Mobility Aware Computation Offloading Model for Edge Computing

Natnael Tefera[1(✉)] and Ayalew Belay Habtie[2(✉)]

[1] Department of Computer Science, Arba Minch University, Arba Minch, Ethiopia
natnael3872@gmail.com
[2] Department of Computer Science, Addis Ababa University, Addis Ababa, Ethiopia
Ayalew.Belay@aau.edu.et

Abstract. The technological advancement of mobile devices such as laptops, smartphones, wearable devices, and other handheld devices has resulted in the emergence of various user applications in the entertainment, learning, social networking, and community computing sectors. However, these devices have limited capacity and battery charge to process computation intensive tasks. As a result, offloading is one of the most important approaches for connecting mobile devices and powerful systems. It minimizes complexity and improves mobile computing capacity. Computation on cloud computing is a powerful solution for computation of tasks on devices with limited computing capacity. Still, due to the distance issue the energy usage for transmission and the network delay to send and receive computing requirement degrades the performance. As the result, edge computing is the promising enabler for latency sensitive and energy efficient computation in proximity. However, user mobility and the restricted coverage of Edge Computing (EC) server service pose new challenge for computation offloading. Delivering task offloading requests to servers and results to users is difficult in networks where user movement is frequent, resulting in increased latency, higher energy consumption, and inefficient resource utilization. The key problem in offloading computing to edge server is, determining how to efficiently decide to assign computation tasks to edge servers in such a way that the decision captures the mobility inherent in mobile devices and results in minimal latency, energy consumption and execution time during application running. In this paper, to tackle the constraints related to intermittent connectivity due to mobility, network changes, device heterogeneity, and resource load mobility aware computation offloading model is proposed. Dependency graph-based mobility prediction is adopted to trace next mobility locations and edge servers. Dependency graph with fuzzy logic algorithm is proposed which considers available computation and communication resources both on device and server. This fuzzy logic decision considers edge device battery level, data size, bandwidth, network coverage, delay sensitivity, edge server Virtual Machine (VM) utilization and load of the server. The proposed model is implemented using PureEdgeSim simulator with mobility traces. The simulation result was analyzed with respect to task failure rate, failure rate due to mobility, energy utilization and average task processing delay. The analysis of the simulation result is done using a comparative analysis with the state of the art works fuzzy decision-based cloud-MEC collaborative task offloading management system (FTOM) and a fuzzy decision tree-based task orchestration (FDT) considering task failure rate,

K. Doug et al. (Eds.): SMC 2022, CCIS 1690, pp. 54–71, 2022.
https://doi.org/10.1007/978-3-031-23606-8_4

energy utilization, task processing latency, and task completion delay. The proposed model performs better with minimum task failure rate, energy consumption and task processing delay compared to FTOM and FDT with an increase of mobile devices. For example, with an increase of mobile devices from 100 to 700, task failure rate increase from 2% to 70% in FTOM, 1% to 45% in FDT and from 0.9% to 40% in the proposed model. The simulation result also affirm that the proposed mobility aware computation offloading provided improved task failure rate with mobility when compared to FTOM, which is the worst one. The energy utilization with an increase of mobile devices and the task processing delay time (0.45 s for 700 mobile devices, 0.73 s for FTOM) for the proposed model is better than the other task offloading schemes.

Keywords: Edge computing · Mobility prediction · Computation offloading · Fuzzy logic-based computation offloading

1 Introduction

Dramatic increase and advancement in technology bring the emergence of Internet of Things (IoT), connected physical objects to collect and exchange information, which can greatly improve many aspects of our daily lives [1]. These mobile gadgets have become smarter with the introduction of new processing cores and memory architectures, but they are still limited to computing and residual battery capacity. Many computationally intensive applications, such as 3D modeling, augmented or virtual reality, online games, ultra-HD image and video processing, artificial intelligence, and the IoTs, are resource expensive and create massive amounts of data [2–4]. Such applications may generate a significant amount of computing workload, potentially draining the battery in smart/mobile devices [2]. High computational tasks can be offloaded to the nearby cloud and edge servers that have adequate computing resources [4].

The principle of computation offloading or cyber-foraging was originally introduced in Mobile Cloud Computing (MCC) [5, 6]. This principle is used to leverage powerful infrastructures (*e.g.,* remote servers) to augment the computing capability of less powerful devices (*e.g.,* mobile devices) [7] since, computation offloading allows low-resource devices (*e.g.*, smartphones) to run CPU-intensive applications like 3D gaming [8]. Computational offloading involves the delegation of computationally heavy tasks from mobile devices into central cloud data centers. While competing for computational offloading, mobile devices confront significant challenges. Offloading the computation task provides the possibility of supporting user equipment with ultra-low latency requirement, prolonging the device battery lives [9]. However, sometimes, time and resources saved by mobile devices are offset by the latency involved in locating an appropriate cloud server and migrating the workload. Several real-time applications, such as financial transactions, online gaming, video conferencing require high quality of service and low latency. Failing to provide desired result adversely affects the user experience [10].

Researchers introduced the concept of cloudlets to reduce network delays and increase user experience when compared to MCC. A cloudlet is a small-scale data center that is located near the users [11, 12]. To execute computing and offloading, a few fixed servers are connected via a wireless network in a micro cellular zone [13].

Various studies [20–22] have been conducted which cover the different computation offloading approaches, frameworks, schemes, methods, algorithms, strategies on Edge computing. Most of these studies considered the edge devices as static where user positions are considered to be time-invariant and user mobility-related information is not fully exploited. Some studies including [16–18] has considered mobility. In others study like [17] user mobility traces are considered as historical records of the mobile users at different geographic location. Still some literature considered user mobility using machine learning approach [18].

These mobility prediction approaches have limitation on the erroneous mobility record removal, leaning complexity and mobility feature learning as, mobility of the edge device is linear in machine learning and the mobility in reality is nonlinear due to random mobility of edge devices [26]. The studies lacks to address purely the mobility aware computation offloading in edge computing with noise tolerant and lightweight mobility prediction. This study considers fuzzy logic decisions on computation offloading in combination with graph dependency-based mobility prediction on edge computing for lightweight computation decision and noise tolerant mobility prediction.

The reminder of this paper is structured as follows. The related work on computational offloading is discussed in Sect. 2. The proposed mobility aware computation offloading is presented in Sect. 3. Section 4 discuses experimentation and result analysis. Section 5 depicts the conclusion and future perspectives.

2 Related Work

Computation offloading and resource allocation is an important issue in edge computing. Based on previous literatures, different computation offloading approaches were discussed including fuzzy logic based, machine learning, with mathematical modeling approaches and so on. For example, Hossain *et al.* [14] introduced a fuzzy decision-based cloud-MEC collaborative task offloading management system named as FTOM which is a collaborative task offloading management system based on fuzzy decision-based for using the remote cloud computing capabilities and utilizing neighboring edge servers. Selecting the best target node for offloading based on server capacity, latency sensitivity, and network state is the goal of the proposed approach. Based on the states of the server utilization, network condition, and delay sensitivity, FTOM makes dynamic decisions for offloading delay-sensitive computations to local or nearby edge servers, and delay-tolerant high resource-intensive computations to a remote cloud server. This approach was simulated and compared with other offloading schemes including local edge offloading (LEO), two-tier edge orchestration-based offloading (TTEO), fuzzy orchestration-based load balancing (FOLB), fuzzy workload orchestration-based task offloading (WOTO), and fuzzy edge-orchestration based collaborative task offloading (FCTO) for three different applications. The results of FTOM Simulation showed significant improvement on, rate of successful executed offloaded tasks and reduced task completion time when compared with LEO, TTEO, FOLB, WOTO and FCTO. Nguyen et al. [15], also introduced flexible computation offloading method in fuzzy based mobile edge orchestrator for IoT applications. The authors proposed three-tier architecture, mobile edge orchestrator, edge devices, edge orchestrator with edge server and cloud

server. The edge orchestrator follows two stages in the proposed work. The first stage is application placement and the second stage is application deployment that places the incoming request either on the edge device, edge server, or neighbor edge server. The second stage deploys the application on either the edge server or the cloud server. The proposed work was simulated using four applications and showed an improved results from other techniques. However, the mobility of the edge device was considered as static and energy utilization was not given attention.

Shi et al. [16], proposed computation offloading based on reinforcement learning for mobility aware edge computing by considering mobility, deadline constraint, and available resource on edge servers. To learn servers' users power allocation policies, a deep deterministic policy gradient is proposed which is a reinforcement learning based approach. The researchers model the user mobility by contact time with user and server following the position distribution with mobility intensity parameters. Reinforcement learning is used to map the situation with actions to maximize the reward and in the meantime, the environment is formulated as a Markov decision process. When resource limitation occurs in MEC servers, the unprocessed tasks will be forwarded to the core network. The experiment was done taking the proposed deep reinforcement learning algorithm and random power allocation algorithm with different connect frequencies. The result of the experiment showed that the proposed deep reinforcement algorithm provides better allocation scheme. However, the proposed approach considers communication and computation cost only for edge servers and in the meantime, deep reinforcement learning introduces high time computation complexity as, space and complexity increase with the increase in number of heterogonous server and number of users present in the server. Moreover, Reinforcement learning models are essentially designed for linear mapping and systems with several input parameters and multiple features but user mobility traces are non-linear in nature.

Maleki *et al.* [17], proposed an offloading approach considering dynamic mobile applications features including mobility and changing specifications. Two forecasting methods, the drift method and the weighted moving average method was used to predict the mobility of device. The experiment was conducted for mobility traces of taxi cabs in San Francisco and the experimental results were compared with three baseline approaches. Results showed that the approach finds close to optimal latency and execution time. However, the researchers consider execution time and latency which is not sufficient as edge devices are constrained by the battery life as well.

Zhan *et al.* [18] Presented Heuristic Mobility Aware Offloading Algorithm (HMAOA) for mobility aware multiuser offloading optimization. HMAOA is used to optimize computation offloading in MEC contexts for fast moving users and maximizes the system utility under the constraints of user mobility, resource limitation, and task latency. The user's next expected base station is forecasted by taking into account not just the computing resources but also the channel fading, noise, interference, and distance from the end user to the base station. The researchers investigated multiuser single base station offloading decision and resource allocation considering mobility to achieve reduced task latency and energy consumption. The authors divide the original optimization problem into numerous local optimization problems and sub problems. The offloading choice is accomplished through the use of non-linear integer programming

(NLIP). To determine the approximate optimal offloading option, the NLIP sub problem is solved using a partial order based heuristic technique. The result of this approach is compared with six baseline algorithms and performs and demonstrated close to an optimal solution. The proposed method is found to improve latency and energy consumption, but its mobility prediction component only offers short-term results. Because of this limitation, tasks required to be offloaded cannot be assured to have optimal services.

Lu *et al.* [19], introduced a propagation based and cluster assisted computation offloading strategy considering mobility and social associations between edge device and edge server. To supply candidate edge servers, clustering is used. Channel conditions and available resources are considered to optimize offloading scheme. KALMAN filter with mobility model of mobile user is used to predict the location and trajectory of users. The clustering technique takes into account the historical connection as well as mobility prediction. The simulation was conducted and simulation results demonstrate that the proposed offloading strategy enhances the data processing capability of power-constrained networks and cut down the computation delay. Traditional mobility prediction techniques only consider a specific user's next expected location just based on its historical mobility patterns. However, the above-mentioned model does not consider new user who has not previous record.

3 Mobility Aware Computation Offloading Model

The design objectives of Mobile Aware Computation offloading model is concerned in the reduction of execution time, energy consumption, and latency. Computational task characteristics are concerning for both delay sensitive and computationally heavy activities. As a result, in the case of delay sensitive tasks, completion time reduction is an appropriate design target, whereas energy consumption minimization is a good design objective in the case of computationally intensive tasks. To develop the model, review of the architectures/models of the previous related literatures was made.

The developed model differs from state-of-the-art works in that, the offloading decision is comprised in edge device layer in the existing works which drains the battery of the edge device. As a result, to improve this limitation, the computation offloading decision is incorporated in the edge server layer in the proposed model. Moreover, the proposed model comprises combined fuzzy logic and dependency graph computation offloading decision, and mobility monitoring module on the edge server layer which is not included in the existing baseline studies. Additionally, local and regional edge server are identified based on the geographic distance from the edge device and included in the proposed model. A mobility aware algorithm for computation offloading is also developed.

3.1 The Proposed Model

The proposed mobility aware computation offloading model as shown in Fig. 1 comprises two major components. Edge device component which is the source of data to be processed either local by the edge device itself or by the corresponding edge server. This component includes the edge devices such as smart phones, laptops and other IoT

devices like wearable devices and sensors. The second component contains the edge server which is responsible for running the tasks requested by the edge device. This component includes the local edge servers, regional edge servers and edge orchestrator.

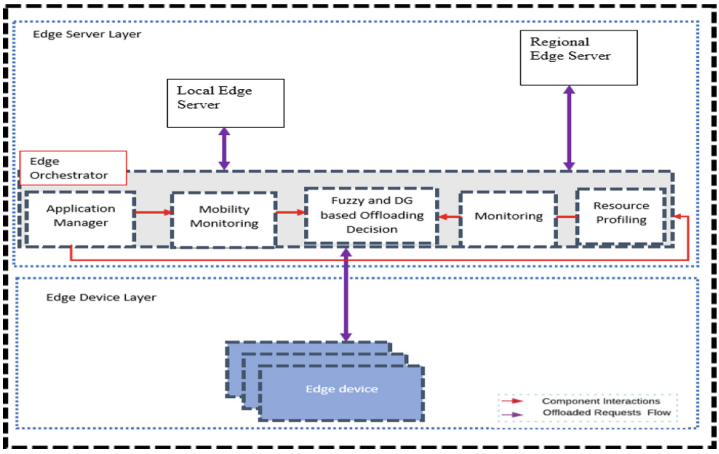

Fig. 1. The proposed mobility aware computation offloading model

Edge Device: The edge devices in the proposed model include IoT devices and edge user equipment, such as smart mobile phones, laptops, and watches. These devices are able to produce data, communicate with the upper layer through the WLAN network. Compared with the Servers in Data Canters, most of the IoT devices at the edge are somewhat limited in both computing capability and battery power. Edge devices can be connected directly to nearby edge servers via wireless links. Each node in the system can directly communicate its resources with the edge orchestrator. Besides, edge devices in our case are considered as heterogeneous in edge computing environment having different capabilities in power, memory size and CPU. Each device forwards offloading requests to the edge orchestrator which decides what to do with it. The resources such as CPU load, task size, remaining battery charge, and current mobility status are the decision-making parameters from the edge device's perspective. For offloading decisions, the volume of data to be transported is also taken into account. When the edge device requests task offloading, device characteristics and the request-related information are collected by the edge orchestrator.

Edge Server: The Edge Server has adequate computing and storage resources as compared to the edge computing and can be used to perform data analysis, and scheduling and computation activities/tasks. An Edge Server contains one or more physical machines hosting several virtual machines, covering the mobile users in proximity. These Edge Servers are interconnected with each other via MAN. The Edge Server is powered in computation and processing capacity in relative to edge devices and IoT gadgets. Edge servers contain the logic of the computation decision making strategy that is the edge

orchestrator that is responsible for handling end user requests for computation offloading decision to reduce resource utilization and energy consumption as well as latency constraints. Based on the proximity of the edge servers to the edge devices, edge server can be local edge server, located near the source of the request initiated and regional edge server little bit distant from the edge device where the request is initiated. However, due to the mobility of the edge device the regional edge server will be the local when the user moves towards the regional edge server. The decision to offload is influenced by server-side aspects such as resources and characteristics. The size of users connected to the server and the number of virtual machines active are the server parameters taken into account.

Edge Orchestrator: The edge orchestrator (EO) is the main decision maker in handling incoming requests from end user devices. EO is a logic which is deployed in the edge servers that helps to reduce the latency as the edge server is powerful in resource in relative to edge devices. Basically, network resources, and the requirements of incoming application tasks, edge device information, mobility status of edge device is managed and controlled by EO and also it maintains a catalogue of the applications that are available. Furthermore, it is also responsible for deciding on whether to offload or not, selection of candidate server in the entire flow of computation based on fuzzy logic approach. Edge orchestrator determines the target computation server of the end user request by monitoring edge device features, edge server resources, the application characteristics, and the available network information factors such as the network bandwidth, edge server utilization, and task characteristics, predicted movement, network latency, and energy consumption to select edge servers.

For the mobility aware computation offloading, a fuzzy decision-based algorithm is proposed for a multiple of reasons. The edge computing environment is dynamic, with resource stages changing on a regular basis based on offload requests. As the quantity of incoming user requests is not known in advance, it's difficult to decide where a job should execute. Furthermore, task offloading management is primarily performed online and is regarded as an NP-hard problem [15, 20]. We require a low-complexity problem-solving technique to deal with these unpredictable environments. Furthermore, many input and output parameters are involved in the edge computing environment, and these parameters are part of the environmental behavior. This approach is inherently fuzzy. In this regard, fuzzy logic is one of the best alternatives for dealing with the aforementioned rapidly changing situation.

The main goal of the fuzzy decision-based computation offloading algorithm is to identify a target server for the offloaded task by monitoring various factors such as the size of the incoming task, the state of the network, and the resources already in use in the servers, energy of the user device and the mobility of the user. Figure 2, shows the fuzzy logic architecture together with dependency graph (DG) based Computation Offloading decision which is used in the proposed model.

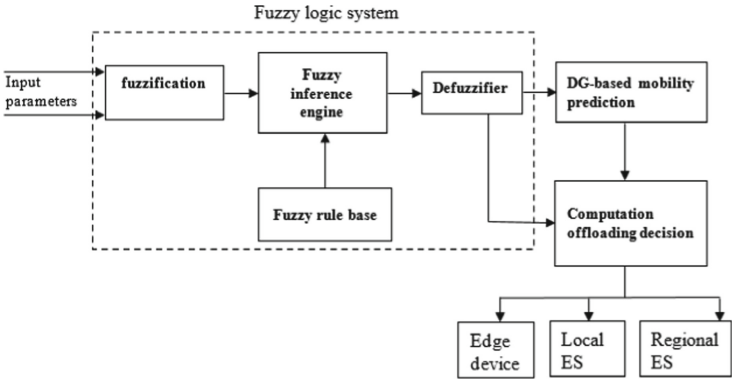

Fig. 2. Mobility aware fuzzy logic architecture with DG-based computation offloading decision

3.2 Mobility Prediction

To know the future station of the mobile edge device, Graph Dependency-based mobility route prediction is adapted from [21] which is noise tolerant while considering individual and collective behavior of mobile edge users. This approach represents paths as a graph, which is then used to accurately match road network arrangement with real-world Edge user movements. This mobility prediction approach is based on the assumption that edge user mobility is order dependent, as end users follow a specified order and Travers road segments in a specific direction to reach some destination of interest.

To forecast the next location of the edge device which is moving, for example as shown in Fig. 3, the current location of an edge device and its historical data are utilized, if available. In this mobility prediction, first, a prediction graph is constructed, in which nodes (vertices) represent road segments and arcs (directed edges) reflect the traversal order of road segments by edge users. The prediction graph is then used to anticipate the edge devices upcoming route by attempting to match its present trajectory with graph paths. However, due to the existence of noise in the graph, graph matching is problematic. In this case, the prediction graph creates graph edges with the following road segments lists appearing within a user-defined look ahead window. This is used to discard noise occurred in the data to increase prediction accuracy. To model collective and individual mobility behaviors, GMG (Global Mobility Graph) and the PMG (Personal Mobility Graph) prediction techniques, are presented to model both global and personal mobility patterns.

Data Preparation: The first edge device location data (GPS records) is collected periodically and sends it to EO. During collection, location data is split into trips by defining stay points. A stay point is a geographic area expressed as a set of consecutive GPS records where the distance from the first and last GPS records exceeds a distance threshold D_{thre} and the driver spent more time than a threshold T_{thr}. The resulting trips are then converted into mobility sequences by map matching GPS trajectories using a cloud map-matching based API.

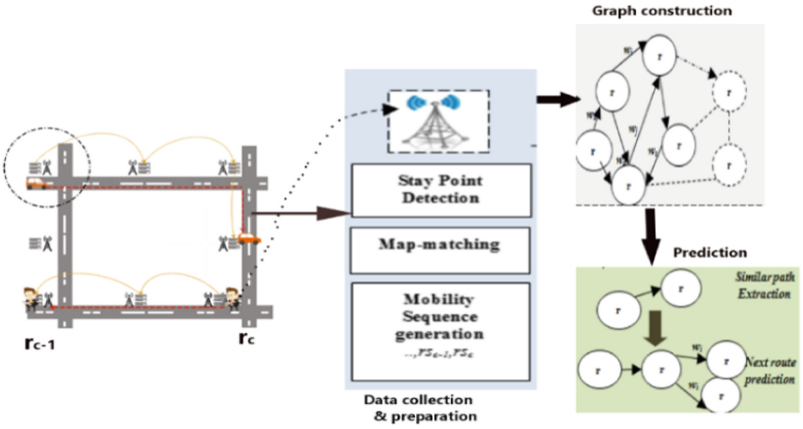

Fig. 3. Mobility aware computation offloading decision

Graph Construction: After obtaining mobility sequences, the EO incrementally updates its mobility graph. The mobility graph is initially built and then extended by inserting new road segments as graph nodes, whereas edge device movements between pair of road segment within the *lookahead* window size are represented by arcs. The weight of each new arc is set to 1. In the case where a road segment appears on newly collected mobility sequences, the weight of the corresponding arc is incremented accordingly.

Prediction: Once a mobility graph has been built, predictions can be performed using it. To predict the next route segment that will be visited by an edge device *D*, its current trajectory, denoted as *CT* (Current Trajectory), is required. It contains the current road segment where *D* is located in addition to its previous locations for the same trip, if available. Formally, let $RS = \{r_1, r_2, \ldots, r_n\}$ be the set of all road segments in a road network. $CT = \{p_j, pj + 1, \ldots, p_c\}$ is a sequence of road segments traversed by *D* where p_c is the current road segment of *D*. Having a trajectory *CT*, the prediction of the next route segment is performed in two steps.

Graph Matching: The first step consists of finding a path $SP = \{s_i, s_i + 1, \ldots, s_m\}$ in the mobility graph that matches with *CT* where $s_i \in RS$. We say that *SP* matches *CT* if and only if each road segment in SP appears in the same order in *CT*, that is $\forall_I$, $s_i = p$ and $m = c$. Note that graph matching is noise sensitive. Finding the path that exactly matches a trajectory *CT* can be challenging since erroneous positions may appear in location data. Using *MG,* built according to *lookahead* window, more flexibility to handle noisy data could be obtained.

For instance, if the first road segment *Ne* that comes after a given node *Nx* in a mobility sequence *s* is considered as noise, another arc will be created that skips *Ne* and go directly to next road segment, given that $w \geq 2$.

Next Road Extraction. The second step is to find the next road segment that edge device user will visit following *SP*, denoted as *Nr*. This road segment is predicted as the destination of the arc having the highest weight emanating from the last road segment in *SP*. More formally, let $E = \{a1, a2, \ldots, an\}$ be the set of outgoing arcs for the last node of $SP(sm)$. Then, *Nr* is defined as: $Nr = Dest(a_k)$ such that $W(a_k) \geq W(a)$ for a $\in E$ and $Uak \in E$ and $source(a_k) = sm$.

The mobility graph is adapted to consider both global (collective) and personal (individual) movement behaviors of drivers.

Global MG (GMG): GMG is used to represent global mobility behavior of a set of persons. The prediction graph is constructed from mobility data of all edge users. GMG is employed to perform predictions for an edge device user when no prior knowledge about his mobility pattern could be found such that edge users newly seen in prediction framework.

Personal Model (PMG): PMG consists of creating a mobility graph for each driver comprising his previous trajectories. Since each PMG only considers a single user, the prediction graph is only trained with his mobility sequences rather than the data of all users. A PMG is a sub-graph of the GMG. Therefore, a PMG can be considerably smaller than a GMG. By default, to forecast the next location of a given user device, the GMG model is used unless matching personal data is found in the PMG of the user. In such situation, the user's PMG is used for prediction.

After all this information is determined the computation offloading activity can be performed. Algorithm 1 represents the mobility aware computation offloading activity.

Algorithm 1: Mobility Aware Computation Offloading Algorithm

Input: *Incoming task T with parameters* $T_{Mobility}$, $T_{Baterry}$, $T_{Network}$, T_{Delay}, $T_{localVM\ utilzation}$

output: *Select target computational resource for offload* O_{Edge}, $O_{Local\ ES}$, $O_{Regional\ ES}$ *with minimum latency, minimum execution time, minimum energy;*

1. *Read profile of incoming task*
2. *T; Read bandwidth;*
3. *Read network coverage;*
4. *Read VM utilization;*
5. *Read mobility speed;*
6. *Fg ← FuzzyLogic(τ, ι, ɗ ,η, ω, e, ɱ); // Output value that fuzzy logic returns*
7. *Calculate a crisp output value X← result of centroid defuzzifier;*
8. *If X<=8 then*
9. *If Required capacity < existing* ***then***
   O= edge device;
   else
10. *Check state of edge device* ***then***
11. *If device moving speed is low* ***then***
    O= Local edge server;

else

12. *If mobility speed is medium and Local edge server network coverage is high* ***then***
    O= Local edge server;
    else
13. *Predict_Next_Location by Graph dependency-based mobility route prediction* ***then***
    O=regional edge server
    else
14. *if X>8* ***then***
15. *check mobility state* ***then***
16. *if not mobile* ***then***
    O=Local edge server;
    else
17. *if mobility speed is low* ***then***
    O= Local edge server;
    else
18. *if mobility speed is medium and Local edge server network coverage high* ***then***
    O=Local edge server;
    else
19. *if mobility speed is medium and Local edge server network coverage meduim* ***then***
    O= Regional edge server;
    else
20. *Predict_Next_Location by Graph dependency-based mobility route prediction;* ***then***
    O= Next_Regional edge server;
    end

Algorithm 1 describes the precise steps for determining the best target server for offloading computation based on fuzzy logic and mobility prediction. Initially, the tasks of edge devices and their characteristics, including task length, necessary number of cycles for task, and deadline requirement to finish task are collected from the edge device collected and from the available edge nearest edge servers (line 1–5). Based on the collected, parameter values are feed into the fuzzy logic system (line 6), then crisp values are calculated based on centroid defuzzification method (line 7). Offloading decision will be conducted based on the crisp output values and hence, if the result is less than 8 and if the device has required capability to compute the request, computation offloading will be handled there (*i.e.*, the edge device itself) line (8–9). However, the required capacity to handle request may not be in the device itself, in this case, the computation offloading will be granted based on the mobility status and network coverage of the available edge server (line 10–13). For crisp values greater than 8 computation offloading will be granted based on the mobility status, mobility prediction graph and network coverage of the available edge sever (line 14–20).

4 Experimentation and Result Analysis

In this section the evaluation of mobility aware computation offloading in edge computing through a simulator is presented in detail. The simulation tool, implementation of the proposed algorithm for the desired model and the parameters of the simulation and the simulation setup and the analysis result is presented.

4.1 Simulation Environment

To evaluate the performance of the proposed mobility aware computation offloading model, the most commonly known edge computing simulator – PureEdgeSim is used. PureEdgeSim is based on CloudSim Plus [22], which is event-driven simulation framework intended to simulate cloud, fog, and edge computing environments. It offers the necessary simulation components such as the network model, the mobility model, and the description of edge device characteristics (i.e. CPU capacity, battery, mobility, storage, and so on), making it usable for the proposed scenarios.

The simulation setup on this simulator comprises of an edge device including IoT devices, and a Local and regional server. The regional edge server is also emulated as a single data centre with one host and eight virtual machines, each with a processing rate of ten thousand MIPS. The number of mobile users is deployed equally among the edge servers. Each location is covered by a dedicated wireless network WLAN between edge servers and WLAN between edge device and edge server. Moreover, to investigate the performance when the system is overloaded, we vary the number of mobile devices from 100 to 1500. When they move to the related location, they will join WLAN, and they based on their offloading decision send tasks to the edge orchestrator for offloading to edge server or local processing or edge processing. The simulation parameters used is shown in Table 1.

Based on the application type, these parameters are random numbers with an exponential distribution. Following the creation of the list, the tasks are sorted by start time,

Table 1. Simulation parameters

Parameter	Measurement	Values
Edge devices range	Meter	15
Wan bandwidth	Mbps	400
WAN propagation delay	Second	0.1
WLAN bandwidth	Mbps	300
WLAN propagation delay	Seconds	0.2
Orchestrator deployment	–	Edge server
Orchestration algorithm	–	Mobility aware fuzzy logic, fuzzy logic, ECOOA
Architecture	–	Edge

and parameters such as the network model between the devices are initiated, and determined based on the average data task sizes, which differ for input and output. A network model is in charge of calculating the queuing delay of WLAN connections between edge devices and the edge orchestrator, as well as WAN connections between edge servers through edge orchestrator, in both uploading (task input) and downloading (task output) directions.

Finally, the virtual machines for the local and regional edge servers are launched, and the simulator's initialization step is completed and starts generating tasks automatically. For devices that are moving, based on the mobility speed the next location and edge server will be predicted based on the graph dependency method. When the simulation begins, the tasks are served depending on their start time and regardless of the application to which they belong. An end device manager is responsible in each IoT device for making the decision of where to process based on the decision algorithm.

4.2 Experiment Result and Analysis

The analysis of the simulation result is done using a comparative analysis with the state of the art works fuzzy decision-based cloud-MEC collaborative task offloading management system (FTOM) [14] and a fuzzy decision tree-based task orchestration (FDT) [23] considering task failure rate, energy utilization, task processing latency, and task completion delay. It is recalled that FTOM was proved to be a better computation offloading scheme compared to previous techniques including LEO, TTEO, FOLB, WOTO and FCTO in relation to success of task failure rate, task processing latency and task completion time.

To measure task completion time of the developed computation offloading model Fig. 4 shows the average task failure rate (the y-axis) versus the number of mobile devices (the x-axis, from 100 to 700). Task failure occur due to mobility, network delay or server capacity which are considered in the experiment. From analyzing Fig. 4. Task failure rate is approximately zero until mobile devices are 100. This is due to the fact that the mobility is low and the system is lightly loaded. However, when the mobility and edge device load increases the situation is changed as congestion is created. For example, the

task failure rate increases from 2% at 100 mobile devices to 80% at 700 mobile devices in FTOM, from 1% at 100 mobile devices to 45% at 700 mobile devices in FDT and from 0.9% at 100 mobile devices to 40% at 700 mobile devices in our proposed model. Comparing all the models, the proposed model provided a lower task failure rather than the others because the proposed model uses mobility for computation offloading.

Further experiment on the task failure rate due to mobility indicated that the failure is increasing with an increase of mobile devices in FTOM and FDT as shown in Fig. 5. When 100 mobile devices are used in all the schemes, FTOM, FDT and the proposed mobility aware offloading computation model, the task failure rate due to mobility is almost similar approximately 10%. As the number of mobile devices increases the task failure rate due to mobility rapidly increases in all models but in FTOM is the worst. The proposed model offload task better that the others with very significant difference, it takes mobility as one of the parameters in task offloading.

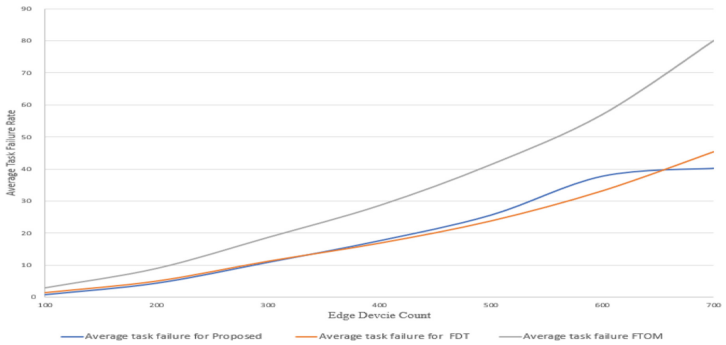

Fig. 4. Task failure rate for the proposed approach

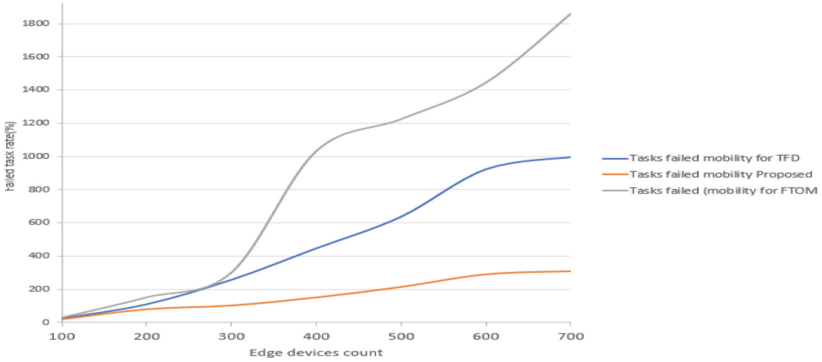

Fig. 5. Failure rate due to device mobility

Energy consumption utilization is one of the most critical parameters for edge devices specifically, for battery powered devices. As the result while delegating computation tasks battery consumption should be considered. In our experiment energy consumption

was one of the parameters for task offloading decision which was not considered in the previous studies. Figure 6, show the battery consumption of the proposed study.

Analyzing Fig. 6, on energy utilization of mobile devices, the energy consumption rate (y-axis) increases with the increase of mobile device count (x-axis, from 100 to 700). For example, at 100 mobile devices all the models FTOM, FDT and the proposed model utilize 0.8% of energy. When the mobile device count is increase from 100 to 250, the energy consumption for FDT is higher that FTOM and the proposed model. From 250 to 420 mobile devices, the proposed model energy utilization is larger when compared to the other schemes. From 420 to 700 mobiles, energy consumption for FTOM is high. Comparing the overall energy consumption, the proposed model energy consumption at 700 mobile devices is 6.9% whereas FDT is 7.2% and FTOM consumption is 7.9%.

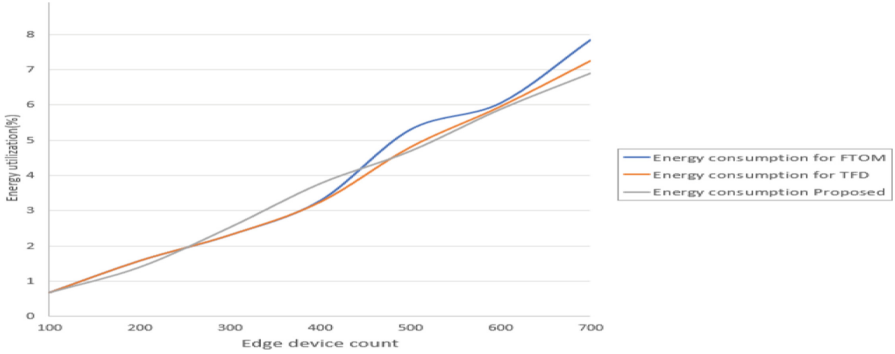

Fig. 6. Energy utilization for battery consuming devices

Another parameter considered in our experiment is task processing delay. It is true that the computation decision is handled considering the sensitivity of tasks and the mobility of edge devices. This gives the proposed approach better constraint from the baseline state of the art works, FTOM and FDT as shown in Fig. 7. Analyzing Fig. 7, when 100 mobile devices are used in the experiment the task processing delay (y-axis) for FDT and FTOM is 0.51 and 0,55 s respectively. But the time delay for the proposed model is 0.2 s. At 700 mobile devices FTOM, FDT and the proposed model processing time delay is 0.73 s, 0,7 s and 0.45 s respectively. Primarily, during computation offloading the task latency is considered as a fuzzy logic parameter and the available resource of the edge servers within the proximity is considered based on the device location and the state of the edge device either mobile or static. This makes the proposed approach to outperform from the other studies. Even though, the increase in edge device affects the task processing latency, as resources in edge servers are also limited, the available resource utilization during offloading decision lowers the task processing delay from other studies.

In relation to this, task completion time which is the summation of processing time and network delay was also considered in the experiment. With the increase in numbers of mobile devices, the overall, the average task completion time was increased. However, as the proposed model decision considers mobility and tasks size to balance the utilization

of resources on the regional and local edge servers, the proposed approach showed better performance on average.

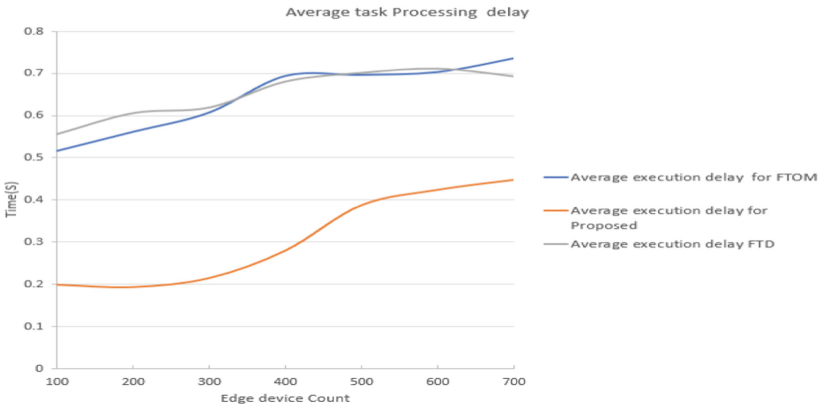

Fig. 7. Average task processing delay

5 Conclusion

In this study, we proposed mobility aware computation offloading model with fuzzy logic-based edge orchestration for edge computing applications, which controls computing resources to improve performance and the energy consumption. The EO selects where to compute the incoming client requests based on the available information on network connections and the status of the edge server and the application features. A fuzzy logic-based workload orchestrator is proposed in our system to give an efficient offload decision: a mobile device, a local edge server, or a regional edge server to assign edge resources to the incoming user requests.

Computation offloading to the nearest edge servers augments the capabilities of mobile devices and ensures lower-latency services and energy efficiency. However, a decision on computation offloading concerning uncertainties such as user highly scalable. Moreover, they reduce the number of migrations significantly compared to other non-optimal approaches.

The simulation results were analyzed with metrics, average task failure rate, and task failure rate due to mobility, average latency and average execution delay with state-of-the-art studies. Experimental evaluations showed that our proposed computation offloading model find energy efficient, latency reduction, and execution time minimization considering the mobility of the edge device in a reasonable time.

For future work, we plan to investigate the real data traces and propose efficient mobility-aware and energy efficient offloading methods while guaranteeing mobility of the edge device.

References

1. Chen, Y., Zhang, N., Zhang, Y., Chen, X.: Dynamic computation offloading in edge computing for internet of things, pp. 2327–4662. IEEE (2018)
2. Wang, Z., Zhao, Z., Min, G., Huang, X., Ni, Q., Wange, R.: User mobility aware task assignment for mobile edge computing. Future Gener. Comput. Syst. **85**, 1–8 (2018)
3. Zaman, S.K., Maqsood, T., Ali, M., Bilal, K.: A load balanced task scheduling heuristic for large-scale computing systems. Comput. Syst. Sci. Eng. **34**(2), 79–90 (2019)
4. Jehangiri, A.I., et al.: Mobility-aware computational offloading in mobile edge networks: a survey. Cluster Comput. **24**, 2735–2756 (2021)
5. Shuja, J., et al.: Towards native code offloading based MCC frameworks for multimedia applications: a survey. J. Netw. Comput. Appl. **75**, 335–354 (2016)
6. Yu, W., et al.: A survey on the edge computing for the internet of things. Mob. Edge Comput. **6**, 6900–6919 (2017)
7. Lin, L., Liao, X., Jin, H., Li, P.: Computation offloading toward edge computing. Proc. IEEE **107**, 1584–1607 (2019)
8. Messaoudi, F., Ksentini, A., Bertin, P.: On using edge computing for computation offloading in mobile network. IEEE (2017)
9. Guo, F., Zhang, H., Ji, H., Li, X., Leung, V.C.M.: An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. IEEE/ACM Trans. Netw. **26**, 2651–2664 (2018)
10. Sheng, J., Hu, J., Teng, X., Wang, B., Pan, X.: Computation offloading strategy in mobile edge computing. Information **10**, 191 (2019)
11. Zhao, Z., Liu, F., Cai, Z., Xiao, N.: Edge computing: platforms, applications and challenges. J. Comput. Res. Dev. **55**, 327–337 (2018)
12. Sun, X., Ansari, N.: Latency aware workload offloading in the cloudlet network. IEEE Commun. Lett. **21**(7), 1481–1484 (2017)
13. Shi, W., Sun, H., Cao, J., Zhang, Q., Liu, W.: Edge computing-an emerging computing model for the internet of everything era. Comput. Res. Dev. **54**, 907–924 (2017)
14. Sonmez, C., Ozgovde, A., Ersoy, C.: Fuzzy workload orchestration for edge computing. IEEE Trans. Netw. Serv. Manag. **16**(2), 769–782 (2019)
15. Hossain, M., et al.: Fuzzy decision-based efficient task offloading management scheme in multi-tier MEC-enabled networks. Sensors **21**(4), 1484 (2021). https://doi.org/10.3390/s21041484
16. Spatharakis, D., et al.: A scalable edge computing architecture enabling smart offloading for location based services. Pervasive Mob. Comput. **67** (2020)
17. Lu, Y., Chen, Z., Gao, Q., Jing, T., Qian, J.: A mobility- aware and sociality-associate computation offloading strategy for IoT. Wirel. Commun. Mob. Comput. **12** (2021)
18. Maleki, E.F.: Mobility-aware computation offloading in edge computing using prediction. In: IEEE 4th International Conference on Fog and Edge Computing (ICFEC) (2020)
19. Nguyen, V.D., Khanh, T.T., Nguyen, T.D.T., Hong, C.S., Huh, E.-N.: Flexible computation offloading in a fuzzy-based mobile edge orchestrator for IoT applications. J. Cloud Comput. **9**(1), 1–18 (2020). https://doi.org/10.1186/s13677-020-00211-9
20. Shi, M., Wang, R., Liu, E., Xu, Z., Wang, L.: Deep reinforcement learning based computation offloading for mobility-aware edge computing. In: Gao, H., Feng, Z., Yu, J., Wu, J. (eds.) ChinaCom 2019. LNICSSITE, vol. 312, pp. 53–65. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-41114-5_5
21. Zhan, W., Luo, C., Min, G., Wang, C., Zhu, Q., Duan, H.: Mobility-aware multi-user offloading optimization for mobile edge computing. IEEE Trans. Veh. Technol. **69**(3), 3341–3356 (2020)

22. Viger, P.F., Ouinten, Y., Lagraa, N., Amirat, H.: MyRoute: a graph-dependency based model for real-time route prediction. J. Commun. **12**(12), 668–676 (2017)
23. Silva Filho, M.C., Oliveira, R.L., Monteiro, C.C., Inácio, P.R., Freire, M.M.: CloudSim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity. Extensibility and correctness. In: IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (2017)
24. Mechalikh, C., Taktak, H., Moussa, F.: A fuzzy decision tree based tasks orchestration algorithm for edge computing environments. In: Barolli, L., Amato, F., Moscato, F., Enokido, T., Takizawa, M. (eds.) AINA 2020. AISC, vol. 1151, pp. 193–203. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44041-1_18