latency. A dynamic allocation policy has been used to allocate workloads based on the state of execution rate. The workload segmentation, allocation, resource scheduling, and transmission relationship makes an impact on the resource allocation/offloading mechanism of edge-cloud orchestration to reduce both transmission and computation latency. Perversely, the existing works have not collectively conceded the four features.

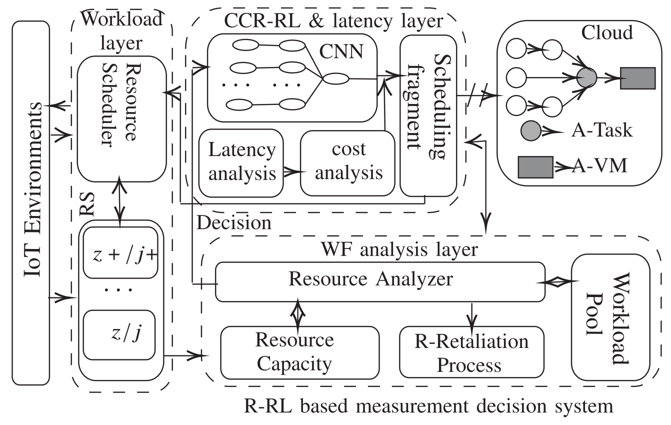


Fig. 2. Deep learning and ECPO architecture.

TABLE I
NOTATIONS

Symbol	Definition
$\Theta_{j,\hat{j}}(\lambda, \tau)$	Amount of instructions required for offloading data from server j to server $\hat{j}$
$X_{\mathcal{Z}}^{\mathbb{R}}$	Average computing time for offloading process from edge devices to the cloud
$Y_{\mathcal{Z}}^{\mathbb{R}}$	Average transmitting time for offloading process from edge devices to the cloud
$r_z^{\mathbb{R}}$	Service execution request from edge device z to edge servers in $\mathbb{R}$
$\mathrm{O}_z^{\mathbb{R}}$	Central Processing Unit (CPU) cycles for executing task from the edge device z to region servers.
η_z	Actual computing capacity of edge device z
$\Phi_z^{\mathbb{R}}$	Workload arrival rate from the edge device z to servers in region $\mathbb{R}$
Λ_z	Amount of processing workload at the edge device z
$\Gamma_{\mathbb{R}}^{z,j}$	Average bandwidth transmission capacity from the edge device z to relayed device j
$\delta(j)$	Computation Capacity of j-th edge server
$a_r^{z,j}$	If it is 1, then workload request r from z has been assigned to j server; otherwise, not assigned indicates 0
$b_{z,j}$	If it is 1, then link (z, j) is in use; otherwise, not in use which indicates 0
$\omega(X_v^{\mathbb{R}})$	Cloud fixed computing time to execute workloads
$\omega(Y_v^{\mathbb{R}})$	total round time to execute workload in cloud; offloaded from j downloaded from cloud.
d, A_o	d indicates distance, A_o indicates constant attenuation.
$\mathfrak{R}$	Edge device requesting resource size in bits.

III. Deep Learning System Model

Deep learning and ECPO architecture are shown in Fig. 2. As shown in the figure, there are three layers: workload layer, deep learning CCR-RL model and latency algorithm layer, and the Work-Flow (WF) analysis layer. The deep learning algorithm layer has two functional modules: R-retaliation function model and CCR-RL model. The R-retaliation function estimates every device workload and its computing capacity. The CCR-RL model is used to select the workload scheduling fragment decision. That is, first the selection of suitable computing device. Next, fetching the entail resource to enhance the device functioning capacity. It should be noted here that a valid option is offloading the workload towards a cloud based solution on the basis of the latency-usage rate. This then will lead to the usage rate of every device updated with regards to the state it belongs to at each layer of RL. If any deviations (rule violation) in terms of the latency constraint happens, an action is triggered. For instance, whether the edge device is capable of handling the task or not. If not, the offloading process would come into the picture to make an accurate decision. It is executing the WL to meet its deadline to avoid latency impact, which falls under the reward section of RL. Subsequently, the WF analysis layer enables multiple hidden layers (for each hooked device), and each layer's neutron (edge device) has its resource usage rate and execution cost factors established.

The deep learning-based Resource Scheduling (RS) algorithm allocates the segmented sub-workload towards a suitable Edge devices or Access Point (AP) based on Service Request (SR) type, data arrival speed, computation, and transmission capacity of devices as the main factors. It also makes use of the prognosticate latency value as well as the usage rate to meet the subscribers' demand. Additionally, it is important to mention here that the WF analysis layer can do a flexible climb up or down the entailed resources per requirement. Table I lists the notations used in the paper.

A. Network Model

Let us denote the AP set by J, where j and $\hat{j} \in J$. A pair $(j, \hat{j})$ has a reinforced relation to ensuring a constant rate of transmission $\gamma(j, \hat{j})$ for every workload. For every access point $j \in J$, it receives workload/request from end-users through an Edge-CPS Orchestration (ECPO) which can be observed in Fig. 3.

Let Z represent the set of edge devices in a zone, where z and $\hat{z} \in \mathcal{Z}$. Each edge device is able to receive

- A Service Request (SR), that is denoted by $r \in \mathcal{R}$, where $\mathcal{R}$ is the set of request services;
- A Computation Offloading (CO) request, that is denoted by $v \in \mathcal{V} = \{1, 2, \ldots, V\}$; and
- A Resource Sharing (RS) request, that is denoted by $m \in \mathcal{M} = \{1, 2, 3, \ldots, M\}$.

The edge device runs a SR at each slot, that is, $|\mathcal{Z}| = |\mathcal{V}| + |\mathcal{M}|$ SRs. A task/SR/workload request (r) is denoted by $r(j, \Delta, \lambda)$, where j represents AP, Δ specifies the essential task accomplishment delay, λ refers to task/workload size and it is measured in Millions of Instructions Per Second (MIPS). Offloading a part of workload $\Theta_{j,\hat{j}}(\lambda, \tau)$ from the workload λ through the link $(j, \hat{j})$ in time window τ requires

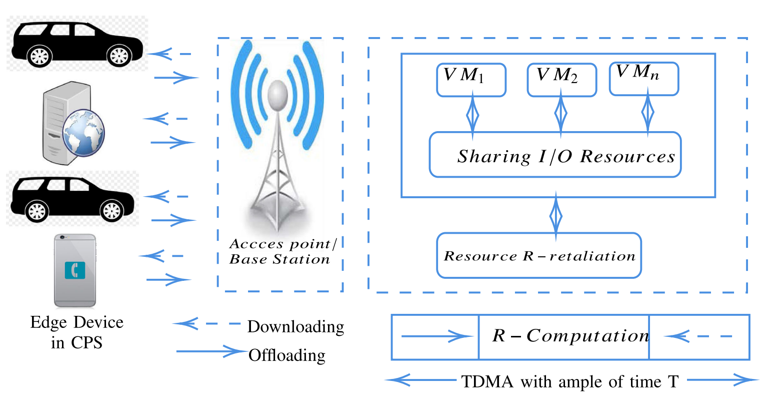
$$\Theta_{j,\hat{j}}(\lambda, \tau) = \gamma(j, \hat{j}) \cdot \lambda \cdot \tau - \Lambda_j \tag{1}$$

instructions, where $\gamma(j, \hat{j})$ represents the link quality between j and $\hat{j}$, and τ represents the time window required for offloading the workload from one access point (server) to one another and Λ_j represents the amount of workload being executed at server.

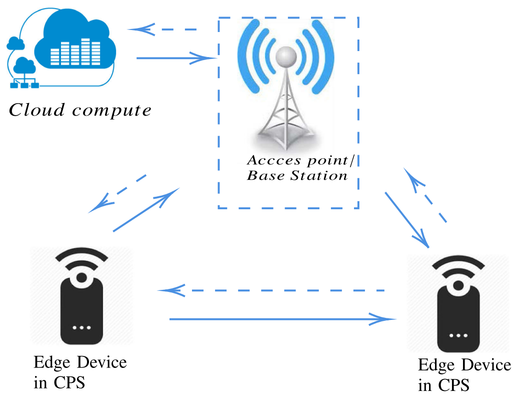
B. Problem Formulation Based on Deep Learning Approach

The rationale of our proposed orchestration is to decrease the latency of the system. We use the following definitions to formulate our deep learning based approach.

Definition 1: The required processing time (computing + transmitting) commencing from the edge device to the Cloud



(a) CCR-RL model based resource sharing



(b) Offloading mechanism

Fig. 3. Edge-cyber-physical orchestration model.

Centre (CC) is named task latency. Let us assume that, I-level receives data from L-level for computation and analysis, and sends back the outcome. Moreover, the unprocessed data is streamed towards the moderate level (cloud level) with a compression rate ∂. The latency $\mathcal{Q}$ of the j-th device in the I-level is evaluated as

$$\mathcal{Q}^{\mathbb{R}}_{j\in\mathbb{R}} = \sum_{j=1,j\in\mathbb{R}}^{|J|} \sum_{z=1,z\in\mathcal{Z}}^{|Z|} z_{\hat{\varpi}} \cdot \hat{\varpi} \cdot \left(X^{\mathbb{R}}_{\mathcal{Z}} + Y^{\mathbb{R}}_{\mathcal{Z}}\right),$$
$$X^{\mathbb{R}}_{\mathcal{Z}} = \frac{r^{\mathbb{R}}_z \cdot \mathrm{O}^{\mathbb{R}}_z}{\eta_z}, \quad Y^{\mathbb{R}}_{\mathcal{Z}} = \frac{0.5 \cdot r^{\mathbb{R}}_z \cdot \Phi^{\mathbb{R}}_z + \Lambda^{\mathbb{R}}_z + r^{\mathbb{R}}_z \cdot \partial \cdot \Phi^{\mathbb{R}}_z}{\Gamma^{z,j}_{\mathbb{R}}}, \tag{2}$$

where $\mathbb{R}$ represents the set of edge servers in a region (I-level). The variable ∂ denotes the data compression ratio.

Let us assume that the base-station or AP bandwidth is split into $\hat{\varpi}$ sub-channels. During each time window τ, the edge device z is allocated a bandwidth $z_{\hat{\varpi}}$ from one of $1, 2, \ldots, \varpi$ sub-channels.

Definition 2: The latency is estimated by the data processing rate from *z-level to c-level*. Hence, for $j = \{1, \ldots, J\}$ and $z = \{1, \ldots, Z\}$, the cloud latency $\mathbb{Q}^{\mathbb{C}}$ is estimated as

$$\mathbb{Q}^{\mathbb{C}} = X^{\mathbb{C}} \cdot \mathbb{Q}^{\mathbb{R}}_{j\in\mathbb{R}}. \tag{3}$$

Definition 3: The edge server has to decide the allocation of sub-channel among connected devices. It ensures a secure sub-channel communication during computation offloading at each server node to optimize the average latency. This is formulated as

$$\begin{aligned} &\underset{RS}{\text{Min}} \sum_{\tau}^{T-1} [f_c(\eta_z(\tau)) + f_c(\Gamma^{z,j}_{\mathbb{R}}(\tau))] \\ &\text{Subj. to} \quad 1. \sum_{z=1,Z\in\mathbb{Z}}^{|Z|} z^{\hat{\varpi}}_z \tau) \cdot \hat{\varpi} < \varpi, \quad z^{\hat{\varpi}} \in \{1, 2, \ldots, \varpi\} \\ &\qquad 2. \sum_{z=1,Z\in\mathbb{Z}}^{|Z|} \xi_z(\tau), \text{ with } \xi_z(\tau) \leqslant \xi_{\max} \\ &\qquad 3. \sum_{z=1,Z\in\mathbb{Z}}^{|Z|} \mathrm{O}_z(\tau)\cdot\hat{\mathrm{O}} < \mathrm{O}, \text{ with } z^{\hat{\mathrm{O}}} \in \{1, 2, \ldots, \mathrm{O}\} \end{aligned} \tag{4}$$

where RS is resource sharing, T is the average total time, $\xi_z(\tau)$ is the energy consumption of edge device at window τ.

Definition 4: The edge server j receives a Work Load (WL), if only if the j has an optimal rate of Computation Capacity (CoC) $\delta(j)$. In case, if the assigned workload on j surpasses $\delta(j)$, then the server j offloads a part of the workload to other AP; otherwise, it offloads the workload to cloud through Eqs. (5)–(9).

$$\begin{aligned} \text{Min} &\sum_{j=1,J\in\mathbb{R}}^{|J|} a_j \cdot \xi_o \cdot t_{pt} + \Phi^j_r \\ &+ \sum_{j=1,J\in\mathbb{R}}^{|J|} \sum_{z=1,Z\in\mathbb{Z}}^{|Z|} b_{z,j} \cdot t_{pt} \cdot \xi^j_z + \Phi^c_r \cdot \xi^c_p \end{aligned} \tag{5}$$

where $\xi_o, t_{pt}, \xi^j_z, \xi^c_p$ represents initial energy consumption, total active time of device, energy consumption for transmitting data from edge device z to edge server j, and energy consumption for data processing, respectively. The workload allocation constraint is greater than $2|R| \cdot a_j$, where

$$2\,|R| \cdot a_j \leqslant \sum_{r=1}^{|R|} \left(\Phi^j_r + \Phi^c_r\right), \tag{6}$$

$$\sum_{r=1}^{|R|} \left(\delta(\Phi^j_r) + \delta(\Phi^c_r)\right) \leqslant \delta^{\mathbb{C}}, \tag{7}$$

where $\delta(\Phi^j_r)$ represents the required computation capacity of the arrival workload rate. The computation constraint for the execution of the workload is

$$\begin{aligned} &\sum_{r=1}^{|R|} \delta(\Phi^j_r) \leqslant \delta(j) \text{ and } \sum_{r=1}^{|R|} \delta(\Phi^z_r) \leqslant \delta(z) \\ &\text{for } j = \{1, \ldots, J\} \text{ and } z = \{1, \ldots, Z\}. \end{aligned} \tag{8}$$

The task constraint is

$$\sum \left(\Phi^j_r + \Phi^c_r\right) \triangleq r \cdot \lambda. \tag{9}$$

C. CCR-RL Based Computation Model

The resource capacity, cost ratio-reinforcement learning (CCR-RL) approach depends on the computation and communication analysis of ECPO system. We deployed a divide and conquer rule to assess the computation offloading towards the device, the edge server, and the cloud based on the resource capacity, cost ratio, through the proposed deep learning mechanism, named the CCR-RL model.

1) Device-Level Computing: The required computing capacity remains balanced by the size of the data processed in the computing device as follows:

$$\eta_z = \partial \cdot r_z \cdot \mathrm{O}_z, \quad \text{therefore, } \eta_z \leqslant \hat{\eta}_z^{\mathbb{Z}}, \tag{10}$$

where $\hat{\eta}_z^{\mathbb{R}}$ refers to the required computing capacity threshold value. Each edge device z transmitting capacity has been regulated based on the transmitting data size, and it might be related to j-th transmission-resource capacity.

$$\Gamma_z^{z+1} \geqslant \frac{1}{R} 0.5 \times \partial \cdot (r_z^{z+1} \cdot \Phi_z^{z+1}),$$
$$\text{therefore, } \sum_{z=1, Z\in\mathbb{Z}}^{|Z|} \Gamma_z^{z+1} \leqslant \Gamma_{\mathbb{R}}^{j} \tag{11}$$

where Γ_z^{z+1} refers to the required transmission-resource capacity of z-th edge device to one other.

2) Server-Level Computing: The required computation capacity to execute the arrival data from the zone-level ($\mathbb{Z}$) is estimated with (12).

$$\delta(\Phi_r^{z,j}) = \frac{1}{R} \Gamma_{\mathbb{Z}}^{z,j} \times \frac{r_z^{\mathbb{R}} \cdot a_r^{z,j} \cdot \Phi_r^{z,j}}{\partial \cdot r_z^{\mathbb{R}} \cdot \Phi_r^{z,j} + \Lambda_z^r},$$
$$\sum_{z=1, Z\in\mathbb{Z}}^{|Z|} \sum_{r=1}^{|R|} \delta(\Phi_r^{z,j}) \leqslant \delta(j) \tag{12}$$

Here, $\delta(j)$ refers to the CoC threshold value of j. The entail CPU cycles/s of j denoted with $\hat{\mathrm{O}}_z^r$. The number of CPU cycles is regulated with CoC and evaluated with (13).

$$\hat{\mathrm{O}}_z^r = \sum_{z=1, Z\in\mathbb{Z}}^{|Z|} \sum_{r=1}^{|R|} r_z^{\mathbb{R}} \cdot \mathrm{O}_z^r \tag{13}$$

3) Cloud-Level Computing: CC receives data from edge servers for computation, and the outcomes are sent back to end-users to meet Service Level Agreement (SLA) for service reliability. Therefore, CC receives data with an ample speed and it estimates as

$$\Phi_r^c = \frac{1}{R} \sum_{z=1, Z\in\mathbb{Z}}^{|Z|} \sum_{r=1}^{|R|} \Gamma_{\mathbb{Z}}^{z,j} \times \frac{r_z^{\mathbb{R}} \cdot a_r^{z,j} \cdot \Phi_r^{z,j}}{\partial \cdot r_z^{\mathbb{R}} \cdot \Phi_r^{z,j} + \Lambda_z^r}. \tag{14}$$

4) Computation Offloading Cost Function: The CoC of the edge server is measured by the amount of CPU cycles/s O, and it is split into Ô blocks. The cost and the usage rate analysis is shown in Fig. 4. The edge server concludes the WL execution, whether it has to be executed on a device or be offloaded to the cloud. The D_r^N plays a significant role in making the WL r decision of execution and is assessed with (15) as follows:

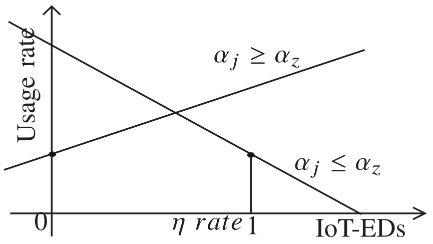


Fig. 4. Cost analysis with various edge-devices versus usage rate.

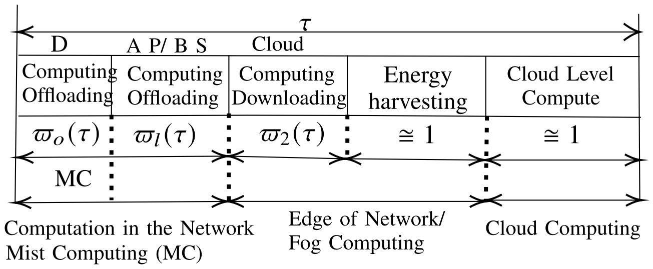


Fig. 5. Channel allocation mechanism for offloading.

$$D_r^V = \begin{cases} 1, & \text{Execute on server} \\ 0, & \text{Execute on cloud} \end{cases} \tag{15}$$

The computation offloading function cost is estimated as

$$\alpha_r^V = \sum_{v=1}^{V} \frac{1}{V} \Big(Y_{\mathcal{Z}}^{\mathbb{R}} + X_{\mathcal{Z}}^{R} \cdot D_r^V + (1 - D_r^V) \times \omega(Y_v^{\mathbb{R}}) \cdot \omega(X_v^{\mathbb{C}}) \Big), \tag{16}$$

where V represents computation request set, $Y_{\mathcal{Z}}^{\mathbb{R}}$ refers to communication time, $X_{\mathcal{Z}}^{R} \cdot D_r^V$ refers to execution cost at edge device, and $\omega(1 - D_r^V) \times (Y_v^{\mathbb{R}}) \cdot \omega(X_v^{\mathbb{C}})$ refers to the execution cost at the cloud.

D. CCR-RL Based Communication Model

Fig. 5 illustrates channel allocation model which includes bandwidth strength attenuation with damage exponent ($\nu \geqslant 2$), where Channel Gain (CG) is renewed at each block time, and is estimated as

$$g = \begin{cases} \hbar\sqrt{A_0 (d_{z+1}/d_z)^{-\nu}}, & d_{z+1} \geqslant d_z \\ \hbar\sqrt{A_0}, \text{otherwise} \end{cases} \tag{17}$$

where $\hbar$ represents the coefficient of Symmetric Gaussian (SG)-[0, 1] and its variance is β^2, A_o represents the constant attenuation.

1) Device-Level Communication: During the initial channel slot, the edge device offloads a portion of data/bits k_r^z, where $k \in K$ bits is sent over to the other device (Edge devices or APs). The data transfer consumes a small amount of power $\varepsilon_j \geqslant 0$. Let $g_0 > 0$ indicate the channel energy gain between z and j, and ϖ represent S-bandwidth (S: Sub-channel). Therefore, the possible data rate (bits/s) between z to $z+1$ or to j is computed with (18)

$$\Omega_z^{z+1} = \varpi_i \log\left(1 + \frac{\xi_z \varepsilon_j \xi_z^{ol} g_0 \varpi_0}{\varpi_i \beta^2}\right), \tag{18}$$

where $0 \leqslant \xi_z \leqslant 1$ indicates the consumed vitality during CO and ϖ_i is the device feeding time. The expected base-band signal (G_z^{z+1}) is

$$G_z^{z+1} = g_i \sqrt{\frac{\xi_z \varepsilon_j \xi_z^{ol} g_0 \varpi_0}{\varpi_i}}. \tag{19}$$

2) S-Level Communication: The possible data rate Ω_z^j between z to j or j to $j+1$ is estimated as follows:

$$\Omega_z^j = 0.5 \times \varpi_i \log\left(1 + \frac{\frac{\varepsilon_j \cdot \varepsilon_{j+1} \cdot g_j \cdot g_{j+1}}{g_{j,j+1} \cdot \varepsilon_{j+1} + \beta_{j,j+1}^2}}{\frac{\varepsilon_j \cdot g_j}{g_{j,j+1} \cdot \varepsilon_{j+1} + \beta_{j,j+1}^2} + \beta^2}\right). \tag{20}$$

The edge server j has a data noise ratio from edge device z over the channel slot and it is

$$H_z^j = \frac{\varepsilon_j \cdot g_j}{g_{j,j+1} \cdot \varepsilon_{j+1} + \beta_{j,j+1}^2} + \beta^2. \tag{21}$$

Accordingly, the possible data rate between j and $j+1$ is estimated as

$$\Omega_j^{j+1} = 0.5 \times \varpi_j \log\left(1 + H_j^{j+1}\right). \tag{22}$$

3) Communication Offloading Cost Function: The ratio of the total number of requests and a distinct resource request count from z is called the RS rate [27], [28]. The RS mechanism is an ongoing research problem, but the resource demand estimation is out of the scope of this research. The resource demand is denoted by $\mathrm{B}_m \epsilon [0,1]$. The cost of resource communication is determined by

$$\alpha_r^M = \sum_{m=1}^{M} \frac{1}{M}\left(\frac{\Re}{r_m^j} + \omega\left(Y_v^{\mathbb{R}}\right) \times (1-\phi_m) \times (1-\mathrm{B}_m)\right), \tag{23}$$

where r_m^j represents the communication request to j, $\frac{\Re}{r_m^j}$ shows the edge device transmission delay, and $\omega(Y_v^{\mathbb{R}}) \times (1-\phi_m) \times (1-\mathrm{B}_m)$ refers to the overall delay-transmission of the demanded resource. Subsequently, if the demanded resource value is high, then the cost value remains became small. If $\phi_m = 1$, then z demands a required resource, which has to be shared by the edge server. Otherwise, $\phi_m = 0$, the edge server retrieves the demanded resource m from the cloud.

IV. Latency Consolidation and Resource Allocation Algorithms

In this section, a latency algorithm and a deep learning-based resource-sharing methodology are proposed as well as a communication algorithm to determine the issues described in the problem formulation Subsection III-B. Our goal is to achieve low latency at all levels, as stated below.

Algorithm 1: Latency Optimization Algorithm.

```
input : 1. Each Device Transmission Capacity $\Gamma_{\mathbb{Z}}^{z,j}$
        2. Computing capacity $\eta_z$
        3. Data arrival speed $\Phi_z^{\mathbb{R}}$
output: Optimal resource scheduling to feasible tasks
1  Let initialize $\Gamma_{\mathbb{Z}}^{z,j}$, $\eta_z^{\mathbb{R}}$, $\Phi_z^{\mathbb{R}}$;
2  for each $z_i = 1$ to $Z$ do
3      $z_i \in Z$;
4      Feasible channel allocation to each request $r_z^{\mathbb{R}}$;
5      Initial allocation of resources based CCR-RL model
         (above listed as per request) with Eq. 24;
6      if $\mathbb{Q}\left(r_z^{\mathbb{R}}\right) \leqslant \mathbb{Q}_{Min}^{\mathbb{C}}$ then
7          $\mathbb{Q}_{Min}^{\mathbb{C}} \leftarrow \mathbb{Q}\left(r_z^{\mathbb{R}}\right)$;
8          Initial allocated resources are sufficient to execute
             the request;
9      else
10         Estimate request resource entail rate, device
             compute and transmission capacity, and their
             cost (Eq. 16) through Eq. 10 to 13 to schedule
             the resources;
11     end
12   end
13 end
14 Optimal resource scheduling to feasible tasks
```

$$\begin{aligned} &\mathbb{Q}^{\mathbb{C}}_{j=1,J\in\mathbb{R} \text{ and } z=1,Z\in\mathbb{Z}} \geqslant \mathbb{Q}^{\mathbb{C}}_{\text{Min}} \\ &\mathbb{Q}^{\mathbb{C}}_{\text{Min}} = X^{\mathbb{C}} \times \sum_{j=1}^{|J|}\sum_{z=1}^{|Z|}\left(X_{\mathbb{Z}}^{\mathbb{R}} + \frac{1}{\Gamma_{\mathbb{Z}}^{z,j}} \times s\right), \end{aligned} \tag{24}$$

where

$$s = \log\left(\sum_{z=1,Z\in\mathbb{Z}}^{|Z|} \sqrt{0.5 \times r_z^{\mathbb{R}} \cdot \Phi_z^{\mathbb{R}} + \Lambda_z + r_z^{\mathbb{R}} \cdot \partial \cdot \Phi_z^{\mathbb{R}}}\right). \tag{25}$$

Algorithm 1 estimates the system latency of every z with non-congested limitations, such as $L_{\min}(z)$. Resource computing and communication scheduling are used for determining the task allocation policy p to achieve low-latency. An optimal resource sharing rate is an important aspect of the computation and communication offloading process, which depends on resource usage upper bound $\Upsilon_{z/j}^u$ and lower bound $\Upsilon_{z/j}^l$. We note here that, the larger value creates a huge Service Level Agreement Violations (SLAVs). Accordingly, lower values might create below usage rates. Therefore, deriving an optimal threshold value remains an internal issue. From a mathematical point of view, if the $\Upsilon_{z/j}^u = 80\%$ and $\Upsilon_{z/j}^l = 55\%$, then the suitable threshold usage factor is estimated as $\Upsilon_{th} = \frac{\Upsilon_{z/j}^l + \Upsilon_{z/j}^u}{2} = 0.675$. Similarly, the allocated value is assessed as $\wp_{z/j} = \aleph_{z/j} \times \frac{1}{\Upsilon_{th}}$. Here, $\aleph_{z/j}$ refers to the resource usage rate of edge device or edge server. The inverse of a suitable threshold usage factor is 1.48. Therefore, at every iteration, the usage rate should be in the range of $(-1,+1)$; the new resources are allocated for the succeeding repetitions, and it is about 1.48 times better than the disbursed value in the prior repetition based on the resource usage factor and the computational capacity of the system in Algorithm 2. Initially, the algorithm checks with its nearest nodes based on the

Algorithm 2: CCR-RL Based Optimization of Resource Allocation and Communication Algorithm.

```
input  : $\Gamma_{\mathbb{Z}}^{z,j}$, $\eta_z$, $\Phi_z^{\mathbb{R}}$
output: Capacity based resource allocation and communication
1  Let initialize $\Upsilon_z = 0$, $\Gamma_{\mathbb{Z}}^{z,j}$, $\eta_z$, $\Phi_z^{\mathbb{R}}$, $\aleph_z$, $\aleph_j$;
2  for each $z_i = 1$ to $Z$ do
3      $z_i \in Z$;
4      $\Upsilon_z = \frac{\aleph_z}{\wp_z} \times 100$;
5      if $\Upsilon_z \geqslant 55 || \Phi_z^{\mathbb{R}} \geqslant 90\,Bytes$ then
6          Do select ED($\therefore \Upsilon_{z+1} \leqslant 55$);
7          else
8              Do select AP or Base-station
                ($\therefore \Upsilon_j \parallel \Upsilon_{j+1} \leqslant 55$) for Offloading $\Theta_{j,\hat{j}}(\lambda, \tau)$ ;
9          end
10     end
11     for each $j_i = 1$ to $J$ do
12         $\Upsilon_j = \frac{\aleph_j}{\wp_j} \times 100$;
13         if $\Upsilon_j \leqslant 55 \parallel \Upsilon_j \geqslant 80$ then
14             Do select near by AP or base-station
                ($\therefore \Upsilon_{j+1} \leqslant 55$) based on Reinforcement
                learning with Bayesian theorem.
15             else
16                 Allocate entail resources from cloud, such
                    as $\hat{\wp}_j = \aleph_j \times 1.48$;
17             end
18         end
19     end
20 end
21 Optimal resource scheduling while offloading process;
```

Euclidean distance factor. If no edge device meets the conditions of Algorithm 2, then Algorithm 2 has to select an edge server to offload the WL for execution.

V. Experimental Results

To examine the efficiency of the ECPO framework, we implemented various workloads to optimize the latency in multiple heterogeneous environments. In the simulation, we used $j = 3$ APs, $z = 25$ edge devices, with the coverage of a single edge router. The workload can be segmented into various sub-workloads for execution at the allocated slot. In our simulation, we segregate the edge devices based on SR value, that is, sub-workload execution/RS execution, at every allotted channel-slot. Let us consider the arrival workload rate $\Phi = \{1, 2, 3, \ldots, 12\}$ at each allotted sub-channel $\hat{\varpi}$. If the spectral bandwidth is $\varpi = 50$, then it will split into 50 sub-channels and each sub-channel bandwidth is $\hat{\varpi} = 1$. Here, the deep reinforcement learning network has 2-hidden layers with 50 neurons. The neuron count and accuracy are related to one another. An increase in the neuron count may cause more difficulty in computing. The used neuron count purely depends on the device capability. In our experiments, we considered 50 neurons.

The CoC is measured by CPU cycles/s count. The communication resources indicate communication bandwidth. We examine the efficiency of three-levels of ECPO with simulation, that is, L-level (zone-level), I-level (region-level), and C-level (cloud level) of the ECPO system. The ECPO system contains a set of devices, which are listed in Table II. In heterogeneous environments, the system topology usually remains constant, and we have to analyse the workload scheduling among three levels. The computing and resource communication capacities of all cases are listed in Table III. We examine the ECPO framework in multiple scenarios. The WL offloads

1) Towards the edge server, or
2) Towards the cloud.

Fig. 6 demonstrates the CoC of the edge server in the range of 60–150 GHz. Our model executes far better than the other two schemes because of the deep learning-based computation offloading constraints during 110 GHz. The j-th device CoC expansion has an impact on the resource sharing process, and its latency is about 2.4 seconds. Enhancing the CoC of j instead of offloading the workload towards $j+1$ exhibits low latency impact, which is essential for reliable communication at 100 GHz. This means extending computing capacity is a better solution compared to offloading at abnormal conditions where the device became potentially strong enough to execute

TABLE II
Network Devices and Capacity Levels

Network Devices and capacity level	L-level	I-level	C-level
Device level (z)	Y	Y	Y
Edge-server level (j)	BS/AP	BS/AP, Switch	BS/AP, Switch, I-Gate-Way
Cloud (c)	Y	Y	Y

TABLE III
Resource Transmission and Compute Capacity of Edge-CPS Orchestration

Device type	Computation capacity (MCPS)	Transmission capacity (MCPS)
Edge device	0.12	-
AP or BS	0.4	1.2
Edge-server (Switch)	1.5	3
Edge-server (I-Gate-Way)	4.2	4.8
Cloud infrastructure	12	12

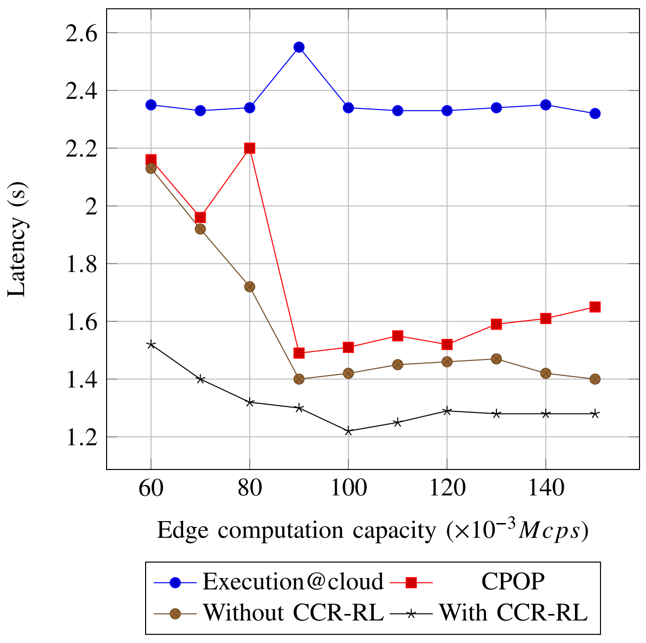


Fig. 6. Latency versus edge compute capability.

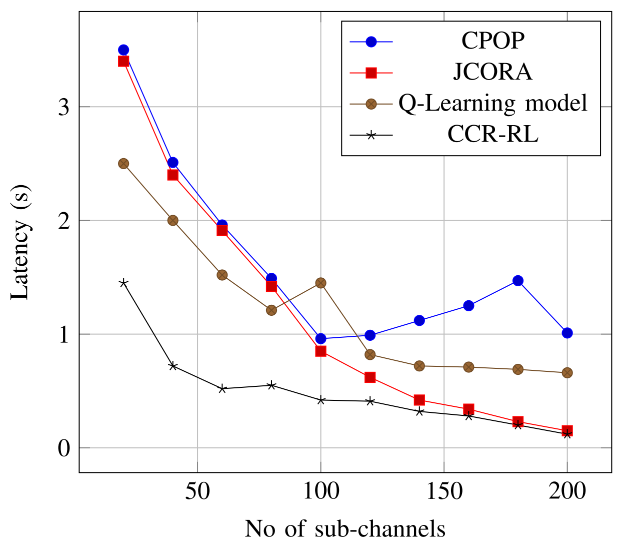


Fig. 7. Latency versus required bandwidth.

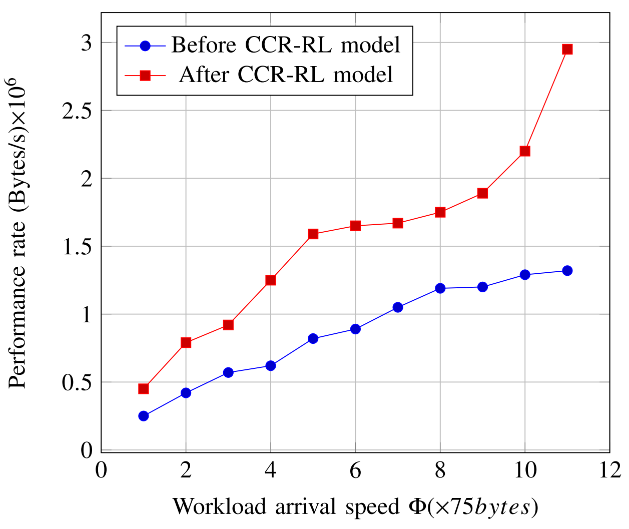


Fig. 9. Performance rate analysis before and after CCR-RL model.

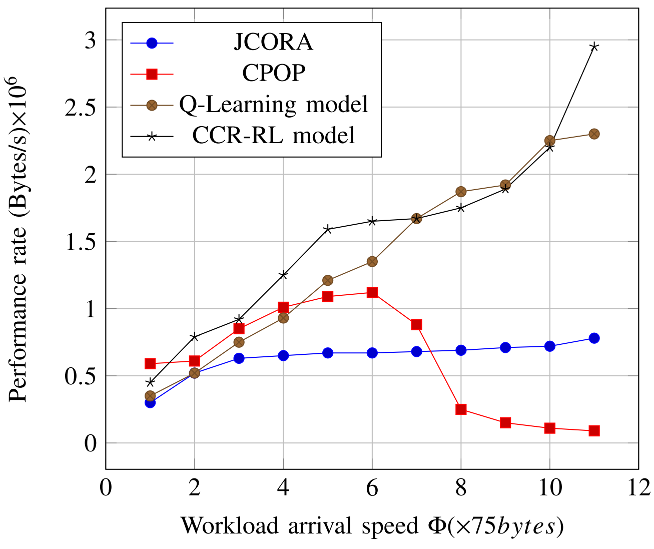


Fig. 8. Workload arrival speed versus performance rate.

TABLE IV
WORKLOAD ANALYSIS WITH VARIOUS PARAMETERS

Workload	RSE rate	Cost rate	Latency rate	Utility rate	SLAV rate
Normal	9.71%	7.24%	4.5%	5.93%	6.99%
Moderate	96.45%	52.65%	69.32%	8.9%	69.81%
High	97.67%	54.75%	75.97%	12.11%	59.01%

the assigned workload. Fig. 7 shows the latency analysis of our system with various extant systems. The latency has diminished significantly because of the expansion of sub-channels quantity between 20–200 with a fixed 1 Hz data transmission rate, which increases the data transfer speed. The ECPO system has a reliable performance because of the deep learning-based, resource retaliation method. It optimizes RS rate with the absence of a resource backup capacity rate. The obtained result shows that the RS strategy, as implemented continuously, brings advantage to the overall system, more specifically during the remote data transmission, where the edge device does not have the sufficient data transmission rate. The CPOP approach did not create any dynamic fluctuations for multi-object complex issues; therefore, during *abnormal* workload where the received work load is higher than the device computation capacity, the latency remains unstable as shown in both Fig. 6 and Fig. 7. Fig. 8 illustrates the performance ratio with various arrival data rates in multiple scenarios. We observe that there is no change in the processing rate as the expectation. The processing rate and arrival speed of $\propto$ are directly proportional. The JCORA and Q-learning models are suitable up to the saturated point; after that, our model has low latency and a high processing rate, mainly when the arrival data rate is enormous. The ECPO framework has accomplished the most noteworthy processing rate because it enables the CCR-RL model, which mutually uses C-and T-resources. Fig. 9 illustrates the analysis of the processing rate of all levels of the ECPO framework with a deep-learning-based CCR-RL model. If our framework has a resource limitation, then it will not allow further processing, which leads to a fall into a saturation point and fails to manipulate the data arrival speed. We observed that level-L and level-I of the ECPO framework have high processing rates although the arrival data speed is more than the transmission capacity. We can also note that the transmission and computing resources usage has immensely enhanced at the I-level of our network. In our simulation, we make use of resource allocation delay and computation offloading delay as a metric for latency estimation between edge devices, servers, and the cloud. The CCR-RL approach uses a resource usage rate and a computation rate for the execution of the arrived workload to avoid latency issues, which is shown in Figs. 6 to 9.

Fig. 10 illustrates the latency rate against the number of edge devices. As shown in the figure, when the latency is low the demand for the equal distribution of transmission and computation resources is low. We observed that in order to meet the user demand, the latency is diminished with an increase in the count j of devices with larger bandwidth and sub-channels. Table IV shows the outcomes of workload analysis with respect to parameters: RSE rate, cost rate, latency rate, utility rate, and Service Level Agreement Violation (SLAV) rate. We observe

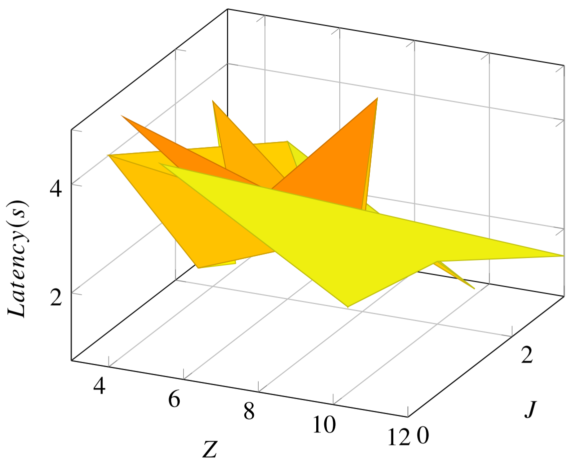


Fig. 10. Latency analysis with various edge devices and edge-servers, where Z refer number of edge devices and J refer number of edge servers.

TABLE V
RESOURCE CAPACITY PARAMETERS IN FOGSIM SIMULATION

Parameter	C-level	I-level	L-level
RAM Gb	64	8	4
Bandwidth (Mbps)	10000	1000	10
PE count	32	16	8 or 4
MIPS	100000	30000	3000

TABLE VI
PARAMETERS

Workload capacity (million instructions per second)	Worth	Sub workload capacity	Worth
workload length	Vary from 500-50000	sub workload length	vary from 250-25000
workload arrival rate	0-12	cloud delay	90 to 150 ms
connected sensors	15	server delay	25 to 75 ms
workload size	vary 8000 and more	workload size	vary 1000 and more

that the usage of the assets is high in H-W (high workload) scenario compared to the rest of *Normal and Moderate* work load scenarios. The SLAVs are also high during higher work loads. Accordingly, the asset usage and residual work load is higher in the normal work load compared to the moderate workload; similarly, the asset usage and residual work load is higher in low SLAVs. In moderate work load, our approach achieved low parameter values compared to other models by 96.45% of RSE, 52.62% of the cost, 69.32% of latency, 8.9% of usage, 69.81% of SLAVs respectively. Even at high workload, RSE, cost, latency, usage, and SLAV of our approach have achieved a rate of 97.67%, 52.65%, 75.97%, 12.11%, 59.01%, respectively. In our approach, during the high workload, the resource scheduling rate is far better than other models.

A. Simulation Parameters

In our simulations, we used 25 different application environments. Table V shows the asset attribute values of three levels (C-level, I-level, L-level) of the framework. Table VI lists other essential attributes of our simulation.

B. Performance Indexes

The performance rate of our approach is measured by combining two measurement indexes, that is, $DV + PDM$. The average deadline violation rate, that is, the number of failed tasks divided by total assigned tasks, is estimated by using a resource ratio of the device. The Deadline Violation (DV) is computed as

$$DV = 0.5 \times \left(\sum_{z=1}^{Z} \frac{\overset{\frown}{R}^{z}}{R^{z}} + \sum_{j=1}^{J} \frac{\overset{\frown}{R}^{j}}{R^{j}} \right) \times 100, \tag{26}$$

where R_z and R_j represent the total SRs of z-th edge device and edge servers, respectively. $\overset{\frown}{R}^{z}$ refers to the number of deadline violated SRs of workload.

The Performance Degradation Measurement (PDM) is assessed based on the required computing capacity of each device as follows:

$$PDM = \left(\frac{1}{Z} \times \sum_{z=1}^{Z} \frac{\eta^{z}}{\overset{\frown}{\eta}^{z}} + \frac{1}{J} \left(\sum_{j=1}^{J} \frac{\Phi^{j}}{\delta(j)} \right) \right) \times 100. \tag{27}$$

VI. CONCLUSION

The proposed approach has achieved low-latency for all service requests. We have considered an uplink communication mechanism from low-level devices to the edge. The arrived workload is scheduled towards edge devices, edge servers/access points/base stations, and cloud-based servers according to the respective capacity levels as well as the device computation capacity. The proposed algorithms have achieved a high processing rate and low latency rate by considering the ratio of workload allocation, transmission, and computation resource usage. In this paper, we provide an optimal solution for the underlying problems of resource utilization and work allocation on eligible edge devices. Through mathematical and theoretical analysis, we developed an ECPO framework. The CCR-RL model simulation outcomes exhibit a good balance between performance and latency rate; our solution reduces the average latency by 49.93 % and improves the resource sharing efficiency by 67.94 %. It also reduces the average cost by 38.2% less than the state-of-art approaches.

REFERENCES

[1] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and A. Polakos, "A comprehensive survey on fog computing : State- of-the-art and research challenges," *IEEE Commun. Surv. Tut.*, vol. 20, no. 1, pp. 416–464, Jan.-Mar. 2018.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1628–1656, Jul.-Sep. 2017.

[3] M. Ghobaei-Arani, S. Jabbehdari, and M. A. Pourmina, "An autonomic approach for resource provisioning of cloud services," *Cluster Comput.*, vol. 19, no. 3, pp. 1017–1036, Sep. 2016. [Online]. Available: https://doi.org/10.1007/s10586-016-0574-9

[4] M. S. Mekala and P. Viswanathan, "A survey: Energy-efficient sensor and vm selection approaches in green computing for x-iot applications," *Int. J. Comput. Appl.*, vol. 42, no. 3, pp. 290–305, 2020.

[5] F. Xu, H. Zheng, H. Jiang, W. Shao, H. Liu, and Z. Zhou, "Cost-effective cloud server provisioning for predictable performance of big data analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1036–1051, May 2019.

[6] D. Cui, Z. Peng, J. Xiong, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for grid or iaas cloud," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1030–1039, Oct.-Dec. 2020.

[7] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[8] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[9] J. Xu *et al.*, "Efficient and lightweight data streaming authentication in industrial automation and control systems," *IEEE Trans. Ind. Inform.*, 2020. doi: 10.1109/TII.2020.3008012.

[10] T. Wang *et al.*, "An intelligent dynamic offloading from cloud to edge for smart iot systems with big data," *IEEE Trans. Netw. Sci. Eng.*, 2020. doi: 10.1109/TNSE.2020.2988052.

[11] K. Huang *et al.*, "Hucdo: A hybrid user-centric data outsourcing scheme," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 3, pp. 1–23, 2020.

[12] M. S. Mekala and P. Viswanathan, "Energy-efficient virtual machine selection based on resource ranking and utilization factor approach in cloud computing for iot," *Comput. Elect. Eng.*, vol. 73, pp. 227–244, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790618315611

[13] M. Duan, K. Li, X. Liao, and K. Li, "A parallel multiclassification algorithm for big data using an extreme learning machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2337–2351, Jun. 2018.

[14] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[15] K. Li, J. Mei, and K. Li, "A fund-constrained investment scheme for profit maximization in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 11, no. 6, pp. 893–907, Nov.-Dec. 2018.

[16] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[17] A. I. Orhean, F. Pop, and I. Raicu, "New scheduling approach using reinforcement learning for heterogeneous distributed systems," *J. Parallel Distrib. Comput.*, vol. 117, pp. 292–302, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731517301521

[18] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[19] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[20] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[21] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-uav enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018.

[22] M. S. Mekala and P. Viswanathan, "Equilibrium transmission bi-level energy efficient node selection approach for internet of things," *Wireless Pers. Commun.*, vol. 108, no. 3, pp. 1635–1663, Oct. 2019. [Online]. Available: https://doi.org/10.1007/s11277-019-06488-7

[23] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[24] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[25] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for iot devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[26] C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, 2017.

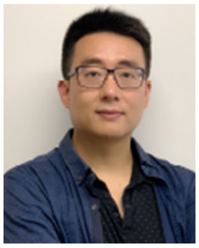
[27] J. C.-W. Lin, Y. Shao, Y. Djenouri, and U. Yun, "Asrnn: A recurrent neural network with an attention model for sequence labeling," *Knowl.-Based Syst.*, 2020, Art. no. 106548.

[28] J. C.-W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, and M. Aloqaily, "Privacy preserving multi-objective sanitization model in 6g iot environments," *IEEE Internet Things J.*, 2020. doi: 10.1109/JIOT.2020.3032896.

M. S. Mekala received the Ph.D. degree in cloud-integrated IoT from VIT University, India. He is currently working as a Research Coordinator and member of FEB Lab with KL University. He specializes in service computing, edge computing, CPS, IoT communication, and reliability analysis.

Alireza Jolfaei (Senior Member, IEEE) received the Ph.D. degree in applied cryptography from Griffith University, Gold Coast, Australia. He is leading the Master of IT in Cyber Security program with the Department of Computing, Macquarie University, Sydney, Australia.

Gautam Srivastava (Senior Member, IEEE) received the B.Sc. degree in mathematics and computer science from Briar Cliff University, Sioux City, IA, USA, in 2004, and the M.Sc. and Ph.D. degrees in computer science from the University of Victoria, Canada, in 2006 and 2012, respectively. In 2014, he joined a Tenure Track position with Brandon University, Brandon, MB, Canada, where he is currently an Associate Professor.



Xi Zheng received the Ph.D. degree in software engineering from the UT Austin. He specializes in service computing, IoT Security, and reliability analysis. He was the recipent of the Deakin University's Outstanding Research Award in 2016. He actively reviews for top journals and conferences. He is currently with the Department of Computing at Macquarie University, Sydney, Australia.

Amjad Anvari-Moghaddam (Senior Member, IEEE) received the Ph.D. degree (Hons.) in power systems engineering from the University of Tehran, Tehran, Iran, in 2015. He is currently an Associate Professor with the Department of Energy Technology, Aalborg University, where he is the Head of the Integrated Energy Systems Laboratory (IES-Lab).

P. Viswanathan received the B.E. degree in CSE from Madurai Kamaraj University in 2002, the M.E. degree in CSE from Annamalai University in 2006, and the Ph.D. degree from VIT University in 2014. He is an Associate Professor with VIT University, Vellore, India. His research interests include IoT, cloud computng, and digital image processing.