TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 4167 – 4187
© TÜBİTAK
doi:10.3906/elk-1902-152

Energy saving scheduling in a fog-based IoT application by Bayesian task classification approach

Gholamreza HEYDARI, Dadmehr RAHBARI, Mohsen NICKRAY*
Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran

Received: 22.02.2019 • **Accepted/Published Online:** 11.09.2019 • **Final Version:** 26.11.2019

Abstract: The Internet of things increases information volume in computer networks and the concept of fog will help us to control this volume more efficiently. Scheduling resources in such an environment would be an NP-Hard problem. This article has studied the concept of scheduling in fog with Bayesian classification which could be applied to gain the task requirements like the processing ones. After classification, virtual machines will be created in accordance with the predicted requirements. The ifogsim simulator has been applied to study our fog-based Bayesian classification scheduling (FBCS) method performance in an EEG tractor application. Algorithms have been evaluated on a practical application of brain signal tracking system. According to the results, the FBCS method, compared with other methods, has reduced the energy consumption in the cloud and the executing task cost in cloud; and also the average of energy consuming in mobiles has been decreased by smart decision making.

Key words: Fog computing, tasks scheduling, machine learning, Bayesian classification

1. Introduction

Based on the comprehensive studies of scheduling algorithms in cloud [1] and fog environments [2, 3] and by the presence of Internet of things (IoT), the future generation networks like mobile nets will have a compact, heterogeneous, and very unstable conditions. Therefore, the current static scheduling techniques would not be proper for the future generations of the networks. Using techniques such as artificial intelligence and machine learning (ML) has attracted the attention of network planning scientists [4]. With the development of devices at the edge of the network, the concept of the fog emerged and provided a low latency, large scale, and geographic distribution, mobility, location awareness, flexibility, heterogeneity, and scalability [5]. According to [6], fog is a local cloud which provides a limited and trustable processing power with a slight delay near the users. The difference between fog and cloud will be defined in the closeness of the final users, geographical distribution, and the support of mobility [7]. Fog computing (FC) was introduced by Cisco Systems to develop the model of cloud computing (CC) to the edge of the network especially for the services of IoT [8]. In [9], some scenarios of the fog environment applications have been analyzed and some scenarios of other documents have also been reviewed and finally, it can be concluded that only these general scenarios of FC can be used:

- Data will be gathered from the edge of the network such as vehicles and sensors.
- A large number of machines in the network send data.
- Processing data and decision making must be done in less than a second.

*Correspondence: m.nickray@qom.ac.ir

Based on [10], CC fails to guarantee the Quality-of-Experience (QoE) requirements for some services like the smart grids (SG) services such as latency, bandwidth, energy consumption, and network cost. FC extends CC into the edge of the network by deploying limited resource, localized computing, and processing facilities. FC is mostly in contact with the service providers and the owners of equipment in fog layer like the end user devices, access points, and edge routers [11]. Regarding the high dynamicity of fog network, fair and optimized distribution of resources among the tasks requires an efficient and optimized scheduling algorithm. Considering the positive consequences of using the ML algorithms in CC [12], we have tried to study the results of the utilization of Bayesian classification algorithm for scheduling in FC. Our key contributions in this paper are as follows.

1. Since a large number of sensors and small things in IoT require low energy consumption; thus, we consider a three-tier architecture so that end devices are at the lowest level, the fog at the intermediate, and the cloud at the highest.

2. The use of the Bayesian method to classify tasks based on their processing requirements has led to the creation of intelligent virtual machines, and the resources of the devices are optimally distributed between tasks.

3. According to the application, cost and energy metrics are analyzed based on the number of departments, mobile devices, users and the EEG signal. The simulation results have shown that using FBCS method has reduced energy consuming and cost in the cloud than random, first-come-first-served (FCFS), delay-priority [13], and energy consumption of IoT application in fog computing (ECIF) [14] methods.

The rest of this paper is organized as follows. In Section 2, related works and results of using the ML methods for scheduling in the fog and cloud environments have been described. In Section 3, the model of the system is presented. In Section 4, the FBCS method including the Bayesian classification and the algorithms of task scheduling in fog environment have been described. In Section 5, the results of simulation have been evaluated and are compared with other methods in detail, and finally in Section 6, the overall results, and future works have been brought.

2. Related works

Scheduling in cloud computing is a new viewpoint for distributed computing and parallel processing. It prepares the computing under the name of a beneficial system for a pay-per-use service [15]. Practical programs can be modeled as the workflows in a directed acyclic graph (DAG). CC also represents infrastructure, platform, and software as a service [16]. When the techniques and the devices of IoT entered the life of ordinary people more and more, the current CC pattern could hardly support the need for mobility, location awareness, and low latency. Therefore, FC was introduced to solve the mentioned problems [7]. In response to the question of what the benefits of using fog in combination with the cloud are, an answer with an analysis of two scenarios including only cloud and a combination of cloud-fog together with a comparison between processing delay and consumed energy has been prepared in [6] according to the increase of the number of users and different workloads. Their simulation has been done by discrete event system specification. The results approve that by using the fog networks, waiting time is less and the data rate will be more.

In many domain-specific applications like industrial applications, the CC will not be able to answer the need of users on time [11]. Datacenters will classify the tasks according to service level agreement (SLA) and the requested services [15]. To prepare a better service with optimized use of resources, the tasks will be loaded on vector machines (VMs) and the resources will be shared. This is exactly the opposite point of the idea of occupying the whole resources [11, 17].

Scheduling can be implemented in three different levels: task level, resource level, and workflow level. Stability and efficiency of systems depend on a variety of factors such as task scheduling.

Scheduling in CC is mostly focusing on improving the use of resources and decreasing the required time to complete a job. It also has this ability to be generalized to FC. In network, the cost to do a specific job depends on the time and exchanged data. One way to reduce the cost of the user is to decrease the volume of sent data to the cloud which is the main idea behind creating the fog. This fact reveals the importance of paying attention to FC and scheduling in this environment. Cloud can connect to an IoT device through fog nodes [18].

Based on [13], in the delay-priority method, tasks are scheduled based on lower delay. Based on this algorithm, the remaining amount of running tasks or runtime queues is obtained and these values are arranged in an ascending list. Therefore, requests with the lowest remaining execution are prioritized to run faster. Since the method of this paper is based on Bayesian classification. This method is the scope of ML. ML is a subset of artificial intelligence. In the following, we survey the researches about scheduling by ML methods.

2.1. ML-based scheduling

For the scenarios in which the learning data will be produced slowly and the states are countable, we can use some type of supervised learning such as classification (discrete variables) or regression (continuous variables) [19]. According to [20] the impacts of using the ML concepts such as the mechanism of automatic resource allocation, scheduling, and the smart management of resources on cloud environments have been studied. For this purpose, some learning methods like SmartSLA which is a cost-sensitive resource management system has been studied. The results show that SmartSLA will be able to calculate the predictive models for hardware and software resources successfully. As in [21], a task scheduling scheme has been designed based on an reinforcement learning (RL) to lessen the makespan and the average waiting time under the limitation of VM resources and the deadline.

A parallel multiagent technology has been used to make the balance between discoveries and occupy in the learning process. The convergence of the Q-learning algorithm has also been speeded up and at the end of each section, a semioptimized policy has been reached. As in [16], in matters of resource scheduling, due to the concentrating on a special purpose like minimizing the executing time or workload and the lack of using the CC features, classification and regression tree and modified bacterial foraging optimization algorithms have been proposed. In [22], the authors recommended a multiagent resource selection technique based on a neural network which would be able to imitate the services of an expert user in the distributed systems like grids and clouds.

According to [19], in comparison with the ad hoc heuristic, the approaches of ML can be helpful through intelligent resource allocation, selection of action according to the conceptual states and environmental factors for scheduling. These approaches can represent a solution based on ML by modeling supervised learning and prepare the architecture.

2.2. ML-based scheduling in cloud

Due to the density of VMs and jobs in the CC environment and this point that the matter of job scheduling is a NP-Hard complete problem a multiagent parallel learning has been used to increase the speed of job optimization scheduler scheme in some books [21]. Neural networks are compatible with the datacenter management and decrease the cost of cooling process of Google datacenters up to 40%. Deep learning (DL) at the edge of network decrease the amount of back propagated data to datacenters [19]. Efficiency of energy, spectrum, delay, stability, and safety are the key parameters which are considered at the stage of networking operation. Optimization of these parameters needs the real-time learning and decision-making algorithms [4]. Feature engineering is the first step of every analysis based on ML. This is the process of selecting the correct data metrics to represent them as input to the ML algorithms [18]. In [19], the facility of resource management has been discussed by the ML in large scale distributed systems. They have studied the way ML automatically detect and understand the workloads and the environments.

2.3. ML-based scheduling in fog

Data mining and ML have been used in some networks such as wireless sensor network and in many recent studies. In [23], authors provide predictive models in the networks where the instability and dynamism are the essential features. The Kernel linear regression and the extended Kalman filter are widely used to predict the sensor values. According to [24], energy-efficient scheduling is proposed based on the deep Q-Learning model and dynamic voltage and frequency scaling for periodic tasks in real-time systems. Usually, much of the power in smart devices is used by CPU and GPU, and a lot of methods have been proposed to reduce the energy consumption in recent years.

In [18], the use of low-resource ML has been evaluated on the wearable health-care fog devices which are kept close to users. In a big data analysis to discover patterns in the physiological data, they have developed a prototype of fog-based traditional unsupervised ML. The results proved that the proposed architecture for low-powered clinical ML is promising, and a computer with limited data mining capabilities can analyze the data collected from various wearable sensors for a remote health-care application.

According to [8], the researchers have proposed an RL based on the offload code mechanism to ensure low-latency services for mobile service users. Based on [10, 14, 25], the offloading-based scheduling algorithm designed for it sets a threshold value so that if there are not enough resources to execute the request, the requests are directed upward through the middle layer devices to the cloud. Based on [26], ECIF, an offloading method, could lead to the integration of IoT and CC applications. It could connect IoT nodes, sensors, edge devices, or fog nodes. Factors such as energy, latency, load balancing, and computational requirements of an application could affect offloading. They present recent offloading schemes proposed for domains such as FC, CC, and IoT.

In [8], the authors used the distributed RL algorithm to drain the base blocks in a decentralized state to deploy mobile codes on the mobile Fogs which are geographically distributed. Their proposed method reduced the mobile access runtime, latency, and also the energy consumption of mobile devices.

According to [23], a distributed learning model on the sensor device and simulation of data flow in the fog has been proposed and the benefits of FC in this model have been investigated. The framework has shown that the combination of Fog and CC is beneficial for IoT applications.

In [5], a data predictor was introduced in a fog-based model and was built on the lightweight Message Queuing Telemetry Transport protocol. They have analyzed four ML algorithms to predict the measurement of the real sensor. Selective methods include multiple linear regression, regression tree, bagged decision tree, and artificial neural network. The results of ML algorithms based on energy and data exchange factors are the same, except for the linear regression having less accuracy.

In [27], Cognitive Radio-based IoT (CIoT) is a promising solution for the IoT applications. The main challenge of CIoT is the efficiency of packet transmissions while using the cognitive network. Therefore, a new

Q-Learning based on the transfer scheduling mechanism is proposed to use the DL method for CIoT.

In [28], the researchers investigated and reviewed many documents about using the ML and intelligence approach in IoT application like remote health-care applications or smart gateways. Based on [29], they designed a dynamic RL scheduling algorithm and a deep dynamic scheduling algorithm to gain a fine-balanced solution to offload the tasks for mobile devices. According to the results, the energy cost and service delay will increase with raising of traffic size and computation workload. With a more detailed statement, the computation workload has more effect on the energy cost than traffic size.

In [30], the authors proposed a manifold learning dynamic spectrum allocation framework combining FC and CC so that the received signal is processed close to where it is generated. Based on the features of the received signal, different ML methods have been applied such as least square logic regression, support vector machine, and manifold learning. The extracted feature could also be projected to a higher feature space to improve the categorization performance. Based on the defined rules, each fog node has the ability to infer and choose the best spectrum candidate to transmit the signal without interfering with the licensed legitimate primary users.

3. System model

As shown in Figure 1, requests in the fog layer are sent by end devices such as sensors, mobile phones, and surveillance cameras to be placed in the queue waiting to run. In one path, requests are executed and their specifications are classified and stored until when proper VMs will be updated and made based on them. On the other path, requests are executed only by referring to premade VMs, and the result will be answered. We run Virtual Reality Game (VRGAME) application [31] in the proposed system.

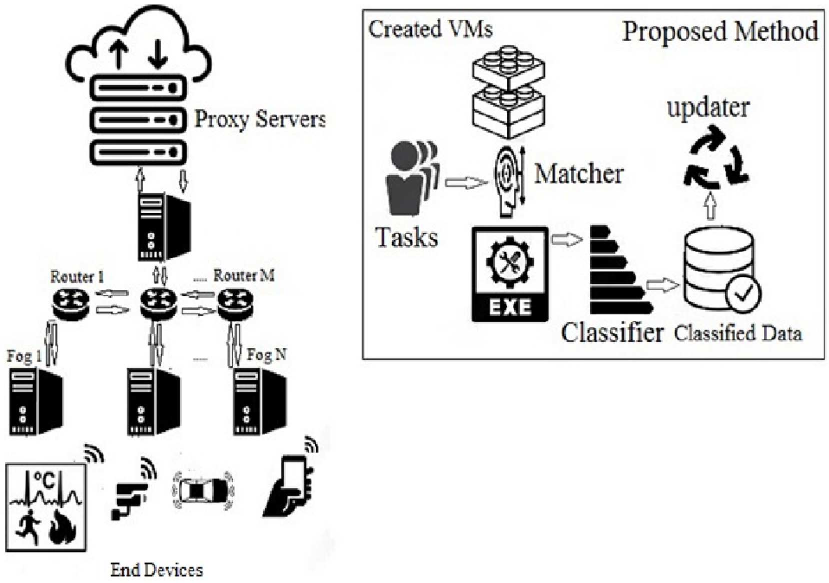

Figure 1. System Model

3.1. VRGAME application model

The VRGAME[1] application is a human-vs-human game. Each player needs to wear a wireless EEG headset which is connected to a smartphone. The application receives the EEG signals sensed by the EEG sensor and calculates the brain state (concentration) of the user and streams raw data to the client module. The

[1]Electroencephalography Tractor Beam Game

client module sends consistent data to the concentration_calculator module, which computes the concentration level of the user and returns it to the client module. Client module updates the game display to the player. The coordinator module gathers and distributes measured concentration among players. The value of the EEG parameter could be used to determine the interval between the two sensed signals. This parameter, in combination with the parameter MAX_SIMULATION_TIME of the Config class, determines the number of generated tuples. When the EEG is set at lower values, more requests are sent per unit time, and the modules that receive the sensed data need more processing resources, and, in the meantime, they also produce more data which makes all devices available in processing environments (such as end devices, fogs, and even datacenters) are challenged. Based on Figure 2, the EEG sensor, display actuator, and client module are placed in the mobile device. The concentration_calculator and the coordinator modules can be placed in the fog devices (such as gateways, routers, proxy servers, and other intermediate devices which are located at the lower level of the cloud) or at a cloud datacenter [13, 31].

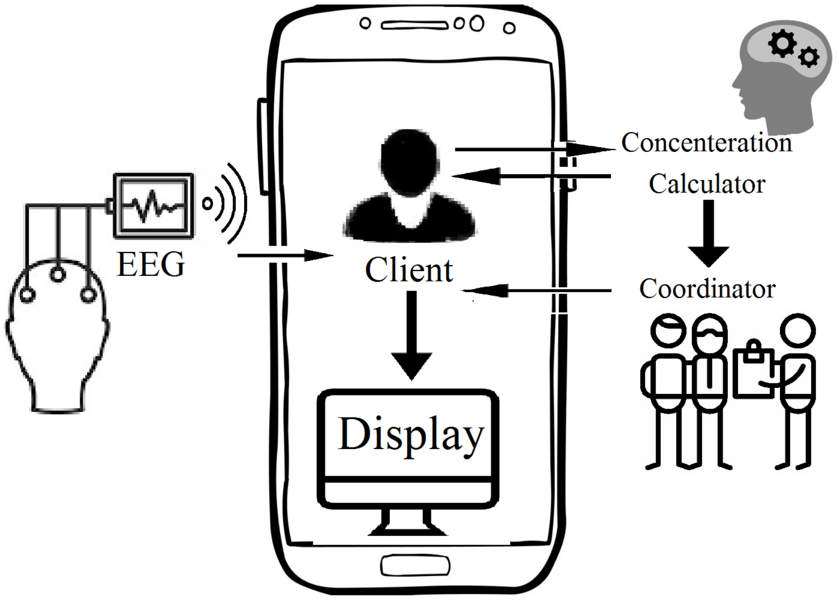

Figure 2. Application model

4. The proposed approach

In this section, due to the positive results of using the Bayesian classifier in [12], as well as by comparing the speed and accuracy of the methods based on Bayesian theory and some of the ML algorithms [32], the utilization of the Bayesian classifier in the fog environment is proposed to schedule tasks in accordance with their requirements, such as processing requirements and destination modules. The FBCS method as Figure 1 is described in the form of two algorithms. In this method, $TS^{2} = \begin{bmatrix} m_1 & p_1 \\ \vdots & \vdots \\ m_n & p_n \end{bmatrix}$ is a two dimensional (2D) array, the first dimension is the tasks destination modules name, and the second dimension is their processing

[2]Tasks Specification

requirements. The 2D array $CT^3 = \begin{bmatrix} m_1 & p_1 \\ \vdots & \vdots \\ m_i & p_j \end{bmatrix}$ is the result of running the Bayesian classifier on TS and the specification is similar to TS. Cloud destination module counters (CDMC) is a one-dimensional array containing counters of different VM in the cloud and mobile destination module counters (MDMC) is a one-dimensional array containing counters of different VMs on mobile devices. *dmn* represents the task destination module name, *rm* is the processing power requested by the task, *nrc* represents the number of tasks running on the VM or module, and *cetc* (current executed tasks counter) is the number of executed tasks from the beginning up to now.

4.1. Bayesian classifier

In step 23 of Algorithm 1, the Bayesian classifier is applied to the 2D array of TS. The arrays $M = TS \begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix}$ and $P = TS \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$ are the first and second columns of the 2D array of TS that has been defined as M and P. The indices j' and i' are derived from Eq. (1) and represent a corresponding between the module name and the amount of the required executed task processing.

$$(i', j') = argmax\left(P(M_i|P_j) = \frac{P(P_j|M_i) \times P(M_i)}{P(P_j)} \right), \tag{1}$$

where $P(M_i|P_j)$ calculates the probability of a VM M_i which has P_j processing requirements. For every value of i, the probability $P(M_i|P_j)$ is calculated for all j values. According to Eq. (1), we seek the best adaptation between the module and the processing requirement of that module. In fact, we plan to allocate the most probable requirement (the processing requirement) to the module. Therefore, physical resources are optimally shared for logical components and the waste of resources would be at the lowest level. $P(M_i)$ and $P(P_j)$ are obtained by Eqs. (2) and (3), respectively.

$$P(M_i) = \frac{|M_i|}{|M|}, \tag{2}$$

$$P(P_j) = \frac{|P_j|}{|P|}, \tag{3}$$

$$CT_{(i',1)} = M_{i'} \ and \ CT_{(i',2)} = P_{j'}, \tag{4}$$

where $P(M_i)$ is the probability of M_i module relative to total modules observed and $P(P_j)$ is also the probability of processing requirement P_j relative to total processing requirements observed. $P(P_j|M_i)$ is the probability of processing P_j when the module is of type M_i and can be calculated for the constant value i and values from 1 to n for j.

[3]Classified Tasks

To prove the correctness of Eq. (1), Table 1 is presented. Numeric values are specified in this table. In Table 2, based on the values in Table 1, all possible probabilities are calculated and the best values of i and j are gained considering Eq. (1). Of course, in the numerical example given in Table 2, the values of the obtained indices are randomly equaled. Ifogsim simulator has a very random and dynamic environment; during the implementation of multiple simulations, it is determined that the CT is sometimes set to duplicate values. Certainly, these duplicate values do not affect the classification, but since resources are allocated on the basis of this 2D array, it can affect the scheduling. For this reason, Eq. (5) is defined to be repeated through a loop with k (k is smaller than n) and possible duplicate values can be eliminated. Finally, the CT is completely refined and each row would exclusively identify some type of VM and its processing requirements. Because all i and j variables can have values from 1 to n, the time complexity is $O(n^2)$ and the time complexity of Eq.(5) will be obviously $O(k^2)$. Since n is much larger than k, the time complexity of the algorithm is $O(n^2)$. This polynomial time complexity allows the algorithm to run on devices with limited resources, like devices in the fog environment.

$$if\ CT_{(k,1)} == CT_{(h,1)}\ \ and\ \ CT_{(k,2)} == CT_{(h,2)}\ \ then\ remove\ CT_h\ from\ CT \tag{5}$$

Table 1. Introducing modules and probabilities.

Row	Module	PR	Number	Row	Module	PR	Number	Row	Module	PR	Number
1	M_1	P_1	1000	1	M_2	P_1	100	1	M_3	P_1	10
2	M_1	P_2	100	2	M_2	P_2	1500	2	M_3	P_2	320
3	M_1	P_3	500	3	M_2	P_3	120	3	M_3	P_3	1600
4	M_1	P_4	50	4	M_2	P_4	0	4	M_3	P_4	410
5	M_1	P_5	30	5	M_2	P_5	450	5	M_3	P_5	120
Total			1680	Total			2170	Total			2460

Table 2. The calculated probabilities.

$P(M_i)$	$P(P_j)$	$P(M_1\|P_j)$	$P(P_j\|M_1)$	$P(M_2\|P_j)$	$P(P_j\|M_2)$	$P(M_3\|P_j)$	$P(P_j\|M_3)$
$P(M_1) = 0.266$	$P(P_1) = 0.175$	$P(M_1\|P_1) = 0.904$	$P(P_1\|M_1) = 0.595$	$P(M_2\|P_1) = 0.090$	$P(P_1\|M2) = 0.046$	$P(M_3\|P_1) = 0.068$	$P(P_1\|M_3) = 0.004$
	$P(P_2) = 0.304$	$P(M_1\|P_2) = 0.052$	$P(P_2\|M_1) = 0.059$	$P(M_2\|P_2) = 0.779$	$P(P_2\|M_2) = 0.691$	$P(M_3\|P_2) = 0.166$	$P(P_2\|M_3) = 0.130$
$P(M_2) = 0.343$	$P(P_3) = 0.351$	$P(M_1\|P_3) = 0.225$	$P(P_3\|M_1) = 0.297$	$P(M_2\|P_3) = 0.053$	$P(P_3\|M_2) = 0.055$	$P(M_3\|P_3) = 0.720$	$P(P_3\|M_3) = 0.650$
	$P(P_4) = 0.072$	$P(M_1\|P_4) = 0.109$	$P(P_4\|M_1) = 0.029$	$P(M_2\|P_4) = 0.000$	$P(P_4\|M_2) = 0.000$	$P(M_3\|P_4) = 0.900$	$P(P_4\|M_3) = 0.166$
$P(M_3) = 0.389$	$P(P_5) = 0.095$	$P(M_1\|P_5) = 0.049$	$P(P_5\|M_1) = 0.017$	$P(M_2\|P_5) = 0.748$	$P(P_5\|M_2) = 0.207$	$P(M_3\|P_5) = 0.199$	$P(P_5\|M_3) = 0.048$
	argmax =	(i = 1 , j = 1)		(i = 2, j = 2)		(i = 3, j = 3)	

4.2. Tasks classification

In Algorithm 1, tasks are classified according to the destination module and their processing requirement. The input of this algorithm includes the application ID, broker ID, number of departments, number of mobiles in each department, logical cloud deployment mode, and the 2D array of TS. The output will be the CT which is obtained after applying the Bayesian classifier on the TS.

4.2.1. Training phase

In this section, Algorithm 1 is explained and the job is to classify the tasks. The input of the algorithm includes variables such as *AppID* representing the application identifier, the *BrokerID* represents the Fogbroker identifier, the *department* identifies the number of departments, *mobilesperdept* represents the number of mobile devices in each department and boolean $CloudDeploymentMode$ variable. If $CloudDeploymentMode$ was true, deployment mode would be cloud-based. The final input is 2D array TS, the first dimension of the array contains the destination modules and the second, processing requirements of the destination modules. After executing this algorithm, the 2D array CT is obtained, the first and second dimensions of the CT, are like the TS. However, the number of entries in the CT is less than the TS.

In the first step, based on the application model defined in Figure 2, the application is created. Then, in steps 4 through 13 based on the value of the $CloudDeploymentMode$, modules are mapped on devices. In step 4 $CloudDeploymentMode$ will be checked and if it is true, the model will be cloud-based. Then in steps 5 and 6, the number of departments multiples the number of mobiles in each department, the coordinator and concentration_calculator modules are deployed in the cloud. In step 8 for all the mobiles, the client module would be installed on them. If $CloudDeploymentMode$ is false, in steps 10 and 11, as the number of departments multiples the number of mobiles in each department only the coordinator module will be deployed on the cloud. In step 14, the simulator starts. From steps 15 to 20, as long as there is a task in the queue, the tasks would be executed and their specifications, such as the destination module and the processing requirement, are included in the TS. When there are no other tasks to be executed, in step 21, the Bayesian classifier runs on TS and its output is stored in CT. In the end, in step 22, the simulator will be stopped and the training phase is completed.

4.2.2. Test phase

The test phase of Algorithm 1 starts in step 25 and $ExecutedTaskCounter$ and $MaxIteration$ variables are initialized. The $ExecutedTaskCounter$ counts the number of tasks that have been executed since the beginning of the test phase and the initial value is zero. The $MaxIteration$ determines the number of tasks which are to be done to update the operation. In the simulation, the $MaxIteration$ has been initialized with 10, which is explained as follows.

- **Max iteration:** The appropriate policy to determine the amount of $MaxIteration$ could be based on QoS. In this paper, the constant value for $MaxIteration$ was considered to reduce the complexity of the simulation. The dynamic method to obtain $MaxIteration$ and determine the best time for updating resource allocation could be a challenging topic to improve system performance. The ifogsim simulator works through the *updateAllocatedMips* function in the Fogdevice class to update the processing resources assigned to VMs or modules. In order to execute the algorithm, it is necessary to call this function at certain intervals and make the necessary update. Since the behavior of this simulator and the call of this function is very dynamic and random, the $MaxIteration$ variable is defined to specify when the simulator engine calls the *updateAllocatedMips* function and ignores the update operation when the time is not appropriate. In this way, the behavior of the algorithm gets out of a random state and is controlled. The most convenient way to find the best time to perform an update operation is to call the function after

```
Input: $AppID$, $BrokerID$, $department$,$mobilesperdept$, $CloudDeploymentMode$, $TS$ [$dmn$, $rm$]
Output: $CT[dmn,\ rm]$
    Training Phase:
    Create Application by $AppID$ .
    Create Fog Broker by $BrokerID$.
    Module Mapping.
    if $CloudDeploymentMode$ then
        for $i < department \times mobilesperdept$ do
            Add ”connector” and ”concentration-calculator” Modules To Cloud.
        end
        Add ”client” Module To All Mobile Devices.
    else
        for $i$=0 to $i < department \times mobilesperdept$ do
            Add ”connector” Module To Cloud.
        end
    end
    Start Simulation.
    $n = 0,\ m = 0$.
    while Task Queue Is Not Empty do
        Execute Task.
        $dmn$ = Destination Module Name Of Executed Task.
        $rm$ = Requested Mips Of Executed Task.
        $TS[n,\ m] = [dmn,\ rm]$.
        $n++,\ m++$.
    end
    $CT$=Bayesian Classifier(TS). // The result of running Bayesian classifier on TS array.
    Stop Simulation.
    Testing Phase:
    $ExecutedTaskCounter = 0,\ MaxIteration = 10$.
    CDMC[$c_0$,...,$c_i$] = [0,...,0], MDMC[$m_0$,...,$m_j$] = [0,...,0].
    while Task Queue Is Not Empty do
        Execute Task.
        $ExecutedTaskCounter++$.
        if $ExecutedTaskCounter$ % $MaxIteration$ == 0 then
            Update Allocation Of Mips.
            CDMC[$c_0$,...,$c_i$] = [0,...,0].
            MDMC[$m_0$,...,$m_j$] = [0,...,0].
        end
        if Executed Task Class == CDMC[...,$c_i$,...] then
            $++c_i$.
        end
        if Executed Task Class == MDMC [...,$m_j$,...] then
            $++m_j$.
        end
    end
```

Algorithm 1: Tasks Classification

executing a certain number of tasks. In order to obtain the most appropriate value for this variable, the algorithm runs for different values of iteration. Table 3 shows the results of the FBCS method for values of 1, 10, 100, and 1000 of $MaxIteration$. By evaluating different parameters, the best value was obtained. The results show that if the value of this variable is 10, the cost and energy consumption of the Cloud will be less. Moreover, the table values indicate that this variable does not have a significant effect on the average energy consumed in mobile devices.

In step 26, according to the implemented classification of step 23, CDMC and MDMC arrays will be defined and all entries are set to zero. These arrays contain counters. The CDMC modules which will run in the cloud and MDMC modules that will run in the mobiles (all devices in the lower layer of the cloud). These will count the number of tasks whose destination modules are matched with the classifications derived from step 23. In steps 27 to 41, as long as there is a task in the task queue and the number of executed tasks is equal to the $MaxIteration$, the processor assigned to the modules are updated and the values of the counters in the two CDMC and MDMC arrays are set to zero. In steps 35 to 40, according to the classification type of executed task, the values of the current counters in CDMC and MDMC arrays are increased. Based on the counter values of CDMC and MDMC, in Algorithm 2, the processing power assigned to the modules will be updated.

Table 3. Max iteration Analysis. (Dep#=1)

Mob#	EEG	User#	Iteration	Cost of execution in cloud	Average energy consumption all mobiles per department	Cloud energy consumption
30	10	1	1	344,491.8000	827,609.6990	13,562,989.75
30	10	1	10	330589.3260	827,615.0795	13,553,183.54
30	10	1	100	333,441.1959	827,620.0728	13,555,195.12
30	10	1	1000	340,751.8014	827,616.0418	13,560,351.71
20	3	1	1	462,768.6891	875,251.6404	13,646,417.20
20	3	1	10	457,810.5450	875,244.0441	13,642,919.93
20	3	1	100	461,964.2201	875,247.9670	13,645,849.76
20	3	1	1000	461,738.0251	875,247.3434	13,645,690.21
120	10	1	1	1,000,244.8000	827,350.3303	14,025,529.81
120	10	1	10	977,039.6004	827,364.7526	14,007,161.86
120	10	1	100	990,737.5004	827,349.8267	14,018,823.77
120	10	1	1000	951,656.6004	827,357.1667	14,008,568.65
20	10	10	1	249,046.1834	827,589.5391	13,495,666.50
20	10	10	10	232,677.8130	827,636.4867	13,484,120.95
20	10	10	100	245,541.5895	827,662.9379	13,493,194.51
20	10	10	1000	267,101.8911	827,613.3241	13,508,402.22

4.3. Resource allocation

Algorithm 2 updates the processor allocation to the modules of the device. The input consists of CT derived from Algorithm 1, $cetc$, $MaxIteration$, and $availablemips$ variables and CDMC and MDMC arrays. The $cetc$ counts executed tasks and the $availablemips$ is equal to the available processing power of the device. Output has made up modules based on the derived classes from Algorithm 1. In Step 1, if the number of executed tasks is equal to the $MaxIteration$, steps 2 to 11 are executed. For the number of the derived classes from Algorithm 1 which their destination device is cloud or fog and conforms to the current counter value in the CDMC and MDMC arrays, if the target device has sufficient processing power, the proper module will be made in accordance with the requested processing power. Then, when all the requested modules are made, in step 14, if the module has an executing task, or it is in the running state and is waiting for the arrival of a task, depending on the requested processing power, it is allocated to the processor and at the end in step 17, the energy consumption will be updated.

```
Input: CT[$dmn$, $rm$], $cetc$, $MaxIteration$, $availablemips$, CDMC[$c_0$,...,$c_i$], MDMC[$m_0$,...,$m_j$]
Output: Created Modules
1  if $cetc$ % $MaxIteration$ == 0 then
2      $availablemips$ = Available Mips Of The Device.
3      for $k$ = 0 to $k < i$ do
4          if $CDMC[c_k] \neq 0$ and $availablemips > 0$ then
5              Create Module Based On Requested Mips Of CT[$k$ , 1].
6          end
7      end
8      for $k$ = 0 to $k < j$ do
9          if $MDMC[m_k] \neq 0$ and $availablemips > 0$ then
10             Create Module Based On Requested Mips Of CT[$k$, 1].
11         end
12     end
13 end
14 if $nrc$ != 0 or $ModuleStatus$==$Running$ then
15     Allocate Requested Mips Of The Module.
16 end
17 Update Energy Consumption.
```

Algorithm 2: Update Allocation Of Mips

4.4. The method analysis parameters

The cost parameter of simulation implementation is as Eq. (6).

$$Cost = PEC + (CC - LUUT) \times RPM \times LU \times TM, \tag{6}$$

where PEC is the cost of the previous execution in the fog device, CC is the Cloudsim clock, LUUT is the last time the system's efficiency is updated, RPM is the million instructions per second (MIPS) rate, LU is the last efficiency rate, and TM is the total MIPS for the host. The overall network utilization is based on Eq. (7):

$$NetworkUsage = \frac{\sum_1^N (TL_i \times TS_i)}{MST}. \tag{7}$$

The relations between the modules are determined by the tuples and the network resources consumption, depending on the size of the tuples that transmitted at a given time. In Eq. (7), TL_i and TS_i are equal to the total delays and the total size of the tuples related to the modules, N the total number of tuples and MST is the maximum simulation time. Moreover, the energy consumed by the simulation for the complete integration of network is calculated by Eq. (8):

$$Energy = CEC + (NT - LUUT) \times HLU, \tag{8}$$

where CEC is the current energy consumption, NT is the current time, LUUT is the last time the system's efficiency is updated and HLU is the last utility of the host.

4.5. Evaluations

We need some resource management techniques to realize the potential power of the fog pattern and the IoT in real-time applications [2]. To analyze the validity of proposed methods in the fog domain and the IoT, ifogsim simulator [31] can be applied to evaluate algorithms in this domain in terms of parameters such as energy consumption, cost, bandwidth, latency, etc. Algorithms 1 and 2 are implemented in terms of different parameters such as the number of department, mobile, cloud user, and electroencephalography (EEG) transmission time. If the EEG is set to lower values, the sensor will send more data at a time unit. In fact, the EEG parameter is the time interval between two consecutive signals sent by the sensors. We compared the FBCS results with the FCFS, random, delay-priority[13], and ECIF [14] algorithms in the same simulation conditions. Based on [13], in FCFS, requests are served in the order of their arrival, until there are no more resources available. In the random method, modules or VMs are randomly selected and resources are randomly allocated to them.

4.5.1. Experiment setup

The simulator is implemented in an environment with characteristics such as the eclipse photon 2018, jdk version 1.8, Intel (R) i7-7500U-2.70GHz processor, 8GB memory, Realtek PCI family controller network card, and Microsoft Windows 10 Enterprise x64 operating system.

4.5.2. Simulation configuration

Tables 4–7 show the VRGAME Application edge configuration; the configuration of all devices in the simulation environment, the connection latency between devices, and the host configuration respectively. In Table 5, the department is the same as a gateway and along with the proxy-server forms the fog devices layer.

Table 4. Vrgame application edge configuration.

Source module	Destination module	Periodicity (mS)	Tuple CPU length (B)	Tuple new length(B)
EEG	Client	0	3000	500
Client	Concentration calculator	0	3500	500
Concentration calculator	Coordinator	100	1000	1000
Concentration calculator	Client	0	14	500
Coordinator	Client	100	28	1000
Client	Display	0	1000	500

As shown in Table 5, each device has parameters including MIPS, RAM[4], upper bandwidth by kilobyte per second (UpBW), down bandwidth by kilobyte per second (DownBW), level in the hierarchical topology, rate per MIPS, busy, and idle power[5]. The details of the simulation outputs are shown in Tables 8–10.

Figures 3–14 show the simulation results. In all figures Dept#, Mob#, User# are the abbreviations of the number of departments, mobiles per department, and number of cloud users, respectively.

[4]kilobyte
[5]megawatt

Table 5. Devices configuration.

Device name	MIPS	Ram	Uplink bandwidth	Downlink bandwidth	Level	Rate per MIPS	Busy power	Idle power
Cloud	44,800	40,000	100	10,000	0	0.01	1648	1332
Proxy-Server	2800	4000	10,000	10,000	1	0	107,339	834,333
Department (Gateway)	2800	4000	10,000	10,000	2	0	107,339	834,333
Mobile (End Device)	500	1000	10,000	10,000	3	0	8753	8244

Table 6. Connection latency.

Device name	Device name	Latency (mS)
Cloud	Proxy-server	100
Proxy-server	Department (Gateway)	4
Department (Gateway)	Mobiles	4
EEG sensor	Mobile	6
Display	EEG sensor	1

Table 7. Host configuration.

Architecture	OS	Storage (B)	BW (B/S)	VM Model	Cost	Cost per Memory	Cost per Storage	Time Zone
x86	linux	1,000,000	10,000	Xen	3	0.05	0.01	10

Table 8. Cost of executions in cloud.

Dept#	Mob#	EEG	User#	FCFS	Random	Delay-priority	ECIF	FBCS
1	10	5	1	1217614.776	1036634.200	2,106,466.244	815,042.982	273,042.278
2	10	5	1	2036455.343	935805.672	2,781,901.794	831,466.857	283,283.885
3	10	5	1	2041544.959	917922.664	3,281,684.457	844,219.371	255,996.799
4	10	5	1	2043366.975	923996.440	3,776,663.969	826,730.400	279,086.768
1	30	10	1	1,758,793.784	1,200,462.224	2,296,131.232	844,551.625	335,145.255
1	60	10	1	2,701,530.722	1,527,898.944	3,793,856.031	842,731.486	498,318.134
1	90	10	1	3,856,418.090	1,960,594.944	2,601,740.734	860,751.114	682,502.044
1	120	10	1	4,438,795.200	2,288,928.600	3,837,440.212	896,027.086	965,456.000
1	20	3	1	3,032,186.147	1,593,817.176	4,258,814.965	821,559.660	457,114.665
1	20	5	1	1,929,523.847	1,216,957.832	2,863,199.411	813,019.914	409,337.963
1	20	7	1	1,614,345.962	1,049,233.248	2,459,280.156	828,724.978	334,075.334
1	20	10	1	1,286,523.384	1,114,612.040	1,643,738.976	829,499.734	238,165.503
1	20	10	2	1,303,686.304	1,068,633.248	1,601,035.522	831,254.788	233,136.523
1	20	10	2	1,291,172.648	1,075,946.168	1,652,372.137	828,735.471	244,995.653
1	20	10	2	1262666.736	1122378.824	1,613,598.055	829,096.914	241,758.310

In Figure 3, the maximum energy consumption belongs to the delay-priority method and FBCS has a minimum value in Dept#4. As Figure 4 shows, the greater the amount of EEG signal is, the less the amount of energy consumption in the cloud will be. The main reason is the optimal use of the bandwidth of the fog nodes.

Table 9. Cloud energy consumption.

Dept#	Mob#	EEG	User#	FCFS	Random	Delay-priority	ECIF	FBCS
1	10	5	1	1.42E+07	1.41E+07	1.48E+07	1.39E+07	1.35E+07
2	10	5	1	1.48E+07	1.40E+07	1.53E+07	1.39E+07	1.35E+07
3	10	5	1	1.48E+07	1.40E+07	1.44E+07	1.44E+07	1.35E+07
4	10	5	1	1.48E+07	1.40E+07	1.60E+07	1.39E+07	1.35E+07
1	30	10	1	1.46E+07	1.42E+07	1.49E+07	1.39E+07	1.36E+07
1	60	10	1	1.52E+07	1.44E+07	1.60E+07	1.39E+07	1.37E+07
1	90	10	1	1.60E+07	1.47E+07	1.52E+07	1.39E+07	1.38E+07
1	120	10	1	1.65E+07	1.49E+07	1.60E+07	1.40E+07	1.40E+07
1	20	3	1	1.55E+07	1.44E+07	1.63E+07	1.39E+07	1.36E+07
1	20	5	1	1.47E+07	1.42E+07	1.53E+07	1.39E+07	1.36E+07
1	20	7	1	1.45E+07	1.41E+07	1.51E+07	1.39E+07	1.36E+07
1	20	10	1	1.42E+07	1.41E+07	1.45E+07	1.39E+07	1.35E+07
1	20	10	2	1.42E+07	1.41E+07	1.44E+07	1.39E+07	1.35E+07
1	20	10	2	1.42E+07	1.41E+07	1.45E+07	1.39E+07	1.35E+07
1	20	10	2	1.42E+07	1.41E+07	1.45E+07	1.39E+07	1.35E+07

Table 10. Average energy consumed by all mobiles per department.

Dept#	Mob#	EEG	User#	FCFS	Random	Delay-Priority	ECIF	FCBS
1	10	5	1	874,501.8013	874,461.9370	867746.4057	875299.9634	837547.6655
2	10	5	1	874,599.9199	874,242.2926	869,507.5515	875,300.0000	837,532.1364
3	10	5	1	874,567.7037	874,353.3993	871,750.2613	875,299.9454	837,270.6748
4	10	5	1	874,587.2341	874,362.0509	872,393.2635	875,299.9845	836,897.6000
1	30	10	1	873,364.0687	873,323.1886	869,502.2295	874,271.2302	827,579.9103
1	60	10	1	871,273.7925	870,883.1754	873,069.1859	871,975.2056	827,485.8266
1	90	10	1	866202.4834	866702.4598	874,203.0132	872,005.5700	827,375.3471
1	120	10	1	863,985.8158	863,716.2340	874,331.9420	868,025.1473	827,356.9812
1	20	3	1	875,300.0000	875,300.0000	874,656.7816	875,300.0000	875,242.6953
1	20	5	1	874,669.6223	874,642.1475	870783.1291	875299.7796	836295.9790
1	20	7	1	853,626.7652	853,707.5507	867,998.8951	875,174.0861	827,379.3979
1	20	10	1	873,503.1414	873,751.7732	866,204.4732	874,560.4189	827,645.7652
1	20	10	2	873,568.3474	873,491.4105	866,035.7913	874,568.5950	827,604.6080
1	20	10	2	873,596.4793	873,696.9294	866,518.9769	874,551.4575	827,650.0398
1	20	10	2	873,819.1439	873,965.5506	866,399.7717	874,569.4272	827,670.8757

Figures 3–6 show that, on average, the FBCS method has reduced the energy consumption in cloud 2.53%, 4.43%, 8.28%, and 10.27% compared to the ECIF, Random, FCFS, and delay-priority, respectively. As Figure 5 shows, increasing the number of mobile devices has increased the energy consumption of the cloud. When the number of mobiles reaches 120, the FCFS method has the highest energy consumption and the FBCS method has the lowest energy consumption with a mobile number of 30. Figure 6 shows that for all methods, changing the number of users does not change much of the energy consumption of the cloud.

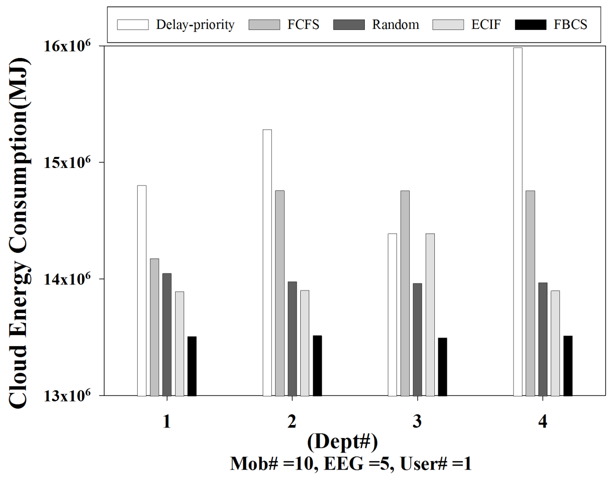

Figure 3. Energy consumed in the cloud based on Dept#.

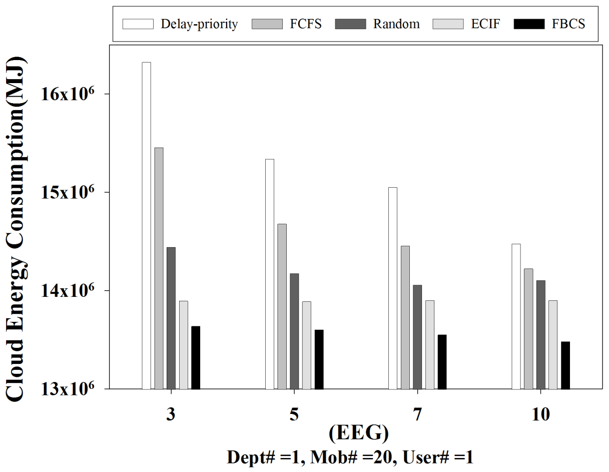

Figure 4. Energy consumed in the cloud based on EEG.

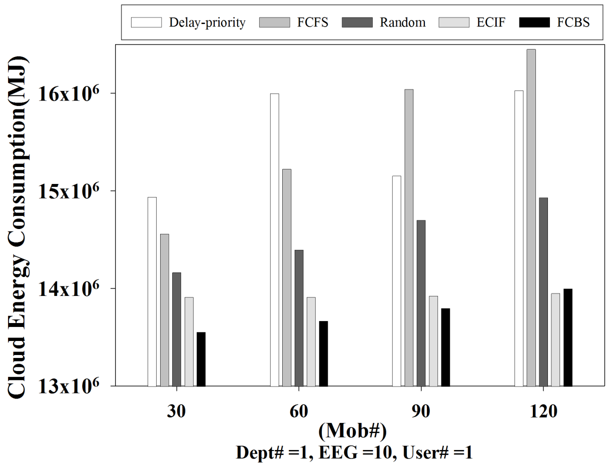

Figure 5. Energy consumed in the cloud based on Mob#.

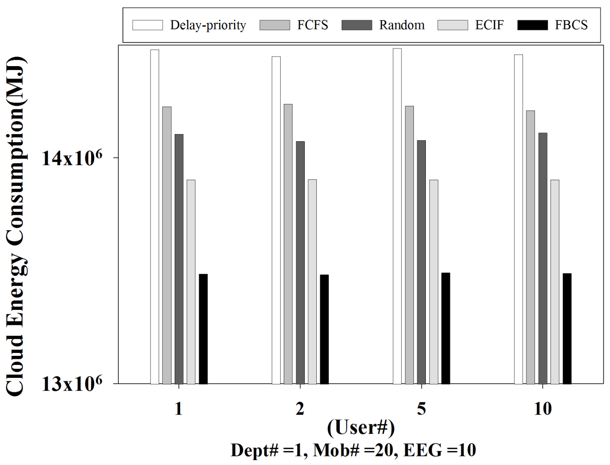

Figure 6. Energy consumed in the cloud based on User#.

Figures 7–10 show that, on average, the FBCS method has reduced the cost of execution in cloud 54.31%, 69.89%, 81.98%, and 85.87% compared to the ECIF, random, FCFS, and delay-priority, respectively. Therefore, FBCS and ECIF methods have the best results in energy consumption and cost with different departments, mobile, user, and EEG signals.

Figures 11–14 show that, on average, the FBCS method has reduced the average energy consumed by all mobiles per department 4.16%, 4.3%, 4.3%, and 4.6% compared to the delay-priority, FCFS, Random, and ECIF, respectively.

Table 11 compares the improvement percentage of FBCS compared to the FCFS, random, delay-priority, and ECIF based on the evaluation parameters.

5. Discussion

The most important feature of devices in the fog environment is their resource constraints, which makes it difficult for them to run real-time programs. One solution to this challenge is to send requests to the cloud, but

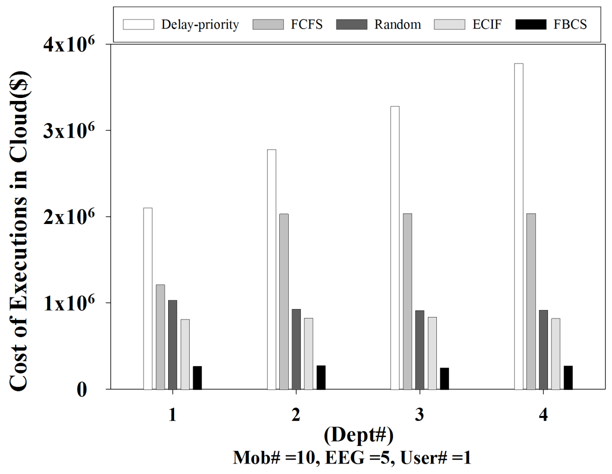

Figure 7. Cost of execution in the cloud based on Dept#.

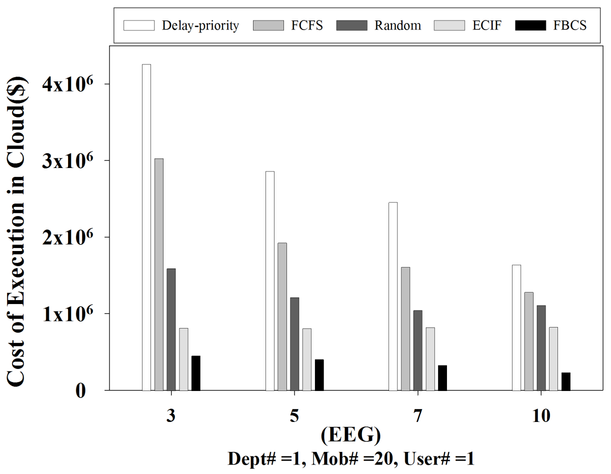

Figure 8. Cost of execution in the cloud based on Mob#.

Table 11. Improvement percentage of FBCS than Delay_Priority, ECIF, Random, and FCFS.

Parameter/Method	FCFS	Random	Delay_Priority	ECIF
Average energy consumed by all mobiles in each department	4.30	4.30	4.16	4.60
Cost of executions in cloud	81.98	69.89	85.87	54.31
Cloud energy consumption	8.28	4.43	10.27	2.53

it may not work in delay sensitive applications. Hence, in real-time programs, a method should be adopted to achieve that:

1. Requests are answered online, and QoS response parameters are provided.
2. Requests are scheduled to use available resources on devices to control network traffic without sending requests to other devices or higher layers (FC and CC).
3. Considering that cloud-based services are pay-as-you-go [33], the cost of maintaining their datacenters is reduced by not sending requests to the datacenters, which reduces costs for users to receive the service.

The results of this research show that the Bayesian method reduces the cost and energy consumption of fog and cloud nodes. The algorithm's execution time is also optimized since the maximum number of replicates is optimally obtained. Therefore, the FBCS method can be used in devices with limited processing resources which allows scheduling to be done online in real-time. The analysis of the results proves that the FBCS method has the best results in terms of energy consumption and cost than other methods. FCFS and delay-priority methods have had poor results in different criteria. Considering IoT, things require low energy consumption, so the proposed method in fog architecture has a superiority over the compared methods.

6. Conclusion and future work

The fog layer has reduced the far distance between CC providers and the end devices. It has also reduced the response time in real-time systems. Intelligent scheduling reduces management costs. For this reason, the method of scheduling, according to ML algorithms, is proposed based on Bayesian classification. The FBCS algorithm has been evaluated in the ifogsim simulator and on the VRGAME application.

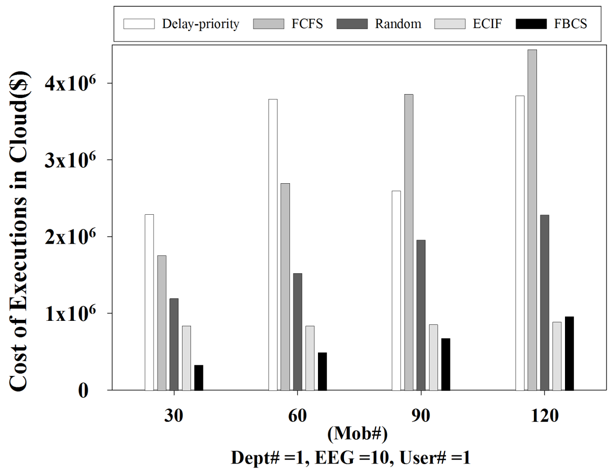

Figure 9. Cost of execution in the cloud based on Mob#.

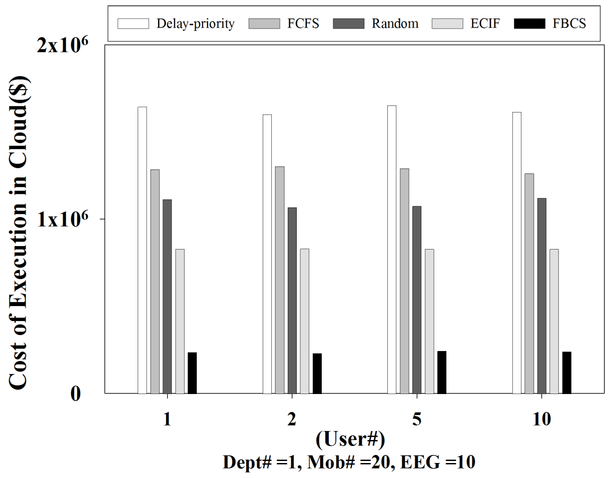

Figure 10. Cost of execution in the cloud based on User#.

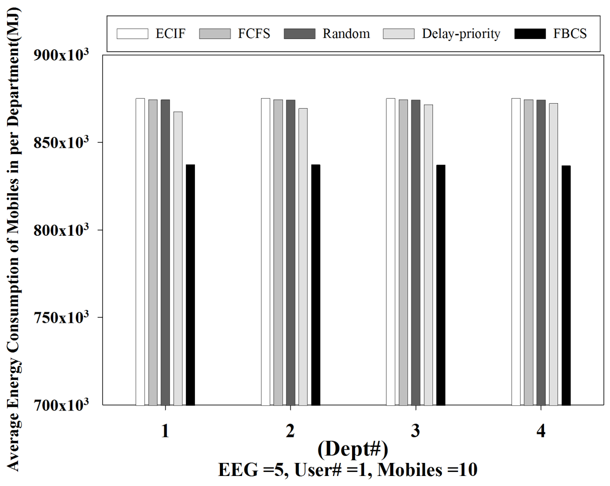

Figure 11. Average energy consumption of mobiles based on Dept#.

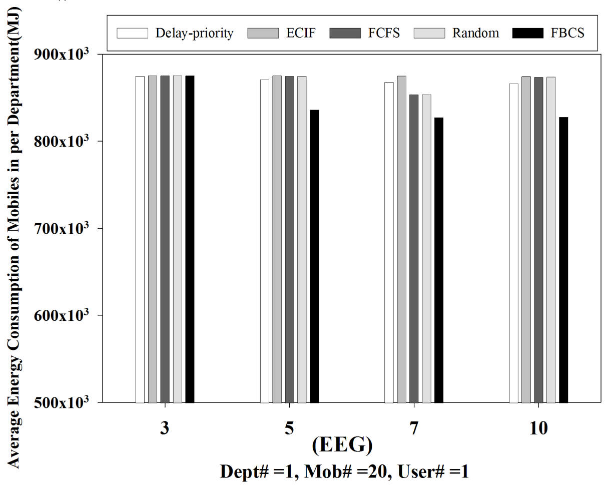

Figure 12. Average energy consumption of mobiles based on EEG

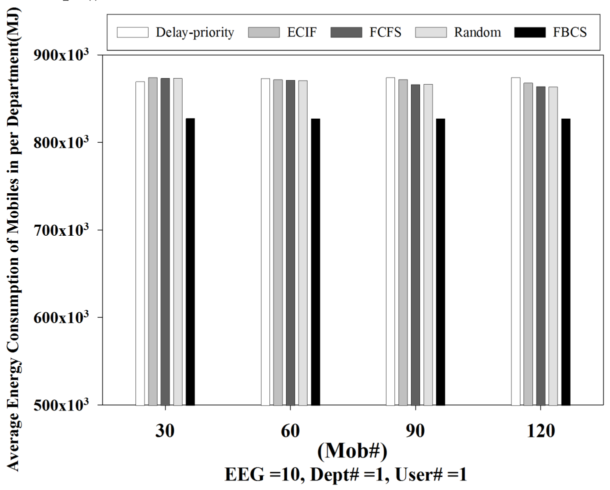

Figure 13. Average energy consumption of mobiles based on Mob#.

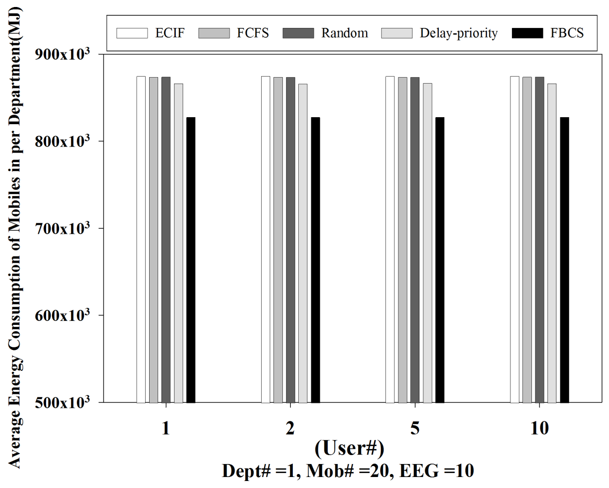

Figure 14. Average energy consumption of mobiles based on User#.

The FBCS method is compared with FCFS, random, delay-priority, and ECIF methods. According to our analysis, FBCS is superior to others. The results show that the amount of energy consumption in the cloud improved minimum 2.53%, maximum 10.27%; cost of executing the tasks in the cloud improved at least 54.31%, maximum 85.87%; and the average of energy consumption of mobile devices minimum 4.16%, maximum 4.6%. The fact that the fog layer and its applications have a lot of dynamicity and the continuous changing conditions is an inseparable part of it suggested that different learning algorithms would be implemented with different environmental conditions then according to the QoS parameters, the best algorithm could be intelligently selected.

In the future, we intend to develop appropriate learning models by using methods such as DL and RL and the implementation of different algorithms under different conditions. Therefore, using these models makes the system learn over time which method would yield a better result in different conditions. Consequently, changing the environmental conditions would affect our method selection and the smart choice of the learning method would be involved in management decision-making.

References

[1] Singh AB, Bhat JS, Raju R, D'Souza R. A comparative study of various scheduling algorithms in cloud computing. American Journal of Intelligent Systems 2017; 7 (3): 68-72. doi: 10.5923/j.ajis.20170703.06

[2] Yousefpour A, Fung C, Nguyen T, Kadiyala K, Jalali F et al. All one needs to know about fog computing and related edge computing paradigms: a complete survey. Journal of Systems Architecture 2019; doi: 10.1016/j.sysarc.2019.02.009

[3] Mutlag AA, Abd Ghani MK, Arunkumar NA, Mohamed MA, Mohd O. Enabling technologies for fog computing in healthcare IoT systems. Future Generation Computer Systems 2019; 62-78. doi: 10.1016/j.future.2018.07.049

[4] Bogale TE, Wang X, Le LB. Machine intelligence techniques for next-generation context-aware wireless networks. CoRR 2018; abs/1801.04223.

[5] Peralta G, Iglesias-Urika M, Barcelo M, Gomez R, Moran A et al. Fog computing based efficient IoT scheme for the Industry 4.0. In: International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics Conference (ECMSM); Donostia-San Sebastian, Spain; 2017. pp. 1-6.

[6] Etemad M, Aazam M, St-Hilaire M. Using DEVS for modeling and simulating a fog computing environment. In: 2017 International Conference on Computing, Networking and Communications Conference (ICNC); Santa Clara, CA, USA; 2017. pp. 849-854.

[7] Stojmenovic I, Wen S. The fog computing paradigm: scenarios and security issues. In: 2014 Federated Conference on Computer Science and Information Systems; Warsaw, Poland; 2014. pp. 1-8.

[8] Rabiul Alam MdG, Tun YK, Hong CS. Multi-agent and reinforcement learning based code offloading in mobile fog. In: 2016 International Conference on Information Networking (ICOIN); Kota Kinabalu, Malaysia; 2016. pp. 285-290.

[9] Yi S, Li C, Li Q. A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 Workshop on Mobile Big Data Conference; Hangzhou, China; 2015. pp. 37-42.

[10] Hussain M, Beg MM. Fog computing for Internet of gs (IoT)-aided smart grid architectures. Big Data and Cognitive Computing 2019; 3(1): 8 doi:10.3390/bdcc3010008

[11] Tang Z, Zhou X, Zhang F, Jia W, Zhao W. Migration modeling and learning algorithms for containers in fog computing. IEEE Transactions on Services Computing 2018; 14 (8): 1-14. doi: 10.1109/TSC.2018.2827070

[12] Zhang P, Zhou M. Dynamic cloud task scheduling based on a two-stage strategy. IEEE Transactions on Automation Science and Engineering 2018; 15 (2): 772-783. doi: 10.1109/TASE.2017.2693688

[13] Bittencourt LF, Diaz-Montes J, Buyya R, Rana OF, Parashar M. Mobility-aware application scheduling in fog computing. IEEE Cloud Computing 2017; 4(2): 26-35. doi: 10.1109/MCC.2017.27

[14] Jalali F, Vishwanath A, De Hoog J, Suits F. Interconnecting Fog computing and microgrids for greening IoT. In: 2016 IEEE Innovative Smart Grid Technologies-Asia (ISGT-Asia); Melbourne, VIC, Australia; 2016. pp. 693-698.

[15] Mathew T, Sekaran KS, Jose J. Study and analysis of various task scheduling algorithms in the cloud computing environment. In: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI); New Delhi, India; 2014. pp. 658-664.

[16] Anusha Bamini BE, Sharmini EME. Optimized resource scheduling using classification and regression tree and modified bacterial foraging optimization algorithm. International Journal of Applied Engineering Research 2015; 10 (16): 37170-37175. doi: 10.1002/cpe.3887

[17] Fang W, Zhou W, Li Y, Yao X, Xue F et al. A distributed admm approach for energy-efficient resource allocation in mobile edge computing. Turkish Journal of Electrical Engineering and Computer Sciences 2018; 26 (6): 3335-3344. doi: 10.3906/elk-1806-112

[18] Borthakur D, Dubey H, Constant N, Mahler L, Mankodiya K. Smart fog: fog computing framework for unsupervised clustering analytics in wearable internet of things. In: 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP); Montreal, QC, Canada; 2017. pp. 472-476.

[19] Yang R, Ouyang X, Chen Y, Townend P, Xu J. Intelligent resource scheduling at scale: a machine learning perspective. In: 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE); Bamberg, Germany; 2018. pp. 132-141.

[20] Hormozi E, Hormozi H, Akbari MK, Sargolzai JM. Using of machine learning into cloud environment (a survey): managing and scheduling of resources in cloud systems. In: 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing; Victoria, BC, Canada; 2012. pp. 363-368.

[21] Cui D, Peng Z, Xiong J, Xu B, Lin W. A reinforcement learning-based mixed job scheduler scheme for grid or iaaS cloud. IEEE Transactions on Cloud Computing 2017; doi: 10.1109/TCC.2017.2773078

[22] Ezugwu AE, Frincu ME, Adewumi AO, Buhari SM, Juniadu SB. Neural network-based multi-agent approach for scheduling in distributed systems. Concurrency and Computation: Practice and Experience 2017; 29 (1): e3887. doi: 10.1002/cpe.3887

[23] Lavassani M, Forsström S, Jennehag U, Zhang T. Combining fog computing with sensor mote machine learning for industrial iot. Sensors 2018; 18 (5): 1532. doi: 10.3390/s18051532

[24] Zhang Q, Lin M, Yang LT, Chen Z, Li P. Energy-efficient scheduling for real-time systems based on deep q-learning model. IEEE Transactions on Sustainable Computing 2017; 4 (1): 132-141. doi: 10.1109/TSUSC.2017.2743704

[25] Zhao X, Zhao L, Liang K. An energy consumption oriented offloading algorithm for fog computing. In: International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness; Greader Noida, India; 2016. pp. 293-301.

[26] Aazam M, Zeadally S, Harras KA. Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. Future Generation Computer Systems 2018; 87. 278-289. doi: 10.1016/j.future.2018.04.057

[27] Zhu J, Song Y, Jiang D, Song H. A new deep-Q-learning-based transmission scheduling mechanism for the cognitive internet of things. IEEE Internet of Things Journal 2018; 5 (4): 2375-2385. doi: 10.1109/JIOT.2017.2759728

[28] Rahmani AM, Liljeberg P, Preden J-S, Jantsch A. Fog Computing in the Internet of Things: Intelligence at the Edge. Switzerland AG: Springer, 2018.

[29] Wang Y, Wang K, Huang H, Miyazaki T, Guo S. Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications. IEEE Transactions on Industrial Informatics 2019; 15 (2): 976-986. doi: 10.1109/TII.2018.2883991

[30] Lu J, Li L, Chen G, Shen D, Pham K et al. Machine learning based intelligent cognitive network using fog computing. In: Sensors and Systems for Space Applications X Conference; Anaheim, California, United States; 2017. pp. 101960G.

[31] Gupta H, Dastjerdi AV, Ghosh SK, Rajkumar B. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Software: Practice and Experience 2017; 47 (9): 1275-1296. doi: 10.1002/spe.2509

[32] Osisanwo FY, Akinsola JET, Awodele O, Hinmikaiye JO, Olakanmi O et al. Supervised machine learning algorithms: classification and comparison. International Journal of Computer Trends and Technology (IJCTT) 2017; 48 (3): 128-138. doi: 10.14445/22312803/ijctt-v48p126

[33] Michael A, Armando F, Rean G, Anthony DJ, Randy KA et al. A view of cloud computing. Communications of the ACM 2010; 53(4): 50-58. doi: 10.1145/1721654.1721672