Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.029337
Article

Tech Science Press

Modelling Mobile-X Architecture for Offloading in Mobile Edge Computing

G. Pandiyan[*] and E. Sasikala

Department of Data Science and Business System, SRM Institute of Science and Technology, Chennai, India
*Corresponding Author: G. Pandiyan. Email: pg2874@srmist.edu.in
Received: 02 March 2022; Accepted: 09 June 2022

Abstract: Mobile Edge Computing (MEC) assists clouds to handle enormous tasks from mobile devices in close proximity. The edge servers are not allocated efficiently according to the dynamic nature of the network. It leads to processing delay, and the tasks are dropped due to time limitations. The researchers find it difficult and complex to determine the offloading decision because of uncertain load dynamic condition over the edge nodes. The challenge relies on the offloading decision on selection of edge nodes for offloading in a centralized manner. This study focuses on minimizing task-processing time while simultaneously increasing the success rate of service provided by edge servers. Initially, a task-offloading problem needs to be formulated based on the communication and processing. Then offloading decision problem is solved by deep analysis on task flow in the network and feedback from the devices on edge services. The significance of the model is improved with the modelling of Deep Mobile-X architecture and bi-directional Long Short Term Memory (b-LSTM). The simulation is done in the Edgecloudsim environment, and the outcomes show the significance of the proposed idea. The processing time of the anticipated model is 6.6 s. The following performance metrics, improved server utilization, the ratio of the dropped task, and number of offloading tasks are evaluated and compared with existing learning approaches. The proposed model shows a better trade-off compared to existing approaches.

Keywords: Mobile edge computing; cloud offloading; delay; task drop; reinforcement learning; mobile-X architecture

1 Introduction

Recently, with the massive advancement in the delay-sensitive and computationally intensive mobile applications like real-time translation services, signal and image processing (for instance, facial recognition), augmented reality, and online gaming imposes enormous computational demands over the resource-based mobile devices [1]. Mobile devices are restrictive in various factors like storage capacity, battery, and computation. In addition, there is a growing demand over the transfer computation or offloading intensive tasks to various powerful resources-based computational environments. It is also termed computational offloading [2]. It diminishes the energy consumption for diverse processing and thus improves battery life. Generally, Mobile-based cloud computing (MCC) helps in computational offloading over the devices. Here, the user devices use resources of various dedicated remote cloud servers for implementing diverse tasks. The server possesses higher computational power, storage capabilities and CPU [3]. Moreover, the longer distance between the cloud server and mobile devices incurs additional energy consumption and latency, significantly reducing performance in network [4]. Thus, the storage and computational ability of the remote locations are migrated towards the edge. It is known as Mobile edge computing [5].

MEC platform offers Cloud Computing services, and information technology services. It is executed by the dense network with various computational servers or enhancing the entities deployed in the prior stage like small cell base stations (BS) with storage and computational resources [6]. The target of the MEC is to fulfil the service distribution and network operation effectually, diminishing the latency and ensures the user experience. Generally, devices offload specific resource demanding applications towards the network edge for timely execution [7]. Smart environments benefit from executing tasks by offloading to edge serves over the framework, also they have monitoring facility to activate action at particular events, and it is known as cyber-physical systems (CPS). It includes drone services or tracking cameras (violation) for geological survey or delivery purposes. CPS process-specific data of their own, forwarding it to the remote cloud [8]. Subsequently, it can enhance the quality of user experience and fulfil service quality-based requirements like energy consumption and lower latency [9]. Unlike MCC, mobile-edge computing functions over the decentralized framework and edge servers are deployed in a distributive manner.

Even though MEC possess huge advantages, there are some research constraints [10]. As demonstrated above, real-time applications are susceptible due to energy consumption and latency. Based on the dynamics and randomness of various MEC, the longer execution time (applications) causes higher energy consumption. Some investigations specify long-term execution is one of the most confronting factors in MEC [11,12]. Thus, there is a huge necessity of performing an efficient computational-offloading model for edge computing. Moreover, constructing an efficient dynamical partitioning approach for appropriate offloading decision-making is most confronting in MEC [13]. Determining task offloading over the multi-edge network environment for reducing the service computing latency (adjacent edge network, proximity edge, or remote cloud) is also another challenging task [14]. The heterogeneity of physical mobile device distribution, user mobility, and edge node resources impose some added computational offloading challenges over edge computing [15]. Fig. 1 depicts the generic view of MEC.

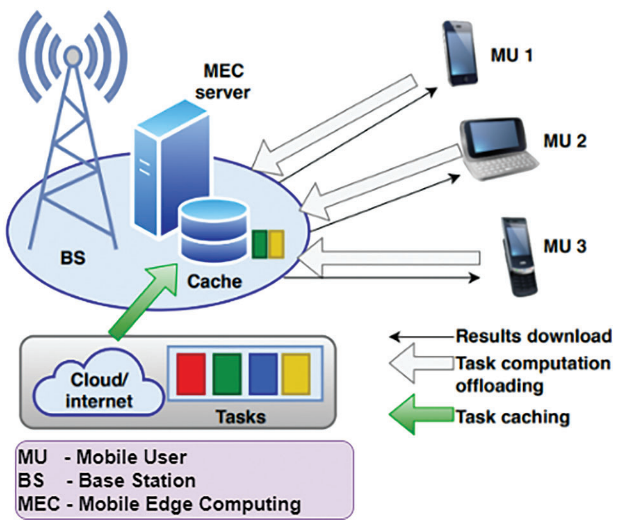

Figure 1: Generic view of mobile-edge computing

There are vast numbers of methods that are modelled to get rid of these challenges. some investigators did not deal with the advantages of adjacent edges for serving offloading tasks while edge servers (nearer) cannot perform these tasks. These techniques are based on the initial workload state and eliminate specific present state scenarios. However, some investigations lack modelling an effectual multi-objective decision-making optimization approaches for choosing the offloading process. The applications of the research are content delivery, computational offloading, video caching, collaborative computing and big data analytics. In this research, many issues are handled and intend to enhance the performance of the anticipated dynamical framework. It considers both users requirements and servers. Specifically, this work is concerned with the computational offloading of mobile edge devices using MEC system utility. It deals with 1) the energy consumption and balancing the processing delay; 2) determination of the mobile application process to offload the task using the reinforcement model known as Deep Mobile-X architecture and bi-directional Long Short Term Memory (b-LSTM); 3) methods to handle the offloading of the multi-edge network process and 4) fulfilling the effectual resource handling of MEC servers.

This research addresses various questions that arise while developing an efficient resource management model for the anticipated MEC server over the multi-edge networking model, including offloading decision-based Deep Mobile-X architecture. In addition, some problems related to mobility management with the mobile devices from one region to the other. Generally, an efficient task offloading and energy-efficient method is a model to improve the MEC-based server utilization via load balancing and scheduling tasks. MEC suffers from diverse constraint computational resources are contrary to centralized MCC. It turns to be an imperative factor in allocating the essential resources proficiently. The anticipated resource allocation facilitates Quality of Service (QoS) and Quality of Experience (QoE) requirements, for instance, latency measure with constraint effort. The significant research contributions are discussed below: Here, a novel MEC system model is designed to handle task offloading, energy consumption, and delay. It is achieved with the modelling of an efficient Mobile-X architectural model using the reinforcement model.

The simulation of the anticipated model is done in the Edgecloudsim environment, and the evaluation of the system model is performed with various performance metrics. The efficiency of the optimal offloading-based decision model is analyzed by handling the preliminary parameters and examining the outcomes that trigger the capabilities of various applications.

The work is organized as: Section 2 provides an extensive insight towards the task offloading in MEC using various mining and learning approaches. In Section 3, the proposed reinforcement learning is designed using Mobile-X architecture and the model efficiently to achieve offloading. In Section 4, the numerical outcomes of the anticipated model are discussed and evaluated with various other approaches to project the model significance. In Section 5, research goals are summarized, and some discussions related to research constraints are also given. The idea to enhance future research is also provided to help the young researchers.

2 Related Works

This section discusses various works related to computational offloading, which is a significant cause for service optimization, time exploitation and resource-based criteria. Various merits and demerits of the existing ideas are presented by considering the QoS, optimization objectives, etc. Panigrahi et al. [16] discussed the mobile application framework, which influences the model performance, offloading and deployment in MCC. The research benefits rely on the taxonomical model with extensive descriptions; however, some drawbacks are encountered in the literature, and it needs to be addressed efficiently.

- Poor and inadequate research to predict the real-time complexities.
- Weak organization of the intermediate nodes;

- ▪ No idea regarding the extension of research works;
- ▪ Incomplete future research ideas;

Liu et al. [17] concentrates on various emerging models with data offloading in mobile devices via diverse communication media like Wi-Fi and cellular networking models in the heterogeneous environment. Its foremost concern is the lack of QoS demands of applications implementing proper computational factors in cloud and fog. The benefits of the work rely on the adoption of learning approaches. Pu et al. [18] demonstrates the idea of internet-based computing models like Mobile Edge Computing (MEC), Mobile Cloud Computing (MCC), and Fog Computing (FC). Some of the closer paradigms of the cloud cannot meet some significant issues like real-time application requirements. It is due to the pervasive exploration of the local environment to implement data-centric applications and necessary entities for the 5G model. Gu et al. [19] discusses the nature of computational offloading in a partial and binary manner. The offloading code is considered for the transmission of remote execution and local execution. Liu et al. [20] chooses some essential metrics during computational offloading and discusses the consequences of total expenses and effectiveness of the MEC system model, which includes QoS, energy consumption, cost and response time. Various other methods are used for fulfilling the offloading needs and metrics devoid of losing generality. Jošilo et al. [21] explains the delay encountered during the task execution. The author describes the term as "sum of delay encountered during local execution, delay offloading, and remote execution where the elapsed time to handle the execution request of the remote server and delay in handling the server response by the locally connected devices. Jin et al. [22] explains the QoS and QoE attained by complementing various cases. Thus, the application-based service access rate needs a resource for running the application and requires time for executing the applications. Wu et al. [23] discuss the response time during computational offloading to enhance the performance. The total time required for task offloading from the local device towards the remote server attains superior performance over the local device. There is some substantial difference between the system latency and response time. The well-established response time over the networking environment is ensured.

Hu et al. [24] discuss the cost required for computational offloading and define it based on every task's remote and local execution. Some metrics related to cost computation is based on task exploitations, task response time and task demand. In the computational environment, the total execution cost comprises remote and local execution costs. This work considers buffering delay and processing delay to motivate the investigators. Vijayabaskar [25] discuss the profits attained during offloading process. There are enormous attributes related to the profit evaluation: energy saving, users' satisfaction and profit, and overall scalability. This process can improve providers and users' profit by optimizing various intermediate devices' costs from source requests to the destination. Moreover, dominant platforms and applications raise the demand while handling the scalability. Zhang et al. [26] discussed the position of the wireless channel and considered some offloading models, and the pattern accessibility of the channels is explained in a deterministic or stochastic manner. Zhou et al. [27] discuss the method for the indirect and direct offloading route. The requests generated by the users are offloaded directly towards the computational serves for execution purposes. The anticipated model does not use intermediate parties as it is simple than other models, i.e., lower performance and higher overhead. The offloading is directly fulfilled by using various repositories.

Hu [28] discuss the delay dependency constraint-based request, and the request for offloading are further categorized into delay-tolerant and delay-sensitive. The response time is measured as an essential factor to fulfil the pre-defined deadlines. The response time does not show a higher significance, and other metrics like energy consumption must be considered. Min et al. [29] discusses various communication capabilities like cellular networks, WLAN, D2D and Wi-Fi communication where Bluetooth is required to fulfil the offloading process. Some delays are encountered during the transmission media constraints while fulfilling the offloading process.

The cost during the offloading process is based on the transmission process and various media used for the transmission process. During the offloading process, the execution delay depends highly on transmission cost, and the transmission rate is considered during the decision-making process. At the same time, for mobile users, the Wi-Fi access point accessibility is an alternative, and the offloading takes enormous time while the users enter the coverage region [30–35]. Based on the learning and big data development, there are modern rends in the intelligent systems like convolutional neural networks are integrated into the detectors like zero forcing, maximal likelihood detectors and minimal mean square zero and it is observed that the detection performance is improved substantially. The Q-learning based algorithm is proposed to predict the communication from attackers and integrated wth the eavesdropping, spoofing, silent modes and interfering model.

Area	References	Model	Purpose
Slicing	[15]	Deep neural networks (DNN)	Adopts spatio-temporal relationship among the traffic patterns
	[16]	Recurrent neural networks (RNN)	Identifies BS pairing for mobile users to form further demand
	[17]	Reinforcement learning (RL)	Slicing strategy is based on prediction and resource requirements
	[18]	DNN	Chose slice and perform effectual balancing
	[19]	DNN	Design a novel network service based on resource utilization
Resource allocation	[20]	DNN	Predict resource capacity and demands based on network probes
	[21]	RL	Identifies traffic demand and provides radio resources among the slices
Caching	[22]	DNN	Diminish energy consumption and computational policy
	[23]	RNN	Predicts user movements and off load task in advance
	[24]	RL	Optimize the caching cost, offloading and computation with mobility and deadline constraints
Offloading	[25]	RNN	Optimize the offloaded tasks and schedules
	[26]	RNN	Reduce energy consumption
Energy	[27]	DNN	Generates energy model and enhances the performance of cloud
Security	[28]	RL	Optimize MEC security to handle unknown attacks.

3 Preliminaries

Consider a MEC system with N BS and M antennas and $N \leq M$ specified by $N = \{1, 2.., N\}$. The BS is directly connected with MEC with D cache size (bits) and F computing capacity. The time is partitioned with slot T_s and shown as $T = \{0, 1, ..\}$.

3.1 Task Model

Here, $'k'$ heterogeneous tasks are specified by the $k \triangleq \{1, 2, \ldots, K\}$ where the task is represented with two diverse parameters: total computing resources and d_k is cycle/second. The numbers of mobile units are considered to deal with a huge amount of tasks $(N \geq K)$. Some computational tasks show higher popularity with constant user requests and execution. In the initial stage, the mobile unit requests the task from set K, and multiple users concurrently generate a request for performing certain tasks. $z_t^n \in$ as the task request from n^{th} unit with time slot $'t'$ and $k_t \triangleq [k_t^1, \ldots, k_t^N]^T$ is task request for all MU. The task request is shown in Eq. (1):

$$\varnothing_{k,t} = \frac{z_{k,t}^{-\eta}}{\sum_{l=1}^{K} z_{l,t}^{-\eta}} \tag{1}$$

Here, η manages the popularity skewness and is set as $'0'$ for task popularity. When the η value is higher, then it specifies that the tasks are larger.

3.2 Cache Model

Here, $c_{k,t} \in \{0, 1\}$ specifies caching decision for every $'k'$ slot in $'t_0$ time. The task is cached in the edge server at $'t$o time, and related data is used in successive time slots $'t+1'$ for task execution. Here, the provided system model does not experience any energy cost and delay for data transmission towards a particular task. Thus, the QoE is highly experienced by the users. Generally, the task cannot be cached because of the constraint server storage size. Thus, it is mathematically expressed as in Eq. (2):

$$\sum_{k=1}^{K} 1\ (c_{k,t} = 1\ )b_k \leq D; \quad \forall_t \in T \tag{2}$$

$$C_t = \sum_{k=1}^{K} 1\ (c_{k,t-1} = 0; \quad c_{k,t} = 1, \ \ k \ \in K_t)_{g_k} \tag{3}$$

Here, $1(\varepsilon)$ is provided with two values, $'1'$ *and* $'0'$. 1 is given when the event is actual; else, the event is not valid. K_t^0 is a set of tasks intended to offload in mobile units at time slot $'t'$. Here, g_k is fetching cost for downloading input data for task k from remote cloud server, g_k depends on size of task data. Finally, D is cache capacity of MEC server.

3.3 Reinforcement Learning (RL)

The RL model is composed of agents, communication environment and state action (S'), available actions and reward function. The agent intends to learn constantly and performs decisions via interaction with the environmental setup discretely. Fig. 2 shows the model of reinforcement learning [28]. In every time frame $'t'$, the agent monitors the state environment $s_t \in S'$ and takes action $a_t \in A$. The nature of the agent is determined based on the policy and consider deterministic policy μ and maps the state action, i.e., $\mu: S' \rightarrow A$. After action execution, the MEC environment gives reward $r_t = R\ (s_t, \ a_t)$ and transitions state from $s_t \rightarrow s_{t+1}$. The discount return is depicted as the sum of rewards attained with the MA. It is expressed as in Eq. (4):

$$R\left(\xi = \sum_{t=0}^{\propto} \gamma^t r^t\right) \tag{4}$$

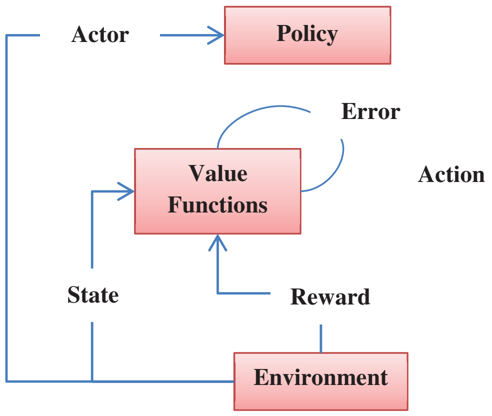

Figure 2: Reinforcement learning model

Here, $\gamma \in [0, \ 1]$ specifies the discount factor, and $\xi = (s_0, \ a_0, \ s_1, \ a_1)$ specifies the sequential state and action related to the reward sequence $\{r_t\}_{t=0}^{\propto}$. The action function $Q^{\mu} \ (s, \ a)$ specifies the return under policy with an initial state, and it is expressed as in Eq. (5):

$$Q^{\mu} \ (s, \ a) = \ \mathbb{E}[R(\xi)|s_0 = s, \ a_0 = a] \tag{5}$$

Here, the expectation is done in a random environment. RL agent target is to learn the optimal policy μ^* selects the optimal action in state s. It is expressed as in Eq. (6):

$$\mu^*(s) = \arg\max_a \ Q^* \ (s, \ a) \tag{6}$$

where $Q^* \ (s, \ a)$ is optimal function and handles the problem in the RL framework. The system state is determined with the set of parameters, and arbitrary time slot $'t'$ is expressed as in Eq. (7):

$$s_t \triangleq \{k_t, \ H_t, \ c_{t-1}\} \tag{7}$$

Here, k_t specifies task request, H_t specifies channel matrix, c_{t-1} specifies caching decision based on the previous slot, representing MEC server cache status. The system randomness is measured by the state variables H_t and k_t. During the initial slot, the available systems and the uplink transmission are evaluated with channel measures. The state vector dimension is shown as $K + (M+1)N$. Based on the MEC system state (observed), the learning agent needs to select the action using the decision variables, and it is expressed as in Eq. (8):

$$a_t \triangleq \{c_t, \ x_t, \ b_t, \ f_t\} \tag{8}$$

Here, c_t specifies the caching decision, a_t specifies offloading decision, b_t specifies transmission power (allocation), and f_t specifies the MEC computational resource allocation decision. The state pair to reward is expressed as in Eq. (9):

$$r_t = R \ (s_t, \ a_t) = -J_t \tag{9}$$

The RL algorithm enhances the discount return, and it is used for approximating the non-discount return. The approaches show the recursive relationship among the action-state function, which is expressed using the Bellman equation. It is expressed as in Eq. (10):

$$Q^* \ (s, \ a) = E \ [R(s, \ a) + \gamma \ \max_{a'} Q^*(s', \ a') \tag{10}$$

Here, s' specifies the successive state of the transition attained from the state under action a. It is tough to attain the appropriate outcomes with reinforcement learning with high dimensional action and state space. Thus, this issue is tackled by attaining a solution with Mobile X architecture and the LSTM model.

3.4 Mobile-X Architecture

Mobile Agents (MA) functionality is assessed among the peers for environmental monitoring. The agent has the competency to reduce the exploration burden among the unseen and unvisited states (environment). Therefore, an efficient learning process is proposed to design an efficient and accurate model to address these issues over a certain period, specifically over a complex environment. This work anticipates a learning-based approach using a tree structure to establish coordination among the mobile agents with lesser memory utilization. This work designs Mobile-X architecture for the MA system with tree-structuring. The tree construction for the real-time agents is used to analyse virtual experiences like elapse time during tree-structure mining with re-sampling and associative rule mining with grafting branches. This structure deals with the MA functionality, which experiences P2P connectivity instead of merging the available MA. In Mobile-X architecture, assume that the mobile network architecture is composed of time slots and epoch tuples. Various cases are noted from mobile communication via the considered mobile agents. It is explained below:

Case 1: Formally, the mobile data streams are defined with an infinite epochs sequence, where the mobile data streams are given as $D_1, \ D_2, \ \ldots D_n$ where $E_{TS(r)}, \ r \in [1, \ n]$. Here, $'r'$ is epoch received. The epoch is determined to be tuples $E(E_{TS}, \ Y)$. Sliding windows $'W'$ is considered a set of epochs among r^{th} and s^{th} (s > r) epochs, and $'W'$ is window size with $|W| = s - r$. Mobile Data Streams (MDS) with sliding window (SW) is composed of three different batches. When $'M'$ epoch, $'N'$ batches of $'W'$, for all batches composed of M/N epochs. Hence, the batch size is expressed as $\left|\frac{M}{N}\right|$. Here, SW is determined as batch-to-batch, i.e., sliding window that accelerates batches and avoids initial batches from the current window.

Case 2: (Data patterns support from sliding window ‘W’): Data stream patterns support $'X'$ in window $'W'$ is represented as $Support_W(X)$ represents the number of epochs in $'W'$ composed of $'X'$. Therefore, data patterns are known as frequent SW $'W'$ if support is less than $\min support$, i.e., $0 \leq min\ sup \leq |W|$.

Case 3: (Data pattern association ‘X’ in ‘W’): Data pattern ‘X’ is known as *the* $'W'$ association pattern; if confidence is higher or equal to mobile transactions of ‘W’. For given MDS, |W|, $\min conf$, and $\min sup$; problem-related to data stream mining determines the complete pattern in |W| where these measures are lesser than confident thresholds of received patterns over MDS.

Support: The generated rules with support (sup) in the mobile transaction; if support % of transactions is composed of $D_1 \ \cap \ D_2$. The transactions probability generated by A and B is given.

Confidence: Rules of ‘T’ with confidence ($conf$) of the complete transactions are composed of D_1 and D_2.

3.4.1 Mobile-X, Hierarchical Tree Model

The Mobile-X tree structure is designed with an ordered structure of pre-defined mobile nodes in a canonical form, i.e., ascending or descending order. It is modelled by reading the epoch from pre-defined MDS and maps the epoch of the prefixed path. Hence, prefix tree is represented by data stream (compressively) while other epochs hold various data. The overlapping of the path type is expressed as prefixed sharing. It is given in a compressive form during the process of prefix-sharing. As a result, the prefix sharing provides enormous gain over the mining process. Initially, the proposed Mobile-X tree structure is modelled with evaluating epochs from the scanned mobile node. However, MDS is a continuous, unbounded, and ordered data sequence. Thus, it is not appropriate to maintain the complete elements from MDS in a tree form over a specific time. Thus, the prior information attained is outdated, and the present information is more appropriate from the knowledge discovery phase. The anticipated tree structure is designed with a SW model to sense the present epochs for handling this issue.

3.4.2 Mobi-X Tree Functionality

The mobile-X tree structure is the prefix tree construction constructed based on the MDS to node connectivity. The restructuring of trees in descending order and tree compression is done by integrating supportive mobile nodes in the tree branches. Then, pattern growth is adopted to mine the data patterns from the tree structure. The tree construction includes two phases: insertion and compression, based on a mobile node database. During the insertion process, it organizes the nodes in the database. It is designed with the insertion of epoch in the database one after another. Thus, the Mobile-X tree maintains the nodes' order list. Similarly, 'NO' specifies the distinctive node over every epoch and comprises support values of the item in the database. Initially, the tree is empty with no proper branches and introduces a null root node. In mobile node database, the first epoch is specified as TS = 1; $\{D_1D_2D_3D_4D_7D_8\}$ is inserted to tree $< \{\ \} \rightarrow D_1{:}\,1 \rightarrow D_2{:}\,1 \rightarrow D_3:1 \rightarrow D_4:1 \rightarrow D_7:1 \rightarrow D_8{:}\,1 >$ threshold. Hence, the primary tree branch is constructed with D_1 as the root node (primary node), and D_8 is a final node. The mobile node entries $D_1D_2D_3D_4D_7D_8$ need to be updated. Before the insertion, mobile nodes of $TS = 2$ are sorted over $\{D_1,\ D_5,\ D_6\}$ and ordered as $\{D_1,\ D_2,\ D_3,\ D_4,\ D_7,\ D_8,\ D_5,\ D_6\}$ to maintain NO and perform insertion to tree $TS = 2$. Then, by performing all the epochs $(TS = 6)$. Each node is composed of frequent incidence (epochs). Finally, the NO-list is expressed as NO. Then, it inserts and restructures the compression phase.

The target of the restructuring compression stage is to acquire a compact tree structure with lesser memory and faster computation. Initially, sort NO in descending order with merge sort and restructure the hierarchical model in descending order. Here, tree restructuring the tree approach is used to re-model the Mobile-X tree, and it is known as branch sort approach known as CP-tree (data compression phase). The branch sorting uses merge sort to the tree for path structure. It avoids unsorted paths; however, sorting the complete path and re-inserts into a tree. A simple and efficient compression model select support mobile nodes over the branches and merges them into the nodes. At last, the Mobile-X tree is compressed and structured. Although the Mobile-X tree and CP tree has three-phase construction-based resemblance, there exist some differences.

1) Mobile-X tree performs compression to integrate the support nodes (single nodes), making it more compact. It handles lesser nodes than the CP tree.
2) However, the hierarchical tree memory is lesser than the CP tree.
3) CP tree makes use of FP-growth based mining process for constructing frequent patterns. Hence, FP mining is not directly connected with the hierarchical tree as it does not mine the frequent patterns; but associates with the entire patterns. Thus, the mining (pattern) approach dealt with the added Mobile-X tree feature.

3.5 Mobile Agents' State Action

Similar to the communication functionality carried out by various networking models, the mobile agent maintains the action process and continuous state. It is known as continuous state-action pairs. While the agents pretend to perform pairing operations, it initiates enormous error for establishing temporal difference based learning and transfers state-action pair to other agents. When the successive agent receives the action pair, it classifies it as a leaf node to own tree. It validates appropriate knowledge-based leaf fitting. If it is not carried out, agents have to tag the data as anonymous data and transfer the request to available agents without discarding the request. The unknown regions need to fulfil the condition, and it is expressed in Eqs. (11)–(12):

$$N_{total} = \sum_{j=1}^{C_s} N_j^a < cN \tag{11}$$

(or)

Continuous state $C_s = (x_1,\ x_2,\ \ldots,\ x_n)$

$$\bigcup_{j=1}^{C_s}\left(\left\{x_i|x_i\ \in [\mu_{i,j}^a - \sqrt[n]{(\sigma_{i,j}^a)^2},\ \ \mu_{i,j}^a + \sqrt[k]{(\sigma_{i,j}^a)^2}],\ \ i = 1,\ 2,\ ..,\ n\right\}\right) \tag{12}$$

From Eq. (12), $'C'$ is constant, $'N'$ is the threshold, i.e., the sum of samples over leaf-node with a confidence range, where $\mu_{i,j}^a$ and $\sigma_{i,j}^a$ is mean and variance of continuous state with $'i'$ dimension. The mean and variance values are evaluated for analyzing squared sums of continuous states and the sum of continuous states of all clusters in leaf nodes. When the agents are not provided under the confidence range of clusters, Eq. (12) is fulfilled. The mobile agents need to transmit the request generated from continuous state-action pairs of other MA. Subsequently, the agents need to receive the state-action pair request and provides the leaf node information where the area is overlapped among the other pair.

Algorithm 1

```
Input : Parameter initialization;
Output: Q-value, rewards $'\boldsymbol{r}'$ and stationary policies $\boldsymbol{\pi}^*$
1.  Initialize action function/profiles, replay function, state;
2.  While max $\neq$ iteration do
3.      if probability is determined then
4.          Perform action movement; //random movement
5.      else
6.          Perform action function;
7.          Compute periodic rewards and successive state $'\boldsymbol{s}'$;
8.      Store state, replay, reward and successive state values in relay function;
9.  Iterate transition from replay function; //random movement
10. Evaluate successive transition;
11. if $\boldsymbol{ss}'$ at terminal state then
12.     $\boldsymbol{C_s = (x_1,\ x_2,\ \ldots,\ x_n);}$    //continuous state
13. else
14.     $\bigcup_{j=1}^{C_s}\left(\left\{\boldsymbol{x_i|x_i}\ \in [\mu_{i,j}^a - \sqrt[n]{(\sigma_{i,j}^a)^2},\ \ \mu_{i,j}^a + \sqrt[k]{(\sigma_{i,j}^a)^2}],\ \ \boldsymbol{i = 1,\ 2,\ ..,\ n}\right\}\right);$
15. Train network architecture;
16. Return Q value;    //reward and action profile
```

3.6 Bi-directional LSTM Model

The general LSTM network model possesses various layers and deals with various MEC network features. It needs substantial time to perform the computation and offloading process, and the bus stream consumes less time to make a network flow. However, there is some significant delay while performing task offloading to a cloud environment. Thus, MEC requires bi-directional LSTM. The significant MEC characteristics like data features, time interval features and multi-dimensional features adopt bi-directional

LSTM. The uni-directional flow performs independently, devoid of sharing any general computation, while the bi-directional LSTM model performs the computing process in parallel by avoiding computational duplication. The standard LSTM model does not show two-fold benefits; however, the bi-directional model performs parallel processing with multiple computing servers. It diminishes the computational cost. MEC facilitates a business-based cloud computing (CC) platform with a radio access network nearer to deal with delay-sensitive applications. The nodes with proximity are connected with the MEC server for advanced services. The servers are connected with backhaul communication. The optimization with the MEC framework is concentrated on partitioning the network model. In the Mobile-X architecture, the network partitioning shows a cutting point and realize the layer from $1 \rightarrow N$ and locally run at the edge. The overall time consumption is reduced. The network is trained with the multi-layer architectural model facilitating the base layers to perform some essential tasks with lesser precision (sparse layer) and ready to execute the task.

The bi-directional network model can extract lower-level dimensionality (input data representation) and shares the data among the multi-tasking model. Thus, the light of the LSTM model is shed among the MEC. The resource allocation is performed over the multiple mobile edges; therefore, the edges can attain various tasks and perform the bi-directional function parallel. In context to MEC, the multi-tasking functions of LSTM are used over the edge-computing servers to perform computation with nearer proximity. The server is competent in performing complex tasks with lower computational latency. The servers are nearer to the end-users, and the devices do not require time for data transmission—the benefit of MEC and bi-directional LSTM help in offloading and pretends to enhance the computational speed. The LSTM layers are included with the computation parties where the sub-network predicts the time and data. The sub-network model passes via the hidden layer tier to adjust the data dimensionality to complete the task with time-series prediction. The input of the bi-directional LSTM is merged with the time interval and data features. Both the features are processed for prediction purposes. The bi-directional LSTM is optimized for diminishing workload optimization and time consumption. This research concentrates on MEC server deployment, and a multi-tasking network needs to be analysed under MEC structural model as shown in Figs. 3 and 4.

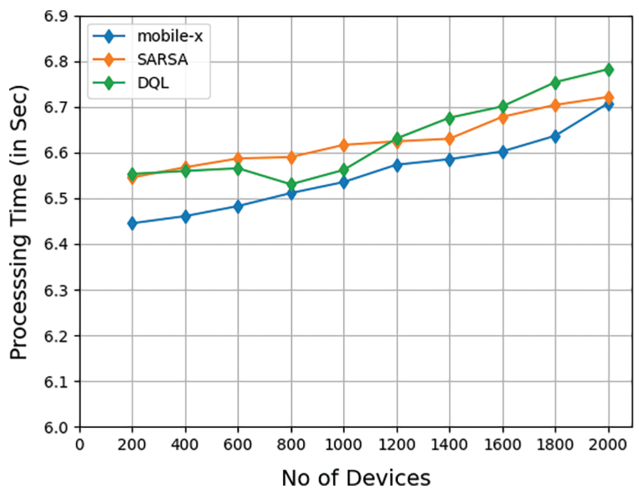

Figure 3: Average processing time of tasks

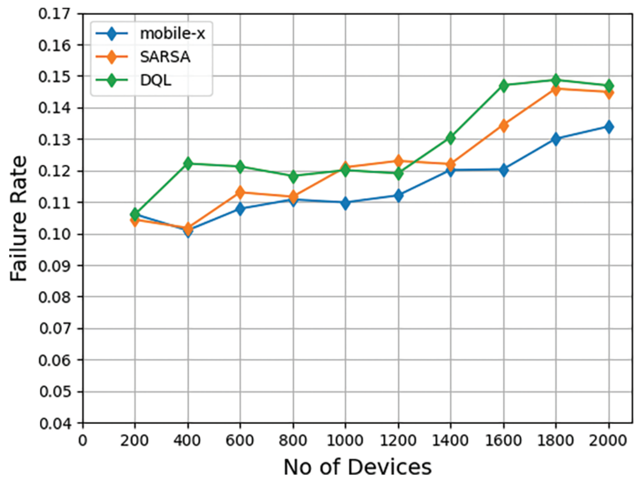

Figure 4: Failure rate over no of tasks

4 Results and Discussion

This section discusses the performance of the system when implying mobile x architecture in different scenarios. The failure rate of the task in the system due to offloading methods and virtual machines utilization is studied. Comparing the result with similar models like SARSA and DQL, which uses reinforcement learning for offloading decisions. Results show mobile x architecture outperforms other competitive models in terms of failure rate, server utilization, processing time, etc.

4.1 Simulation Setup

The simulation is setup in Edgecloudsim simulation in Eclipse 2021 version. Here, $'N'$ tasks are considered for M users for anticipating the best action time to perform offloading. The users can randomly provide the input dataset where the user intends to predict the optimal offloading policy. It is expressed based on several users and tasks over the MEC network. Consider that there are 200 to 2000 mobile users, and every user performs N tasks. The local processing time of the mobile devices are set as $3.75 * 10^{-7}\ s/bit$, and the power consumed during this process is $3.55 * 10^{-6}\ J/bit$. The task size ranges from 15 to 40 MB. Some other parameters like bandwidth (downlink and uplink) among the edge server and the user is set as 200 MB, and it may vary based on the network conditions.

This simulation is tested with different task models with unique pattern on the service in the cloud. Tasks are categorized as infotainment, augmented reality, heavy task, healthcare, The edge side is equipped with 12 servers each with different range of operation; this server are packed with 8 to 12 virtual machines. Configuration of VM includes 10k MIPS and 2 GB of ram.

4.2 Failure Rate and Processing Time

Comparison of the average processing time for the tasks from devices in edge server when applying proposed mobile-X architecture , DQL and SARSA algorithm with 100 to 2000 devices shown in Fig. 3. Proposed model gives better overall performance in solution to tasks offloaded to edge servers by predicting the upcoming request and allocating server for faster response.

Task processing in edge side from devices failed due to many reasons. One among them is due to unbalanced scheduling among edge servers. Some of the servers are under loaded and some of them overloaded due to insufficient knowledge on the load balancing. Our proposed model analysis the load distribution among the servers and their virtual machines in timely manner to identify the appropriate server to done a job. Fig. 5 Shows the failure rate comparison of mobile x model with SARSA and DQL model.

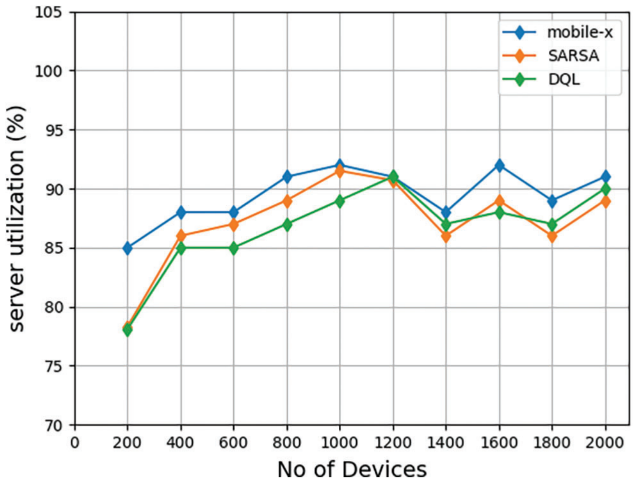

Figure 5: Comparison on server utilization

The average utilization of the server (edge, cloud) when running 100 to 2000 devices with different computation scenario is analysed. Fig. 6 Shows the comparison with DQL and SARSA, our proposed model utilize the server efficiently in most of time.

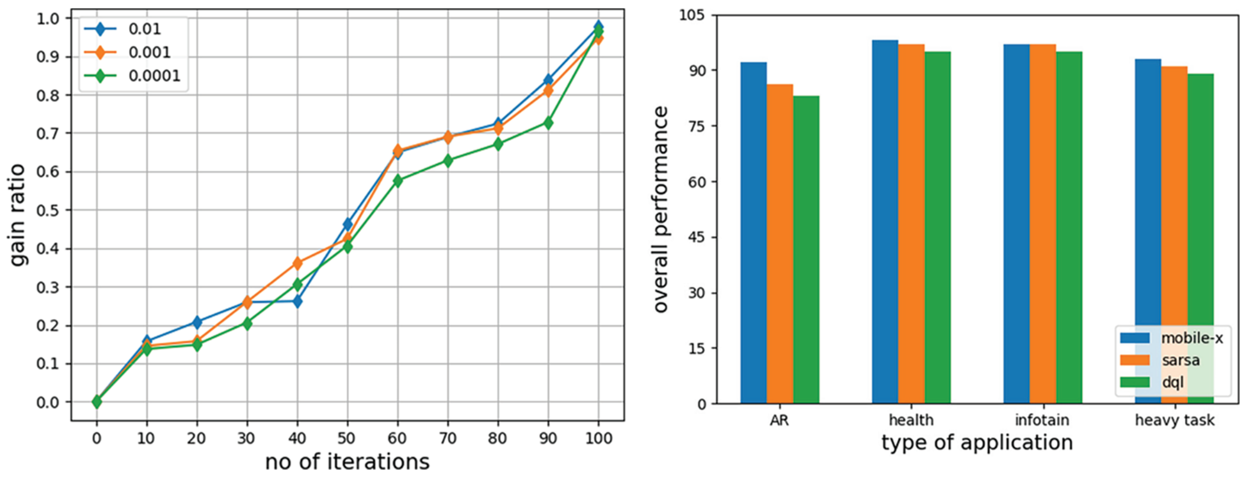

Figure 6: Gain ratio and performance comparison

4.3 Learning Efficiency and Performance

Gain ratio of mobile-x architecture under different learning rate is shown in Fig. 6. Learning rate is adjusted from 0.1 to 0.0001 and gain ratio is monitored on each rate and graphed. Shows the gain value of our proposed mobile x model when learning rate is set in different values.

A comparison of the performance of the proposed model with other models in different scenarios is shown in the Fig. 5. Tasks from different scenarios differ in terms of task size, the need of CPU, priority, processing type, frequency of requests made, etc. for example heavy computation scenarios have a heavy computation task that requires more VMs to process it. However, health care application needs computation but need more storage access. It shows that the proposed model gives a steady performance in all situations where as others give better in one scenario and reduced in others.

5 Conclusion

This investigation shows that the MEC is composed of various layers like access points, servers (multi-edge), and mobile units. Here, mobile units act independently for real-time task offloading. The connectivity is established among the mobile units and the mobile/network access point. This process is performed locally over the remote mobile units, and the offloading process is achieved in three diverse servers: remote, adjacent, and near-edge servers. Here, a novel reinforcement learning approach with Mobile-X architectural model is designed to handle the offloading issues and to offer a better decision-making process regardless of the system cost (time delay and energy consumption). It is noted that the anticipated model outperforms various other approaches like SARSA, Q learning and deep Q-model. Thus, the offloading with a nearby or adjacent server resolves this issue and faces diverse issues identified in CPS. Therefore, it attains optimal outcomes in all resource means. The major research constraints are the modelling of various performance metrics and the evaluation of these metrics. In future, an extensive analysis with the optimizer is done to enhance the Mobile-X architecture model performance.

Funding Statement: The authors received no specific funding for this study.

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

References

[1] F. Fu, Z. Zhang, F. R. Yu and Q. Yan, “An actor-critic reinforcement learning-based resource management in mobile edge computing systems,” *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 8, pp. 1875–1889, 2020.

[2] M. Masdar, F. Salehi, M. Jalali and M. Bidaki, “A survey of PSO-based scheduling algorithms in cloud computing,” *Journal of Network and Systems Management*, vol. 25, no. 1, pp. 122–158, 2017.

[3] M. Masdar and M. Zangakani, “Green cloud computing using proactive virtual machine placement: Challenges and issues,” *Journal of Grid Computing*, vol. 18, no. 4, pp. 727–759, 2020.

[4] M. Ghobaei-Arani, A. Souri, F. Safari and M. Norouzi, “An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing,” *Transactions on Emerging Telecommunications Technology*, vol. 31, no. 2, pp. e3770, 2020.

[5] M. Ghobaei-Arani, A. Souri and A. A. Rahmanian, “Resource management approaches in fog computing: A comprehensive review,” *Journal of Grid Computing*, vol. 18, no. 1, pp. 1–42, 2020.

[6] M. Ghobaei-Arani and A. Souri, “LP-WSC: A linear programming approach for web service composition in geographically distributed cloud environments,” *The Journal of Supercomputing*, vol. 75, no. 5, pp. 2603–2628, 2019.

[7] Y. Sun, T. Wei, H. Li, Y. Zhang and W. Wu, “Energy-efficient multimedia task assignment and computing offloading for mobile edge computing networks,” *IEEE Access*, vol. 8, pp. 36702–36713, 2020.

[8] M. Hu, D. Wu, W. Wu, J. Cheng and M. Chen, “Quantifying the influence of intermittent connectivity on mobile edge computing,” *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 619–632, 2022.

[9] Z. Han, D. Niyato, W. Saad, T. Başar and A. Hjørungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*, Cambridge: Cambridge University Press, 2012.

[10] P. J. Escamilla-Ambrosio, A. Rodríguez-Mota, E. AguirreAnaya, R. Acosta-Bermejo and M. Salinas-Rosales, “Distributing Computing in the internet of things: Cloud, fog and edge computing overview,” in *NEO 2016*, Springer, Cham, pp. 87–115, 2018.

[11] E. Ahmed and M. H. Rehmani, “Mobile edge computing: Opportunities, solutions, and challenges,” *Future Generation Computer Systems*, vol. 70, pp. 59–63, 2017.

[12] R. Roman, J. Lopez and M. Mambo, “Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges,” *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[13] F. Yu, H. Chen and J. Xu, “Dynamic mobility-aware partial offloading in mobile edge computing,” *Future Generation Computer Systems*, vol. 89, pp. 722–735, 2018.

[14] L. Liu, X. Qin, Z. Zhang and P. Zhang, “Joint task offloading and resource allocation for obtaining fresh status updates in multi-device MEC systems,” *IEEE Access*, vol. 8, pp. 38248–38261, 2020.

[15] R. Mahmud, S. N. Srirama, K. Ramamohanarao and R. Buyya, “Quality of experience (QoE)-aware placement of applications in Fog computing environments,” *Journal of Parallel Distributed Computing*, vol. 132, pp. 190–203, 2019.

[16] C. R. Panigrahi, J. L. Sarkar and B. Pati, “Transmission in mobile cloudlet systems with intermittent connectivity in emergency areas,” *Digital Communications and Networks*, vol. 4, no. 1, pp. 69–75, 2018.

[17] X. Liu, H. Tian, L. Jiang, A. Vinel, S. Maharjan *et al.,* “Selective offloading in mobile edge computing for the green internet of things,” *IEEE Network*, vol. 32, no. 1, pp. 54–60, 2018.

[18] L. Pu, X. Chen, J. Xu and X. Fu, “D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration,” *IEEE Journal on Selected Areas Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[19] F. Gu, J. Niu, Z. Qi and M. Atiquzzaman, “Partitioning and offloading in smart mobile devices for mobile cloud computing: State of the art and future directions,” *Journal of Network Computer Applications*, vol. 119, pp. 83–96, 2018.

[20] J. Liu and Q. Zhang, “Code-partitioning offloading schemes in mobile edge computing for augmented reality,” *IEEE Access*, vol. 7, pp. 11222–11236, 2019.

[21] S. Jošilo and G. Dán, “Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 207–220, 2018.

[22] X. Jin, Z. Wang and W. Hua, “Cooperative runtime offloading decision algorithm for mobile cloud computing,” *Mobile Information Systems*, vol. 2019, no. 8049804, pp. 1–17, 2019.

[23] S. Wu, C. Niu, J. Rao, H. Jin and X. Dai, “Container-based cloud platform for mobile computation offloading,” in *Proc. of IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, Florida, USA, pp. 123–132, 2017.

[24] Q. Hu and W. Yue, *Markov Decision Processes with their Applications*, vol. 14. Berlin: Springer Science & Buisness Media, 2007.

[25] M. S. Vijayabaskar, “Introduction to hidden Markov models and its applications in biology,” in *Hidden Markov Models*, Totowa: Humana Press, pp. 1–12, 2017.

[26] X. Zhang and Y. Cao, “Mobile data offloading efficiency: A stochastic analytical view,” in *Proc. of IEEE Int. Conf. on Communications Workshops (ICC Workshops)*, Kansas City, USA, pp. 1–6, 2018.

[27] W. Zhou, W. Fang, Y. Li, B. Yuan, Y. Li *et al.,* “Markov approximation for task offloading and computation scaling in mobile edge computing,” *Mobile Information Systems*, vol. 2019, no. 8172698, pp. 1–12, 2019.

[28] R. Q. Hu, “Mobility-aware edge caching and computing in-vehicle networks: A deep reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10190–10203, 2018.

[29] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu *et al.,* “Learning-based computation offloading for IoT devices with energy harvesting,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, 2019.

[30] S. W. Ko, K. Han and K. Huang, “Wireless networks for mobile edge computing: Spatial modelling and latency analysis,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5225–5240, 2018.

[31] R. Sathyasheelan, “A survey on cloud computing for information storing,” *Journal of Computational Science and Intelligent Technologies*, vol. 1, no. 2, pp. 9–14, 2020.

[32] A. N. Suresh, “A hybrid genetic-neuro algorithm for cloud intrusion detection system,” *Journal of Computational Science and Intelligent Technologies*, vol. 1, no. 2, pp. 15–25, 2020.

[33] P. Sushmitha, "Face recognition framework based on convolution neural network with modified long short term memory method," *Journal of Computational Science and Intelligent Technologies*, vol. 1, no. 3, pp. 22–28, 2020. https://doi.org/10.53409/mnaa.jcsit20201304.

[34] M. B. Sudhan, T. Anitha, M. Aruna, G. C. P. Latha, A. Vijay *et al.*, "Weather forecasting and prediction using hybrid C5.0 machine learning algorithm," *International Journal of Communication Systems*, vol. 34, no. 10, pp. e4805, 2021. https://doi.org/10.1002/dac.4805.

[35] R. Khilar, K. Mariyappan, M. S. Christo, J. Amutharaj, T. Anitha *et al.*, "Artificial intelligence-based security protocols to resist attacks in internet of things," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1440538, pp. 1–10, 2022. https://doi.org/10.1155/2022/1440538.