Computers, Materials & Continua
DOI: 10.32604/cmc.2023.033194
Article

Optimizing Service Stipulation Uncertainty with Deep Reinforcement Learning for Internet Vehicle Systems

Zulqar Nain[1], B. Shahana[2], Shehzad Ashraf Chaudhry[3], P. Viswanathan[4], M.S. Mekala[1] and Sung Won Kim[1,*]

[1]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan-si, Korea
[2]Department of Computer Science and Engineering, KLEF, India
[3]Department of Computer Engineering, Faculty of Engineering and Architecture, Istanbul Gelisim University, Istanbul, 34310, Turkey
[4]School of Computer Science and Engineering, VIT University, Vellore, India
*Corresponding Author: Sung Won Kim. Email: swon@yu.ac.kr
Received: 10 June 2022; Accepted: 22 September 2022

Abstract: Fog computing brings computational services near the network edge to meet the latency constraints of cyber-physical System (CPS) applications. Edge devices enable limited computational capacity and energy availability that hamper end user performance. We designed a novel performance measurement index to gauge a device's resource capacity. This examination addresses the offloading mechanism issues, where the end user (EU) offloads a part of its workload to a nearby edge server (ES). Sometimes, the ES further offloads the workload to another ES or cloud server to achieve reliable performance because of limited resources (such as storage and computation). The manuscript aims to reduce the service offloading rate by selecting a potential device or server to accomplish a low average latency and service completion time to meet the deadline constraints of sub-divided services. In this regard, an adaptive online status predictive model design is significant for prognosticating the asset requirement of arrived services to make float decisions. Consequently, the development of a reinforcement learning-based flexible x-scheduling (RFXS) approach resolves the service offloading issues, where x = service/resource for producing the low latency and high performance of the network. Our approach to the theoretical bound and computational complexity is derived by formulating the system efficiency. A quadratic restraint mechanism is employed to formulate the service optimization issue according to a set of measurements, as well as the behavioural association rate and adulation factor. Our system managed an average 0.89% of the service offloading rate, with 39 *ms* of delay over complex scenarios (using three servers with a 50% service arrival rate). The simulation outcomes confirm that the proposed scheme attained a low offloading uncertainty, and is suitable for simulating heterogeneous CPS frameworks.

Keywords: Fog computing; task allocation; measurement models; feasible node selection methods; performance metrics

1 Introduction

Increasing the deployment of Internet of Things (IoT) devices and maintaining the generated data is a hectic challenge [1]. In this regard, cloud data centres provide reliable services for accomplishing it. Nevertheless, several drawbacks show an impact in terms of latency network congestion, leased service cost, and especially geographical distance, which causes the unavailability of cloud resources [2,3]. The fog computing paradigm enlarges the services to the network edge to overcome all the disadvantages listed above. A conventional fog computerized framework contains a group of distributed fog computing entities (fog nodes) that are geographically connected with limited resources. Fog nodes are frequently systematized over every level in the form of a tiered architecture, depending upon their length of space to IoT equipment. Resource-restricted devices may use an offloading mechanism to forward part of their workloads towards a nearby edge server (ES)/node through the communication channel. This process minimizes vitality wastage and increases the performance of the system [4,5].

Motivation: Technically, fog computing becomes an intermediate layer for fulfilling the requirements of the IoT one by inheriting certain cloud services to improve the framework performance in terms of resource usage, as well as the equilibrium of the arrived load, in addition to service latency. Usually, most existing service offloading methods select the non-overloaded servers due to the high resource availability to achieve high service reliability. As a result, the deadline-sensitive application requirements can be violated due to the long queue length and excessive waiting time. Therefore, one possible solution is to divide the task into sub-tasks, which are preferred for executing the arrived workload parallel through the inspiration of the divide-and-conquer rule. Two fascinating research questions may arise with a trade-off between power and latency.

1. To what extent will the workloads be offloaded towards consecutive fog tiers?
2. What allocation and estimation of resources will be needed to execute the arrived services? That concludes the offload decision.

Making the right time decision about these questions is crucial but still ambitious because of time variations in wireless systems, as well as unreliable offloading and unrecognized data traffic [6,7].

Stable service allocation in IoT systems is a Herculean task [8]. If the device moves to another location, there will be a change in wireless channel conditions, leading to more time being required for wireless transmission and degrading to achieve the quality of service (QoS) [9,10]. Furthermore, IoT errands are distinct from various locations that may demand different computing assets. Hence, the decisions on errand allocation are to be altered according to the required resources.

Generally, the active device location is not static, and that generates multiple computing errands at every location, subsequently offloading all computation-intensive service requests (SRs) to nearby edge servers (ESs). Here, the ES manager is responsible for taking all decisions related to SR allocations. Estimating the future position based on forthcoming SRs is uncertain and unrealistic to optimize. Hence, an optimized asset-intensive reinforcement method is required to make SR allocation decisions according to the modern status of the network [11] to address SR allocation issues. Moreover, due to the restricted battery capacity, every device's energy consumption between various points is vital. In particular, the greater vitality consumption of the present location may strip energy for the next processes as a result [12,13]. Therefore, we developed a deep reinforcement learning approach to streamline the above-listed challenges to formulate errand allocation decisions per the demand by designing an online service offloading algorithm. The significant contributions of our work are listed below.

1. Designing and developing a reinforcement learning-based flexible x-scheduling (RFXS) approach to resolving the service offloading issues according to a reinforcement-based errand allocation method and probabilistic resource requirement analysis model, which includes a set of node-centric measurements, as well as the behavioural association rate and adulation factor.
2. Constructing a reinforcement-based errand allocation method to offload computation-intensive services to various ESs to depreciate the average errand completion time by meeting related constraints.
3. Developing a probabilistic resource requirement analysis model based on future node-centric resource requirements and uncertain local computing information to increase the performance of the IoT-fog system.

The remainder of this article is organized as follows. In Section 2 we review related works. Then, we describe the IoT-reinforcement learning-based flexible task and resource scheduling (RFTRS) architecture for errand and resource allocation in Section 3. Section 4 describes the end user (EU) demand and resource usage rate attribute-based deep reinforcement learning method. The RFTRS scheme and feasible node selection algorithm are introduced in Section 5, and the evaluated performance of the proposed scheme is described in Section 6. Finally, we conclude this article in Section 7.

2 Related Work

This section outlines the research gap correlations and contrasts existing works. Researchers must optimize task allocation and asset scheduling issues in IoT environments with various probability methods. In addition, they must concentrate on scaling the assets up and down as per demand with the design of a novel asset policy that should meet multiple research objectives. In [14], the analysis introduced asset scheduling approaches over a clustering mechanism with a distributed environment to optimize the resource usage. In [15,16], the investigators designed an asset provision model through a token-based service method for a cloud data centre (CDC) to enhance the service execution reliability. The developed method is intended to meet user requirements according to a planned scheduling policy to accommodate the services at less cost. In [17], the study advanced a content-aware mapping method which allocates the virtual machines (VMs) according to a bandwidth-intensive rate. To overcome the consequences of a server perspective policy, the author has constructed a resource allocation method to diminish asset conflicts using graph theory in [18], and also streamlined the VM allocation problem through the asset equilibrium mechanism.

In [19], the researchers streamlined the workload allocation problem of an optimized multi-objective system by conceding the service reliability rate to diminish the service cost with less completion time, and also designed a co-operative method to regulate the asset provisioning issue based on game theory. However, previous works have not examined the task scheduling issues of active IoT devices in IoT-fog networks. In [20], the author has focused on optimizing the impact of fault restoration on service performance by considering the VMs' track record (recovery and communication rate) outcome. However, none of the explained methods resolved fog-service request scheduling issues over IoT environments.

In [21], the study developed an efficient data loss fetching model as well as a practical path estimation one for the IoT framework and concentrated on designing a reliable leased cost system. In [22], the author has created a design to estimate the nodes' flexibility by conceding the statistical analysis of previous node failures with log data. In [23], the researcher has used a smart mobile as a node to accommodate computing assets at the edge with adaptive architecture to accomplish a reliable

service system. However, the above methods are concentrated on the asset failures of the IoT-fog system and not service offloading issues.

The offloading mechanism resolution enables flat offloading and upright offloading/collaboration towards the IoT-fog environment. Most edge servers execute the accommodated data but use flat/upright collaboration. However, a few investigations have been conducted with nearby nodes using flat collaboration and upright partnership towards cloud servers in [24] to achieve low latency. In [25], the authors have designed flat and upright collaborations through novel asset allocation methods to diminish the service request (SR) execution time. In their investigation, waiting queue time has not been considered because of queue maintenance by the ES manager. Furthermore, various delay constraints are vital in diminishing delays by regulating transmission and computing-related asset distribution. The communication delay issues among deployed nodes were conceded in [26], but they were missed when concentrating on multi-user scenario issues. In [27], a model was devised to minimize the performance delay by optimizing queue delay, which plays a vital role during the computation offloading mechanism in multi-tenant scenarios (which receives both end tasks) in a fog environment.

The $\propto$-distribution method has been developed to address multi-path resource allocation issues over a software-defined network [28]. This approach scaled the problem size to exploit the computation power of the controller, and the cost-based switching of the allocation streamlined the system's accuracy. A predictive offloading and resource allocation (PORA) system has been created to address the service offloading issue derived from the traffic prediction of fog servers, in an attempt to minimize the energy consumption of the system [29]. A task selection-task allocation (TS-TA) method was established to effectively allocate resources using a genetic algorithm. The author attempted to minimize resource usage by balancing the server's workload [30] However, the server's current status was neglected in the above models while assisting the arrived services. In addition, none of the methods failed to concede queue delay, mainly in fault collaboration with nearby nodes. In [31,32] a mobility-aware security dynamic service composition (MSDSC) model-based framework has been created to meet the requirements of deadline-sensitive applications through the restricted Boltzmann mechanism. The suggested probabilistic models assess the process at each phase for effective service execution. In [33], a novel communication model has been outlined to make a trade-off between cost, deadline, and relevant constraints for accomplishing highly reliable services with a low latency rate, and an energy usage optimization design has been described as an extension of this contribution in [34]. In [35], a novel resource allocation model has been formed according to a deep-learning Q-network to optimize the system latency and energy consumption to meet the deadline within the demanded resource capacity with a 30% reduced cost in terms of energy and execution time.

To facilitate responsible fog-IoT networks, our previous work [36] discusses the fog asset scheduling over IoT frameworks by contemplating VM flops to reduce the service execution cost. The trade-off between the task allocation cost and reliability has never been examined. In this document, we designed a multi-objective RFTRS mechanism to optimize the uncertainty in the service offloading scheme by examining the trade-off between reliable errand allocation, asset scheduling rate and cost, with a deep reinforcement learning system.

3 System Framework

The proposed fog-cloud RFTRS architecture enables a cloud server, edge servers (ESs) and EU, as depicted in Fig. 1. The ES and EU sets are denoted as $J = \{1, \ldots, J\}$ and $Z = \{1, \ldots, Z\}$, respectively. The edge servers (ESs) and end users (EUs) are deployed randomly over the monitoring area. However, we consider different application workflows (AWs), and the set is denoted as $W = \{1, \ldots, W\}$, to make

sure that every AW demands several central processing unit (CPU) cycles to execute the service requests (SRs)/tasks. Note that each EU can trigger one SR at a time and Table 1 shows the list of notations.

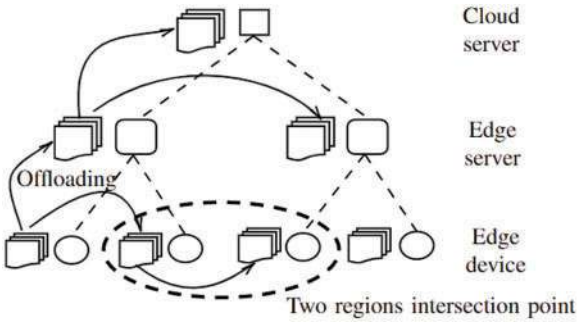

Figure 1: System model

Table 1: Table of notations

Notation	Definition
$\rho_{z,j}$	Content or service offloading probability
∂_z	Service arrival rate at device z
k_{iz}	Expected service completion time at z
v_i^j	Amount of i^{th} task data executed by server
θ_{zj}^w	Processing adulation factor
Δ	Probability of resource scarcity
ξ_{iz}^j	Energy usage to execute i^{th} service request by the server j
$com_{z,j}$	Data transmission time from z^{th} device to z^{th} server
d_{iz}^j	Million instructions per second (MIPS) or CPU cycles required by the server to execute the task

3.1 User demand (UD) Entail Attribute Queuing Analysis in AW Allocation

Initially, the EU resources are used to perform the allocated SR at the time interval $[\tau, \tau+1)$. However, the EU may not execute and meet the SR deadline due to the huge demand for CPU cycles since the edge user is resource-limited. Thus, the EU has to offload a portion of SRs to nearby potential ESs. Let us consider that each EU scheduler enables two queues (local execution and offloading) and functions simultaneously. It is essential to measure the probability of offloading the content to the nearby j^{th} ES.

Consider that $\rho_{z,j}$ is the probability of content offloading between the z^{th} EU and j^{th} ES. Here, $\partial_z^0 = \rho_{z,j}.\partial_z$ represents the SR arrival rate of the z^{th} EU offloading queue and $\partial_z^{lc} = \left(1-\rho_{z,j}\right).\partial_z$ the SR arrival rate of the z^{th} EU local computing queue. The expected SR completion time must be constrained by its deadline, which is evaluated with Eq. (1).

$$k_{iz} = k_{iz}^{com} + k_{iz}^{pro} = a_{iz}^j \times \left(\sum_{j=1}^{J}\left(\frac{b_{iz}}{com_{z,j}} + \frac{b_{iz}.\partial_{iz}}{f_j}\right)\right), \therefore k_{iz} \le k_z^W \tag{1}$$

where $k_{iz}^{com} = a_{iz}^j \times \frac{b_{iz}}{com_{z,j}}, k_{iz}^{pro} = a_{iz}^j \times \frac{b_{iz}.\partial_{iz}}{f_j}$ represents the communication and computation delay, respectively.

3.2 Identifying Potential Nodes

At first, we estimate the behavioural association rate $bar_{z,z+1}(\tau)\, ofz, z+1$ at the time slot with Eq. (2).

$$bar_{z,z+1}(\tau) = \frac{\prod_{Cor_z \in Cor_z^j} act_z^{\tau} \times act_{z+1}^{\tau}}{\sqrt{\prod_{Cor_z \in Cor_z^j} \left(act_z^{\tau}\right)^2} \times \sqrt{\prod_{Cor_z \in Cor_z^j} \left(act_{z+1}^{\tau}\right)^2}} \tag{2}$$

The feasibility factor of the EU is estimated by considering the bar value at every slot, with the below Eq. (3). It plays a vital role in avoiding the redundancy and proficiency of the EU.

$$F_z(\tau) = \frac{\sum_{Cor_z \in Cor_z^j} bar_{z,z+1}(\tau) \times act_{z+1}^{\tau}}{\sum_{Cor_z \in Cor_z^j} bar_{z,z+1}(\tau)} \tag{3}$$

where act_{z+1}^{τ} refers to the activity of the current devices at time slot τ. The targeted device accessing probability $\rho(z|z+1)$ is estimated by a weighting factor function $dist(z|z+1)$ with Eq. (4). It is used to analyse the targeted node feasibility at a given slot, and this value will be employed to assess the completion time of the ED or ES by allocating the AW.

$$\rho(z|z+1) = \frac{dist(z|z+1)}{\sum_{Cor_z \in Cor_z^j} (dist\,(z|z+1), b_{iz})} \tag{4}$$

Moreover, we examine some particular features of concern during time slots, so some of the users may use the adulation factor without examining distance. The adulation factor is evaluated with Eq. (5), where α refers to weight between current and long-time adulation value. The EU adulation factor cross-verification matrix is represented in Eq. (6).

$$adu_{z|z+1} = \alpha \times \frac{\left|M_z^{\tau}\right|}{\sum_{Cor_z \in Cor_z^j} \left|M_{z+1}^{\tau}\right|} \tag{5}$$

$$M_{24\times z}^{\tau} = \begin{pmatrix} m_{11} & m_{12} & m_{1z} \\ \vdots & \ddots & \vdots \\ m_{241} & m_{242} & m_{24z} \end{pmatrix} \tag{6}$$

4 Problem Formulation

The system model concentrates on formulating task scheduling issues with minimized time through demand-based resource sharing over the IoT system. For instance, the EU (for example, a camera) forwarded b bits/second to the j^{th} ES. Normally, not all SRs offload towards the cloud or nearby ESs. When the j^{th} ES permits b_i, z, that is $\delta_i^j = v_i^z \times b_{iz}.\delta_i^j$ refers to the amount of data executed by the z^{th} EU and v_i^z that by the j^{th} ES. In some cases, the ES might forward a portion of the workload to nearby nodes when devices are potentially able to accommodate the workload, or else it decides to offload towards the cloud. This means $v_i^z + \sum_{j=1}^{J} v_{j,j+1} \times b_{iz}$ amount of data has been executed from the edge side. In this scenario, the following conditions and definitions should be considered.

Definition 1: Suppose the j^{th} ES accepts accommodating the workload, although it has to manage local sub-workloads in the pipeline to execute or offload to another nearby node, or it may offload to the cloud. In this scenario, it may consume time, the right way to accomplish the allocated workload—the ready time ($\gamma_{j,j+1}^i$) described as the most prime time, while the whole pressing antecedents of it ought to complete the execution in Eq. (7).

$$\gamma_{j,j+1}^{i} = \max_{\hat{i} \in pred(v_i^z)} \left\{ max \left\{ \wp_{i\hat{j}}^{j}, \wp_{i\hat{c}}^{j}, \wp^{com}(v_i^z) \right\} \right\} \tag{7}$$

Where $\wp_{i\hat{j}}^{j}, \wp_{i\hat{c}}^{j}, \wp^{com}(v_i^z)$ refers to the completion time of the ES, cloud and completion time to transmit the data respectively. But i has three cases.

Case 1: if $\wp^{com}\left(v_i^z\right) < \wp_{i\hat{j}}^{j}$ *and* $\wp^{com}\left(v_i^z\right) < \wp_{i\hat{c}}^{j}$

Then the ready time is to be Eq. (8)

$$\gamma_{j,j+1}^{i} = \max \left\{ \left(1 - x_{i\hat{j}}\right) \times \wp_{i\hat{j}}^{j} + x_{i\hat{j}} \times \wp_{i\hat{c}}^{j} \right\} \tag{8}$$

Case 2: if $\wp^{com} \geq \wp_{i\hat{j}}^{j}$ and $\wp^{com}\left(v_i^z\right) \geq \wp_{i\hat{c}}^{j}$

Then the ready time is to be Eq. (9)

$$\gamma_{j,j+1}^{i} = \wp^{com}\left(v_i^z\right) \tag{9}$$

Case 3: if $\wp^{com}\left(v_i^z\right) | \{\wp_{i\hat{j}}^{j}, \wp_{i\hat{c}}^{j}\}$ then the ready time is to be Eq. (10)

$$\gamma_{j,j+1}^{i} = Max \left\{ \{\wp_{i\hat{j}}^{j}, \wp_{i\hat{c}}^{j}\} \right\} \tag{10}$$

Definition 2: The completion time of a SR on the j^{th} ES. Where $(j \neq j{+}1)$ is defined as a summation of ready time and computation time and is estimated as Eq. (11).

$$\varphi_{j,j+1}^{i}\left(v_j^{j+1}\right) = \gamma_{j,j+1}^{i}\left(v_j^{j+1}\right) + x_{j,j+1}^{i}\left(v_j^{j+1}\right) \tag{11}$$

Case 1: If there is no transmission wait queue during local execution on ES, then the sub-SR finish time is estimated by Eq. (12).

$$\varphi_{i,z}^{trw}\left(v_i^z\right) = \wp^{com}\left(\wp_i^j\right) + x_{j,z}^{i}\left(v_i^z\right) \tag{12}$$

Case 2: Similarly, the cloud server requires some time to execute the received SR, as estimated by Eq. (13)

$$\varphi_{j,c}^{i}\left(v_j^c\right) = \gamma_{j,c}^{i}\left(v_j^c\right) + x_{j,c}^{i}\left(v_j^c\right) \tag{13}$$

Definition 3: The computation time on each ES is described as the summation of computation latency $\frac{\wp_i^j}{2_{oj}\left(o_j - \wp_i^j\right)}$, the delay to fetch entail data to execute the SR $\wp_i^j \frac{1}{o_j}$, required to complete the sub-task on ES $\left(x_{j,j+1}^{i}\right)$ and is evaluated with Eq. (14)

$$x_{j,j+1}^{i}\left(v_i^z\right) = \frac{\wp_i^j}{2_{oj}\left(o_j - \wp_i^j\right)} + \frac{1}{o_j} + x_{j,j+1}^{i} \tag{14}$$

where $x_{j,j+1}^{i} = \frac{\wp_i^j}{\mu_j}$

Now, optimizing the latency while sharing the resources among ESs accomplishes a balanced resource usage. Therefore, the manuscript anticipation is defined as Eq. (15)

$$Avg \min_{j,v} \left(Max\left(\varphi_{j,j+1}^{i}\left(v_j^{j+1}\right), \varphi_{i,z}^{trw}\left(v_i^z\right), \varphi_{j,c}^{i}\left(v_j^c\right)\right) + \sum_{z=1}^{Z}\sum_{j=1}^{J} k_{iz,j}^{com} \right) \tag{15}$$

$$s.t v_i^z + \sum_{j=1}^{J} v_j^{j+1} + v_j^e = 1$$

where $\sum_{z=1}^{Z} \sum_{j=1}^{J} k_{iz,j}^{com}$ refers to the total communication delay to execute the task.

5 RFTRS Algorithm

In this section, a heuristic algorithm is derived from executing the arrived SR at the device level by the device manager strategy at each layer with complete SR demands, such as the SR length and the compute capacity, as well as the SR deadline, etc. The system resolves the local computation issue with less computation complexity. The fundamental objective is to allocate the SR to the ES, which pursues a smaller processing factor (PF) value θ_{zj}^{w} and it is estimated by Eq. (16). The SR allocation is processed as per the sequence of the feasible ES set by keeping the ESs in ascending order based on the θ_{zj}^{w} value. Additionally, it should satisfy the condition in Eq. (17)

$$\theta_{zj}^{w} = \sum_{i=1}^{N} \sum_{z=1}^{Z} \sum_{j=1}^{J} \left(k_{iz}^{j} + \xi_{iz}^{j}\right) \tag{16}$$

$$\sum_{i=1}^{N} b_i \leq S_j \tag{17}$$

where S_j is the total storage capacity of the ES. Otherwise, the sub-SR will be offloaded to a nearby ES, which satisfies all listed conditions, as can be observed in algorithm 1. Lines 1–13 estimate the proposed scheme to assign the SR to the ES node. Lines 5–11 estimate the feasible node, which has a low θ_{zj}^{w} PF value and is estimated with Eq. (16). The selected targeted node has become responsible for executing the SR to meet the deadline. In IoT devices, energy optimization plays a vital role. Therefore, the energy consumption cost ξ_{iz}^{j} of each device is estimated as Eq. (18), where A_z^i is the power usage of a device.

$$\Xi_{iz}^{j} = \sum_{i=1}^{N} \sum_{z=1}^{Z} \sum_{j=1}^{J} \left(\frac{A_z^i \times b_{iz}}{com_{zj}}\right) \times a_{iz}^{j} \tag{18}$$

Selecting a feasible ES is a hectic task, but we measure the four types of attributes through graph theory, as mentioned in algorithm 2. The second algorithm chooses the shortest path among all the hooked nodes in the framework. Through undirected graph theory, a suitable ES is selected based on a weighted matrix mechanism. For each ES, the bar value is estimated to analyse its location at a comfort level and the probability of selecting a nearby ES. The adulation factor is used to analyse the previous track record of the ES and its capability through the matrix mechanism, which can be observed in algorithm 2. Most existing methods consider distance among nodes as a parameter for accomplishing the task, but we considered both the range and track record of the node using a set of measurements, as well as the behavioural association rate and adulation factor. This process iterates in every step to choose the centralized ES over the IoT environment.

Algorithm 1: Heuristic Reinforcement-RFTRS Algorithm

Input: 1. SR set: $N = \{1, \ldots, N\}$
2. Edge servers set $J[J]$

Output: Adaptive SR allocation

(Continued)

Algorithm 1: Continued

While $i \neq 0$ ***do***
  Let initialize $a_{iz}^{j} = 0$;
  for each $t \in N$***do***
    Let estimate b_{iz}, com_{zj}, ∂_{iz}, $S_j|z$,f_z, $\Psi_i^{z|j}$, $\xi_i^{z|j}$;
    for each $j \in J$***do***
      Estimate PF value by Eq. (17);
      Estimating feasible ESs set by using algorithm 2;
      Update $J[J]$ in ascending order;
      If $k_{iz} \leq \theta_{zj}^{w}$ ***then***
        $j^{th}node \leftarrow t[i]$;
        Update $J[J]$ set;
        Update $a_{iz}^{j} = 1$; *break*;
      end
      else
        Continue until SR allocation;
      end
    end
  end
end
Return adaptive SR allocation $a_{iz}^{j} = 1$;

Algorithm 2: Heuristic ESs Algorithm

Input: 1. Edge servers set $J[J$ with classification based on computing capacity
2. It is all resource-intensive sets
Output: Adaptive feasible ESs set

While $j \neq 0$ ***do***
  Let initialize $S_j^i \neq 0$;
  Let estimate com_j, ∂_j,f_j, ξ^j;
  for each $j \in J$***do***
    Estimate behavioural association rate (bas) value by Eq. (2)
    Update $J[J]$ set in descending order;
    Estimate Eq. (3)
    Update $J[J]$ set;
    Estimate adulation factor by Eqs. (4) and (5) to finalize the priority of ES;
  end
end
Return adaptive feasible ESs set;

6 Simulation Setup for Performance Analysis

This section estimates the RFTRS performance for efficient SR allocation with the MATLAB simulation system. We set the computing capacity of each device, and it is updated in Table 2. We used 100 m × 1500 m areas with $J = 5$ ESs, which are deployed randomly. $Z = 25$ EU are also deployed to accomplish the underlying objective with respect to time. Iteratively, all parameters must be estimated

to act as per the online demand. The SR length is assigned between 15–20 Mb. The entail computing rate of each ES is between 250–750 CC/S, with a storage capacity range of 150–250 Gb, and 15 MHz bandwidth. Initial device communication power is 2.5 W, and its battery capacity $\xi = 250\ J$, and the remaining notation definitions can be observed in Table 3. Google clusters are employed to generate the services in our simulation, where each task contains thousands of tasks with specified resource attributes, and as per the resource demand, the services are offloaded to the potential servers. The computation-to-communication ratio is scaled as (0.1,10). Assume that the state-of-the-art approaches generated 50 to 500 tasks; then, 10,000 test cases are fed to assess the performance and makespan of our system.

Table 2: Device-level computing capacity

Device level	Computing capacity (CPU cycles/s)
f_z	5×116
f_j	10×116
f_c	150×116

Table 3: Simulation attributes

Parameter name	Value
No of ESs	5
No of EUs	25
ESs compute capacity	$250 - 750\ CCs$
Storage capacity	150–250 Gb
SR length	15–20 Mb

We considered five ESs deployed around a 1500 m $\times$ 1500 m area in our simulation and the ESs are deployed around a 20 m $\times$ 20 m range. The arrival rate at each ESs is $25 - 50\ SR/s$ with a moderate bandwidth of 0.65 Mb/s. According to the structure, we fixed the traffic range as 0.56 Mbits. The computing capacity may change as per the demand to 5 $kCCS$. The simulation uses a fixed ES computing threshold value of approximately $0.99/node$ to achieve moderate performance. The channel frequency is approximately 2 $kMHz$, the antenna loss is approximately 4 dB, the gain rate is 1 dB, and the target node sensitivity is approximately $-95\ dBm$.

Fig. 2 shows the workload computing variances between ESs, and our approach has a low-risk rate compared to the rest of the approaches ($\alpha - distribution$, PORA, and TS-TA). To optimize the workload, the ESs' compute capacities must be scaled as per the demand of arrived services to achieve the high performance of ESs. In this scenario, the existing approaches have not conceded the latency rate for backpropagation, but the energy optimization scheme towards respective resource usage rates.

Fig. 3a illustrates the delay analysis report during the EUs' upload of a part of the sub-SR to the ES along with the comparative analysis of the ES workload. We have not conceded the compute cost when it is offloaded to the cloud or a nearby ES during local computing by an ES. Here, we can observe the delay when the ES $J = 3$, the delay value is far better than for the remaining ESs and their local computing count. However, the delay apparently rises when the EU count increases along with the

local computing scenario. The reason for this is that while performing local computing, the ES assets are purely utilized, even when considering a greater number of EUs as well as the SR arrival rate.

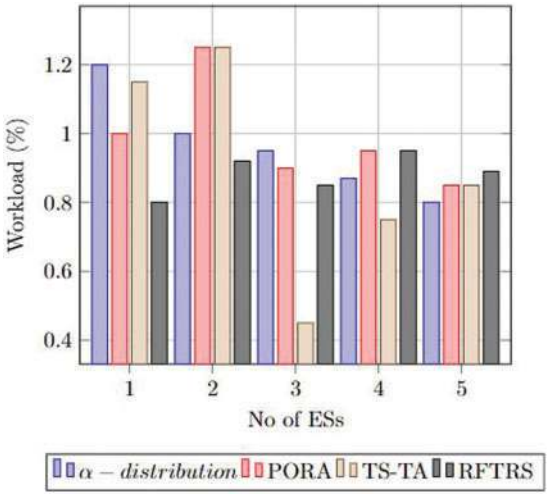

Figure 2: ESs' computing loads

Fig. 3b, shows the delay analysis comparison with respect to the EUs. Assuming the ES count is equal to the EU one, then there is a marked difference in performance. When the work arrival rate raises, there is also an increase in the delay rate. Both are dependent on each other. This is because each EU has accommodated a greater number of sub-SRs, equivalent to its computing capacity. Additionally, the EU is restricted by limited assets (compute, storage) and a minimum number of ESs. In many cases, services beyond the limited capacity of received data are to be offloaded towards the cloud for the computation process. As per the outcomes rate, the delay grew due to the communication one while offloading the sub-SR towards the cloud.

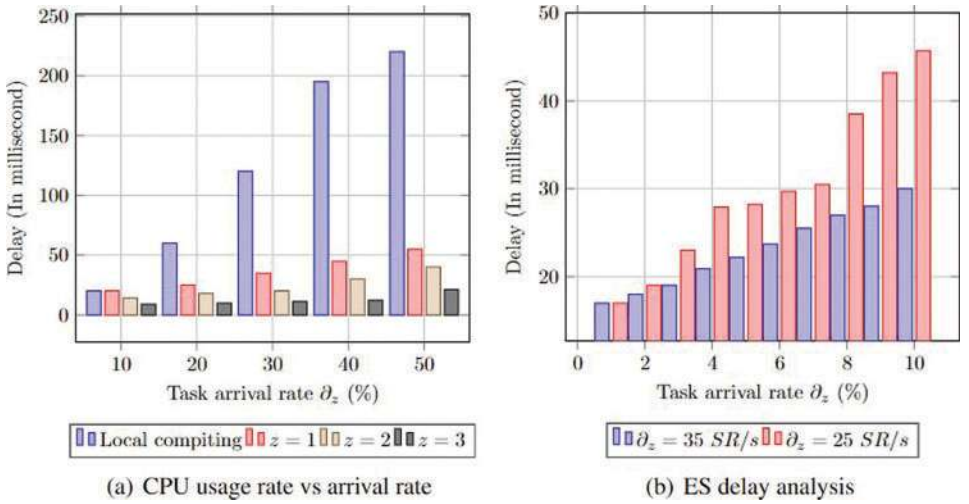

Figure 3: Delay analysis of each ES $\boldsymbol{J = 4}$ with 0.5 Mb/s

Fig. 4a illustrates the offloading influence rate between the EU and ES. Local computing in ES might generally have a low offloading demand since it is packed with all the entail assets. Therefore, it

may not have an unfavourable influence because of the delay in the ES-cloud framework. Generally, the offloading rate is low between the EU and ES, even though the ES has a notable enrichment delay. Our errand allocation package optimally determines the available ES where it is supposed to execute to meet the service deadline. Essentially, the developed policy ensures better computational assets compared with its local computing of ES during cloud and ES collaboration. The ES service rate, communication rate among the cloud, and the ES influence the total delay. Consequently, a further rise in the offloading rate in EUs and ES has not caused the delay.

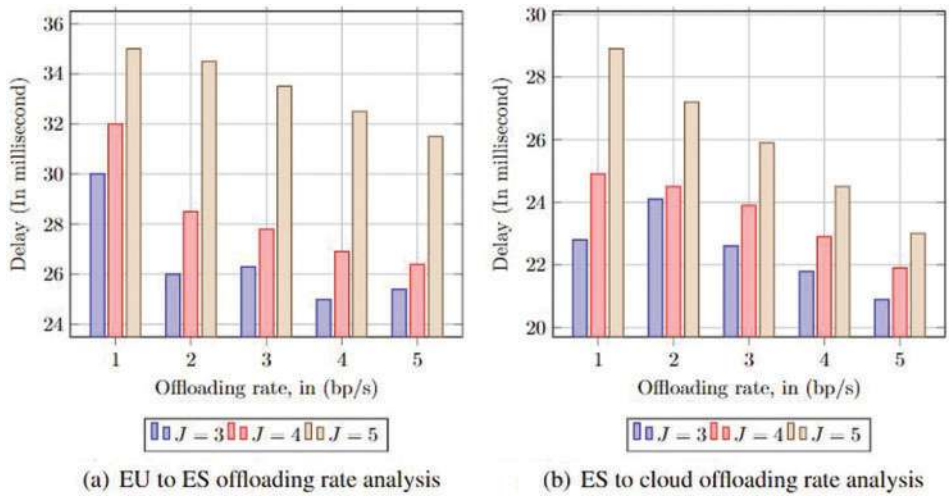

Figure 4: Delay analysis with task arrival rate $\partial_z = \mathbf{50}/\boldsymbol{s}$ (%) and bandwidth 0.5 Mb/s

Fig. 4b describes the offloading rate influence between the ES and cloud. Subsequently, the ES-cloud offloading rate has not affected the local computing of the ES. We can observe in Fig. 4b that as the offloading rate from ES-cloud is raised, high SR data can offload using cloud computation assets, and the outcome is reduced delay. If the ES count is enhanced, then, apparently, the EU's connection rate towards the ESs also increases. Subsequently, the offloading could be balanced among ESs; in terms of the ES to cloud, the offloading time also increases, thereby improving the total delay. Table 4 shows the makespan and probability of the cumulative distribution function (PCDF) outcomes regarding proposed and state-of-the-art approaches. Our system has received an accurate makespan 4.3×10^3, 0.92% rate of PCDF, and 0.62 delay ratio. TS-TA has received the second best accurate makespan 6.8×10^3, 0.81% rate of PCDF, and 2.91 delay ratio.

Table 4: Performance comparison analysis with state-of-the-art approaches

Method	Makespan $\times 10^3$	PCDF [0,1]	Delay
$\propto -distribution$	12.1	0.68	4.2
PORA	9.6	0.76	3.68
$TS-TA$	6.8	0.81	2.91
RFTRS	**4.3**	**0.92**	**0.62**

The task queue length plays an important role in achieving high system performance. In this regard, service arrival and the probability of resource scarcity are essential parameters impacting

system latency. Fig. 5 shows the offloading rate and delay comparative analysis study based on the task arrival rate concerning the probability of the resource scarcity of devices. Consequently, Fig. 5a shows the performance of the proposed system according to the average service execution delay analysis when the resource scarcity probability is $\Delta = 0.75$. The state-of-the-art approach has achieved an abnormal execution rate than the proposed system, but the PORA approach accomplishes a notably lower delay than TS-TA and $\propto$-distribution. Eventually, our approach produced a lower delay rate than other approaches, even at moderate resource scarcity probability, which is approximately $\Delta = 0.5$, as can be observed in Fig. 5b.

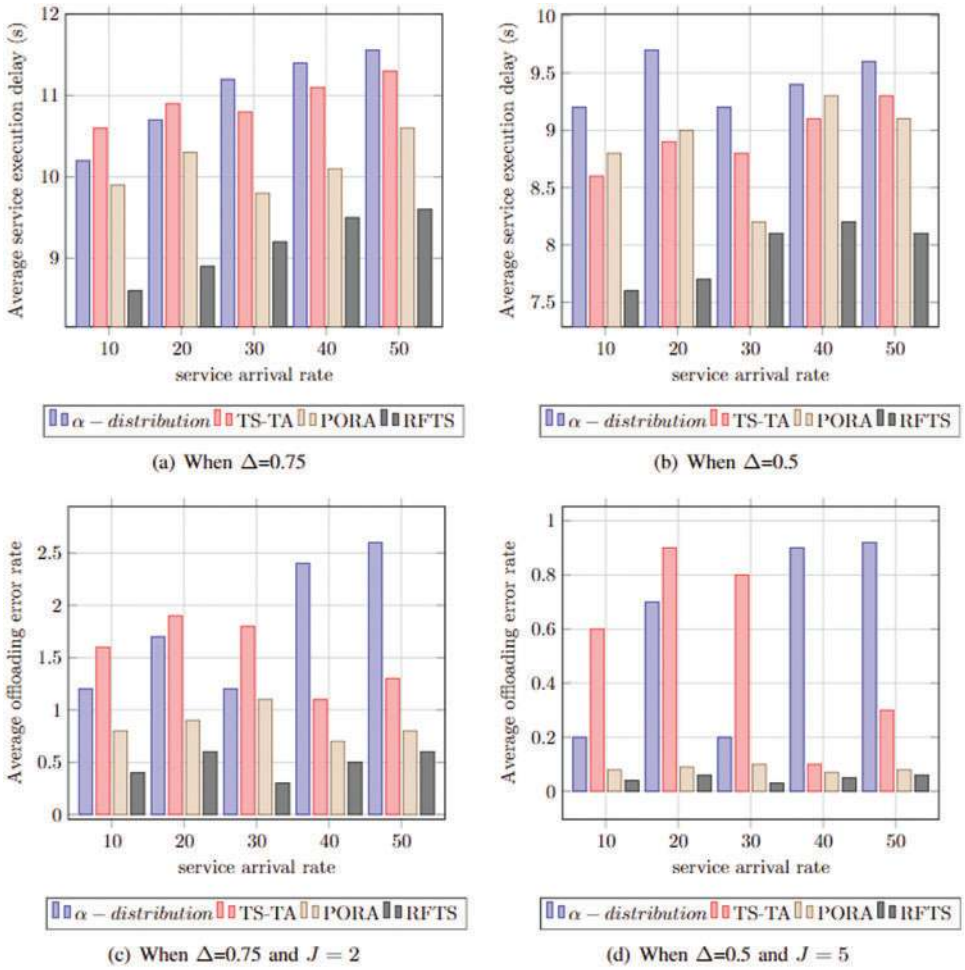

Figure 5: Offloading rate and delay analysis based on task arrival rate concerning the probability of the resource scarcity of devices

Subsequently, the service offloading rate is analysed when the resource scarcity probability is $\Delta = 0.75$ and the server size is $j = 2$ where the proposed system achieved a low offloading error rate because of the reinforcement-based errand allocation method and probabilistic resource requirement analysis model analysis using a set of measurements, as well as the behavioural association rate and adulation factor. During simulation, it is observed that the offloading error rate is reduced, and the success rate of service completion is increased because of the pre-estimation of node-centric measurements. It is

recommended to refer to Table 5 and the optimized offloading error rate can be observed in Fig. 5c. When the service count is increased with moderate resource scarcity probability $\Delta = 0.5$, the proposed performance is drastically increased, which can be observed in Fig. 5d.

Offloading Strategy Functionality Analysis

Let us assume five servers are deployed on an IoT framework with different computation resources. The service arrival rate is also high when the server becomes overloaded. In such a scenario, the server offloads computation-intensive tasks to the suitable sever by an assessment of node-centric attributes, as listed in Table 5.

Table 5: Offloading strategy functionality analysis

J	Adjacent server/device-centric attributes				Decision
	Completion time (ms)	Waiting time (ms)	Residual storage capacity (MB)	Frequency (GHz)	
1	2.8	510.2	160	5	Offload
2	3.2	410.6	450	7.5	Not suitable
3	4.1	395.2	200	4.9	Not suitable
4	1.2	230.6	400	3.5	Offloadee
5	5.6	671.6	360	8	Not suitable

The Table 5 enables two colours of tuples, where the red-coloured one offloads the computation-intensive tasks to the suitable device, which is in green, based on a set of measurements, in addition to the behavioural association rate and adulation factor. In brief, these include the targeted node task frequency, waiting time concerning the task length, task completion time and residual memory to accommodate the offloaded services. The green tuple of server $j = 5$ *i*s selected because it is potentially rich with computation resources. Using this offloading strategy, the server performance has increased more than the state-of-the-art approaches, which can be observed in the below section.

7 Conclusion

In this paper, we examined the performance of the reinforcement learning-based flexible task and resource scheduling (RFTRS) approach to reduce the average latency and errand completion time, stifled by potential battery capacity and the sub-task deadline. The designed adaptive online status predictive model accurately prognosticates the asset requirement of upcoming tasks to make float decisions. In our proposed offloading strategy, a flat collaboration is conceded with nearby ESs and an uplink collaboration with the cloud server for offloading. Furthermore, the RFTRS system considered the communication and computation delay by estimating queuing and local computing time before the service offloading decision using a set of node-centric measurements, as well as the behavioural association rate and adulation factor. The proposed system achieved an average of 0.89% of the service offloading rate, with 39 ms of delay over complex scenarios (using three servers with a 50% service arrival rate). The simulation outcomes confirmed that our system stipulated the uncertainty by explicitly analysing the errand execution with various parameters.

In future, we would like to implement an adaptive mix-integer linear programming (MILP) optimization strategy to increase the system performance to meet the delay-sensitive applications. Subsequently, balancing the performance and convergence time represents a global challenge for accelerating offloading decisions.

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: Zulqar and Kim's research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1A6A1A03039493) and in part by the NRF grant funded by the Korea government (MSIT) (NRF-2022R1A2C1004401). Mekala's research was supported in part by the Basic Science Research Program of the Ministry of Education (NRF-2018R1A2B6005105) and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (no. 2019R1A5A8080290).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Singh, P. Singh and S. S. Gill, "Fog computing: A taxonomy, systematic review, current trends and research challenges," *Journal of Parallel and Distributed Computing*, vol. 157, no. c, pp. 56–85, 2021.

[2] A. A. A. Sen and M. Yamin, "Advantages of using fog in IoT applications," *International Journal of Information Technology*, vol. 13, no. 3, pp. 829–837, 2021.

[3] O. K. Shahryari, H. Pedram, V. Khajehvand and M. D. TakhtFooladi, "Energy and task completion time trade-off for task offloading in fog-enabled IoT networks," *Pervasive and Mobile Computing*, vol. 74, pp. 101395, 2021.

[4] U. M. Malik, M. A. Javed, S. Zeadally and S. u. Islam, "Energy-efficient fog computing for 6G-enabled massive IoT: Recent trends and future opportunities," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14572–14594, 2022. https://doi.org/10.1109/JIOT.2021.3068056.

[5] A. Islam, A. Debnath, M. Ghose and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *Journal of Systems Architecture*, vol. 118, pp. 102225, 2021.

[6] M. S. Mekala and P. Viswanathan, "A survey: Energy-efficient sensor and VM selection approaches in green computing for X-IoT applications," *International Journal of Computers and Applications*, vol. 42, no. 3, pp. 290–305, 2020.

[7] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao *et al.,* "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4192–4203, 2019.

[8] K. Gasmi, S. Dilek, S. Tosun and S. Ozdemir, "A survey on computation offloading and service placement in fog computing-based IoT," *The Journal of Supercomputing*, vol. 78, no. 2, pp. 1983–2014, 2022.

[9] X. Xu, H. Li, W. Xu, Z. Liu, L. Yao *et al.,* "Artificial intelligence for edge service optimization in Internet of Vehicles: A survey," *Tsinghua Science and Technology*, vol. 27, no. 2, pp. 270–287, 2021.

[10] X. Xu, C. He, Z. Xu, L. Qi, S. Wan *et al.,* "Joint optimization of offloading utility and privacy for edge computing enabled IoT," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2622–2629, 2019.

[11] M. S. Mekala and P. Viswanathan, "Equilibrium transmission bi-level energy efficient node selection approach for internet of things," *Wireless Personal Communications*, vol. 108, no. 3, pp. 1635–1663, 2019.

[12] G. Manogaran, G. Srivastava, B. A. Muthu, S. Baskar, P. M. Shakeel *et al.,* "A response-aware traffic offloading scheme using regression machine learning for user-centric large-scale Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3360–3368, 2020.

[13] L. Liu, Z. Chang, X. Guo, S. Mao and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2017.
[14] K. Wang, "Energy-efficient resource allocation optimization algorithm in industrial IoTs scenarios based on energy harvesting," *Sustainable Energy Technologies and Assessments*, vol. 45, pp. 101201, 2021.
[15] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo *et al.,* "MEETS: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4076–4087, 2018.
[16] A. Paranjothi, M. S. Khan, R. Patan, R. M. Parizi and M. Atiquzzaman, "VANETomo: A congestion identification and control scheme in connected vehicles using network tomography," *Computer Communications*, vol. 151, pp. 275–289, 2020.
[17] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo *et al.,* "DEBTS: Delay energy balanced task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2094–2106, 2018.
[18] J. Xue and Y. An, "Joint Task offloading and resource allocation for multi-task multi-server NOMA-MEC networks," *IEEE Access*, vol. 9, pp. 16152–16163, 2021.
[19] E. Haber, T. M. Nguyen and C. Assi, "Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3407–3421, 2019.
[20] H. Zheng, K. Xiong, P. Fan, L. Zhou and Z. Zhong, "SWIPT-aware fog information processing: Local computing *vs.* fog offloading," *Sensors*, vol. 18, no. 10, pp. 3291, 2018.
[21] H. Huang, Q. Ye and Y. Zhou, "Deadline-aware task offloading with partially-observable deep reinforcement learning for multi-access edge computing," *IEEE Transactions on Network Science and Engineering*, pp. 1, 2021.
[22] M. S. Mekala and P. Viswanathan, "Energy-efficient virtual machine selection based on resource ranking and utilization factor approach in cloud computing for IoT," *Computers, & Electrical Engineering*, vol. 73, no. 3, pp. 227–244, 2019.
[23] Y. Huang, H. Xu, H. Gao, X. Ma and W. Hussain, "SSUR: An approach to optimizing virtual machine allocation strategy based on user requirements for cloud data center," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 670–681, 2021.
[24] J. Liu, A. Zhou, C. Liu, T. Zhang, L. Qi *et al.,* "Reliability-enhanced task offloading in mobile edge computing environments," *IEEE Internet of Things Journal*, pp. 1, 2021.
[25] T. Bahreini, H. Badri and D. Grosu, "Mechanisms for resource allocation and pricing in mobile edge computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 667–682, 2021.
[26] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.
[27] T. G. Rodrigues, K. Suto, H. Nishiyama and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–881, 2016.
[28] J. Yao and N. Ansari, "QoS-aware fog resource provisioning and mobile device power control in IoT networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 167–175, 2018.
[29] X. Gao, X. Huang, S. Bian, Z. Shao, Y. Yang *et al.,* "PORA: Predictive offloading and resource allocation in dynamic fog computing system," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 72–87, 2019.
[30] T. S. Kumar, S. D. S. Mustapha, P. Gupta and R. P. Tripathi, "Hybrid approach for resource allocation in cloud infrastructure using random forest and genetic algorithm," *Scientific Programming*, 2021.
[31] A. Lakhan, M. A. Mohammed, A. N. Rashid, S. Kadry, K. H. Abdulkareem *et al.,* "Restricted Boltzmann machine assisted secure serverless edge system for Internet of Medical Things," *IEEE Journal of Biomedical and Health Informatics*, pp. 1, 2022.
[32] A. Lakhan, M. A. Mohammed, S. Kadry, S. A. AlQahtani, M. S. Maashi *et al.,* "Federated learning-aware multi-objective modeling and blockchain-enabled system for IoT applications," *Computers and Electrical Engineering*, vol. 100, pp. 107839, 2022.

[33] A. Lakhan, M. A. Mohammed, M. Elhoseny, M. D. Alshehri and K. H. Abdulkareem, "Blockchain multi-objective optimization approach-enabled secure and cost-efficient scheduling for the Internet of Medical Things (IoMT) in fog-cloud system," *Soft Computing*, vol. 26, no. 13, pp. 1–14, 2022.

[34] A. Lakhan, M. A. Mohammed, J. Nedoma, R. Martinek, P. Tiwari *et al.,* "Federated-learning based privacy preservation and fraud-enabled blockchain IoMT system for healthcare," *IEEE Journal of Biomedical and Health Informatics*, pp. 1, 2022.

[35] A. Lakhan, M. A. Mohammed, O. I. Obaid, C. Chakraborty, K. H. Abdulkareem *et al.,* "Efficient deep-reinforcement learning aware resource allocation in SDN-enabled fog paradigm," *Automated Software Engineering*, vol. 29, no. 1, pp. 1–25, 2022.

[36] J. Yao and N. Ansari, "QoS-aware fog resource provisioning and mobile device power control in IoT networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 167–175, 2018.