QOS DRIVEN OPTIMAL MOBILE EDGE SERVER PLACEMENT INMOBILE EDGE CLOUD

A Thesis

by

PEIDONG SUN

BS, Lanzhou University, 2016

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

May 2020

QOS DRIVEN OPTIMAL MOBILE EDGE SERVER PLACEMENT INMOBILE EDGE CLOUD

A Thesis

by

PEIDONG SUN

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

May 2020

ABSTRACT

Mobile edge cloud is an emerging technology to enhance the Quality of Service (QoS) for mobile users' applications, especially for computation resource-consuming applications. A challenge in mobile edge cloud is the problem of mobile edge server placement, which concerns where to place the mobile edge servers to reduce the transmission delay and computation delay for tasks generated by mobile device users. In this essay, we work on the deployment of edge servers in the mobile edge cloud. We formulate the deployment problem as an integer linear problem and propose a density-based deployment of the MECs algorithm, which combines the K-means approach and integer linear programming. To evaluate the performance of our proposed approach, we conduct experiments using the telecom dataset of Shanghai. From the result of the experiment, we demonstrate that our proposed method could reduce the delay per task by around 10%.

Index Terms: MEC server placement, edge computing, quality of service, service scaling

DEDICATION

I would like to thank the following vital people who have supported me through the Master's program. Firstly, Let me express gratitude to thesis supervisor Dr. Zhang for his insightful guidance and help throughout this research project as well as the Master's program. I am grateful to the thesis committee members Dr. Li and Dr. Garcia, for their academic advice during this research project. Moreover, I would like to thank my parents and all my close friends. You have all encouraged and support me throughout this research project and my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Motivation

Within the last twenty years, cloud computing [1], Internet of things[2], and augmented reality/virtual reality[3] have got lots of attention from not only academic researchers but also industrial companies. Cloud computing is a rising industrial computing architecture for allowing remote on-demand access to central joint servers of computing resources [4] (e.g., computation resources, storage resources, software, and services). With the growth of attention in the cloud computing field, there are four cloud computing product companies: Google Cloud Platform, Amazon Web Services, Microsoft Azure, Alibaba Cloud. In the meantime, mobile devices have been rapid growth. According to Cisco's approximation, by 2021, there will be around 12 billion mobile devices, which consist of phones, pads, smartwatches, etc. [5]. With the widespread use of mobile devices, more and more applications, as illustrated by face recognition, natural language understanding, video stream processing, and interactive gaming, catch more and more attention. These kinds of applications required intensive computation resources, high storage volume, high energy consumption [6]. Only for interactive gaming (Augmented Reality and Virtual Reality) mark is forecast to be nearly $20.4 billion in 2019, a rise of 68.8% over the $12.1 billion [7]. Worldwide outlay on AR/VR merchandise and services can continue this increase throughout the 2017-2022 forecast amount, reaching a five-year compound annual rate of 69.6% [8]. Nevertheless, mobile devices have limited storage and battery capacity, especially computation resources. Motivated by these trends, mobile edge computing (MEC) has been proposed to solve the limited storage and battery capacity problems.

Contrary to the mobile centralized cloud, The mobile edge cloud (MEC) is located closer to mobile devices and could provide ample storage capacity and computational resources to mobile devices [9, 10]. MEC is an addition of the centralized cloud in the edge networks. Mobile edge servers could bring powerful computational resources and storage resources to end personal users. The goal of MEC is to provide abundant computation resources and storage resources to high resource-consuming mobile applications and to improve user's experience for using devices[9].

In the mobile edge computing filed, most current research works focus on the offloading strategy, which is to determine which MEC server the mobile users' tasks should be offload to in mobile edge computing, presuming that the MEC server has been deployed. There are limited studies that pay attention to the placement of MEC servers in the MEC filed. There have been some researches works on the placement of the mobile edge cloud server in recent years. The edge cloud is one server group with computation and storage resources, base stations that could provide wireless access to mobile edge servers in a cellular network. Mobile users can offload their complex tasks to the mobile edge cloud for getting services[11]. In the edge cloud, the base stations are already placed in a specific location. Thus, in this essay, we only consider the mobile edge server placement issue. Although some approaches finished by other researchers are efficient, they do not consider the computation delay for tasks generated by mobile users. In this essay, we consider the transmission delay, which represents sending the task to the server and the computation delay for the task processed in the mobile edge server.

In this thesis, we study the placement for a scalable edge cloud server problem that is to deploy the mobile edge cloud server in the competitor places and assign the base stations and the mobile users within the service range of base stations into different edge clouds. The candidate locations for mobile edge cloud servers could be the same as some base station locations. Contrasted with the present research works, we emphasis on the minimum of the transmission delay and the computation delay for scalable mobile edge servers. Form there, the deployment of edge server problem is formulated as an integer linear problem. Moreover, we indicate that this integer linear problem is an NP-hard problem. To find the optimal solution to the deployment of the mobile edge server problem, we propose the density-based MEC server placement algorithm (DBMECsPA), which is an approximate edge cloud placement approach. To evaluate the performance of our approach, we finish experiments based on base station data set from shanghai telecom[5, 12, 13]. Compared with other research works, our approach could increase the number of validated tasks and reduce communication and computation delay.

Purpose

In this thesis, we propose the density-based mobile edge cloud server deployment algorithm and try to enhance the quality of service (QoS) for the end-personal users. So as to improve the quality of service (QoS), We focus on not only the transmission delay but also the computation delay for the task. The transmission delay consists 1) the transportation latency for transmitting mobile users' tasks to the assigned base station; 2) the delivery delay for uploading mobile users' task from the serving base station to assigned MEC servers; 3) the sending delay for uploading the tasks to the core server; 4) the communication delay for sending the requested task from the core server to mobile users. The computation delay consists of 1) the processing time for the task in the MEC server; 2) the processing time for the task in the core server.

Scope and limitation

Due to the fixed architecture and environment, there are several limitations of this research:

- This thesis mainly focuses on mobile phone users instead of wearable devices, pads, connected cars.
- The complexity of tasks will be limited to action recognition tasks, but the size of the requests will vary from 150KB to 1.5 MB.

- We consider the average computation delay for five times simulations

Target group

The target group for this thesis is the application company and telecommunications company. First, this work is focusing on improving the quality of services(QoS) of applications such as face recognition, action recognition, natural language processing and understanding, video stream processing, and interactive gaming. Thus, application companies could provide better services to customers such as YouTube, Facebook, Pokemon Go, etc. Second, the MEC server needs to be placed with the base station. Thus, telecommunications companies could place the MEC servers in optimal places such as ATT, T-mobile, etc.

Outline

The rest of the paper is structured accordingly. Section II sates related work on placement of the MEC server. Section III introduces several cluster methods and related concepts. Section IV explains the proposed system architecture, system model, formulates the placement problem of the MEC server. The proposed algorithm and analysis are also in section IV. The simulated results are presented in Section V. Finally, we conclude this thesis in Section VI.

LITERATURE REVIEW

In this section, we are going to talk about some related work with MEC server placement. There are several different ways to improve users' experience by reducing the transmission time or computation time or both of the two methods we mentioned above.

Improving the quality of services (QoS) is a vital issue in MEC's architecture that has received widespread notification. QoS consists of transmission delays between mobile users and MEC servers on the core server and delay processing on MEC servers and core servers[14]. QoS consists of a mobile edge cloud server location that affects not only the communication delay from personal end users to the allocated mobile edge cloud server but also the computation delay on the MEC servers and core server. All the above reasons make the location of MEC servers a crucial problem for enhancing service efficiency.

Qazi et al. show that the issue of MEC server placement has a significant effect on QoS and operating costs because of the number of MEC servers and MEC servicing sites. However, the placement issue was not addressed. They suggested a MOP-based allocation orchestration for the placement of mobile edge cloud problems [15]. One branch in the placement of MEC focuses on reducing transportation time between mobile users and MEC servers for the tasks generated by mobile users. However, the computation delay, which is a crucial essential for low latency services, does not measure these studies. Besides, MEC servers are considered to be abundant in computational resources in a single location in some studies, whereas MEC servers are distributed and have limited computational resources. Therefore, these solutions are not suitable for our problem. Guoyu Yang and others suggest that the connection status of the access points can not work efficiently due to the congested links. Thus, they recommend a form of access point classification to clarify the connection status, such as excellent status, poor connection status of the access points, and to deploy MEC servers in ranks [16]. Qiang Fan et al. put their effort into balance for transmitting and delivery expenses. The number of servers is noted as a critical factor impacting production costs. They then implement a strategic position algorithm, which decreases the number of deployed edge servers and improve the quality of services of tasks.[17]. In the meantime, there are some researches mentioned the cloudlets which could provide computational and storage resources, internet access. In this essay, although the MEC server needs internet access to provide services to mobile users, we assume the capability of the cloudlet and the MEC server are the same. Thus, in the rest of the essay, we consider the cloudlet and MEC server could be replaced by each other. Zichuan Xu et al. claims that a wireless network with cloudlets is having a crisis. They intend to install several servers with specific computing capability in sensitive positions to reduce device latency. The problems show NP-hard, and a robust heuristic approach has been suggested[18]. A workload balancing method to deployed cloudlets was introduced by Mike Jia and others to lessen the average response delay for cloud-setting problems through wireless metropolitan networks (WMAN). They prove that the problem is NP-hard and suggest a scalable heuristic approach[19]. Our proposed method mainly differs on these essential aspects: first, the method we proposed is scalable, which vary from 3 to 15. most of all, we place different MEC servers based on our proposed algorithms. The location of base stations are known, and the placement of the MEC server is not limited to base station location set.

THEORETICAL BACKGROUND

This section gives an overview of the related technical support. In improving the quality of service by using mobile edge servers, the following questions are answered:

- Is cloud computing or mobile cloud computing could help mobile phones to save energy and improve energy efficiency?
- What are the advantages of mobile edge server compared to a centralized server?
- Is there any algorithm that could be used in the deployment of the MEC server?

To answer these questions, the first section deals with Cloud computing, followed by a definition of mobile edge computing; mobile centralized computing. Afterwards, a detailed illustration of machine learning in general and relevant algorithms. Further, we will introduce several kinds of cluster methods, and quality evaluation is presented.

Cloud computing

Amazon.com proposed the term" cloud computing" in 2006 with releasing its Flexible Compute Cloud products. The could computing could provide computation resources and storage resources to the user through an internet connection.[20]. Cloud computing combines centralized or distributed computer hardware and software that could be accessed by users via the Internet. Cloud computing contains computing, storage resources, and database management applications that are not located in the user's location. Service providers own and manage all these resources, and users could access the resources remotely[21]. The benefit of cloud computing:

- Users could back up their essential information to protect the robust of the system and reduce the chance of losing data.
- Users could also execute the application in the cloud instead of the local computer to improve teamwork efficiency.

- Cloud companies have flexible plans to satisfy customers' requirements and reduce the cost for customers.

With these benefits, the value of the Cloud computing market is becoming larger and larger. Cloud computing is offered to these industry file electricity, water, gas, etc. There are several cloud computing companies, such as Alibaba's computing cloud, Microsoft's Azure platform, Amazon cloud, Google cloud platform.

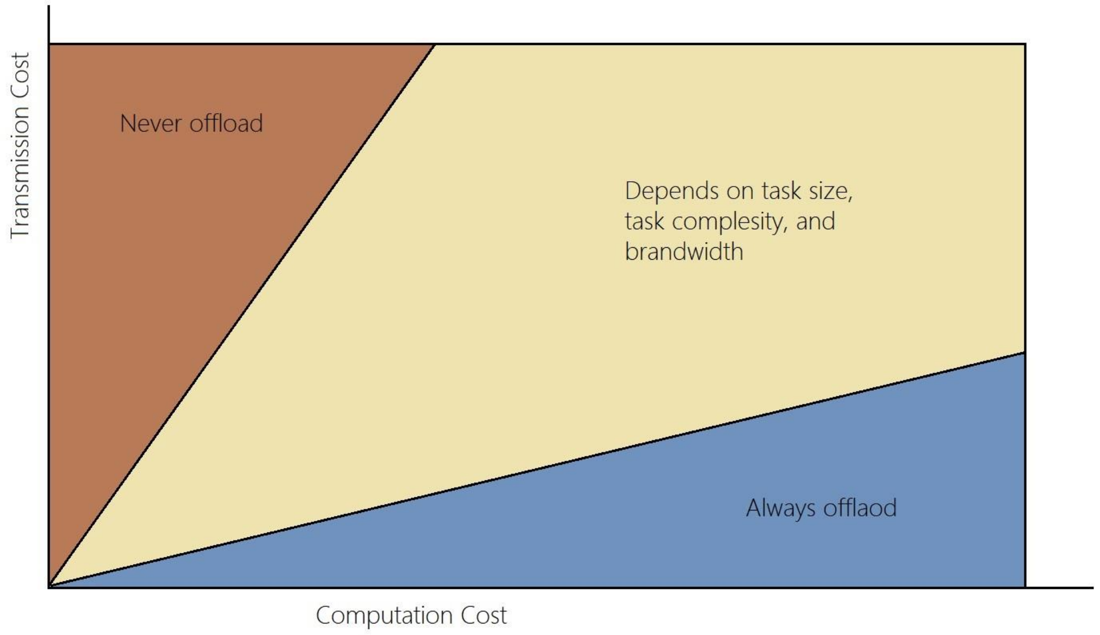


Figure 1: Features that affect task offload decision

Mobile centralized cloud

Mobile centralized could is part of cloud computing. Compared with mobile devices, Mobile cloud computing could provide computation and storage resources to the users of mobile devices through the Internet [22, 23]. We can get some advantages from Mobile Cloud Computing:

- Increasing mobile device battery lifetime. The battery is one of the main limitations of all different mobile devices. Several solutions have been proposed to enhance CPU performance and manage the resources of the mobile device effectively to decrease power consumption.

However, some of these solutions need extra hardware or modify the architecture of mobile devices, which are inconvenient for the designers of mobile devices and increase the cost of investigating mobile phones, which modifies the placement of battery module, camera module, motherboard module in the limited phone space. The offloading system is proposed to move the computation resource-consuming tasks and electricity-consuming tasks to the mobile cloud server, which could avoid the high consumption of electricity in local mobile devices. For example, the mobile device could reduce around 40% for energy consumption by offloading the image processing task to the mobile cloud [24].

- Improving data storage capability and provide computation resources. The storage and computation capabilities are limitations for IoT devices. The IoT devices would gather and generate massive data during execution time. Due to the limited storage capability of IoT devices, the data will be uploaded to the mobile cloud. The mobile cloud also provides computation resources to IoT devices.

In MCC, the user's mobile devices could get computation and storage resources of a centralized cloud. This method has some advantages: extend battery life by offload compute process to a centralized cloud and provide big volume storage for the mobile device, but this method also has some issues: long latency. Based on the disadvantages of MCC, Mobile edge computing(MEC) is proposed, which is an accessible cloud system that could provide services to end-personal-user devices and locates at the mobile edge networks. Mobile Edge Computing could provide abundant storage resources and computation resources for real-time control, computation [25, 26].

Mobile edge cloud

Mobile edge cloud(MEC) is an accessible cloud system that could provide resources and services to end-personal-user devices and locates closer to the mobile user than MCC. The mobile edge cloud is deployed close to mobile users. Normally, the MEC platform consists of base stations, different types of mobile edge servers. There are several cases that MEC could enhance the services:

- Automobile industry. There are lots of connected vehicles that appear in the automobile industry. The connected vehicles would collect massive data to ensure the security of the driver and passengers. Low latency is required by connected vehicles. MEC cloud store and process the data with lower latency than MCC since the edge servers are closer to base stations and cars.
- Augmented reality. Pokemon Go and Google SkyMap are examples of augmented reality applications in the real world. Pokemon Go uses the phone's camera to capture the surrounding environment, and the captured environment and the game information would pop up in the user's screen. The game information includes different Pokemon based on the captured environment, game time based on the user's locations. Augmented reality applications such as Pokemon Go need to extract information form environments such as text, object, and location information, which requires image processing methods and other classification technologies. MEC could store and process the image in real-time to enhance gaming experiences.

Aspect	MCC	MEC
Deployment	Centralized	Distributed
Distance between server and user	Long	Short
Transmission delay	Giant	Light
Computational resource	Abundant	Limited
Storage capability	Sufficient	Inadequate

Table 1: Comparison between MEC and MCC

There are three conditions to help personal end-user to decide should the task offloading on MEC[11]:

- Offloading helps decreases energy consumption and increases execution speed.[27].
- Minimize the processing delay and keep the workload balance for computation resources and connection links[28].
- Improve the service connection if the personal end-user using the MEC servers through the network[29].

Unsupervised learning

Unsupervised learning is a type of machine learning approach used to find finding hidden relationships in the unlabeled data set. Unsupervised learning divides the data into several different groups based on criteria. Unsupervised learning is to find the hidden patterns between metadata. distance[30].There are four popular methods that we are going to introduce as follows: Hierarchical clustering is to build a cluster tree to achieve a multilevel hierarchy clusters[31]. K-Means clustering select k random centroids initially. K-means will find the optimal cluster by repeating the following steps: 1) calculate the distance from the point to all centroids; 2) select the smallest distance; 3) update the centroid by taking the average of all points belongs to the same cluster. K-means would terminate because the execution time reaches the maximum iteration, or there are no changes that happened for the cluster. The complexity of k-means is $O(n^2)$ [32]. Self-organizing maps try to figure out the topology of the data set and distribution of the data with neural networks[33]. Hidden Markov models are used to describe the probability of the event, which will happen based on independent observation. Hidden Markov models are another promising method to solve our problem, which for all tasks generated by the mobile users are independent, and all the edge servers are independent[34].

SYSTEM MODEL, PROBLEM FORMULATION AND PROPOSED ALGORITHM

In this section, we will introduce the network model and the algorithm we proposed in this thesis, problem statements, and formulation.

System model architecture

In our system, figure 4.1, MEC servers are placed in different locations of cellular networks. In our proposed method, the Phones, Pads, computers, and other smart devices get services from the Internet via the MEC server. We assume all the tasks are a computation-intensive task that needs lots of computation resources to finish. MEC server also connects to core internet, which could provide more computation and storage resources for the MEC server. Mobile users get services from different services providers by sending tasks to the assigned base station; then, the assigned base station offloads users' tasks to the mobile edge cloud servers. Note that one base station is only held for a mobile edge cloud, and one mobile edge server could consist of more than one base station. Two problems need to be solved in the placement of edge serve problems. One is to reduce the transmission delay from the user to the MEC server and the core server, and the other one is to decrease the computation delay for users' tasks in MEC servers and the core servers[35].

- The mobile user would send a requirement to the MEC server via the base station.
- The MEC server receives the request and finishes the pre-processing task and sends the remaining task to the core server due to limited computation resources.
- The core server would finish processing the tasks and send the result back to the mobile user.

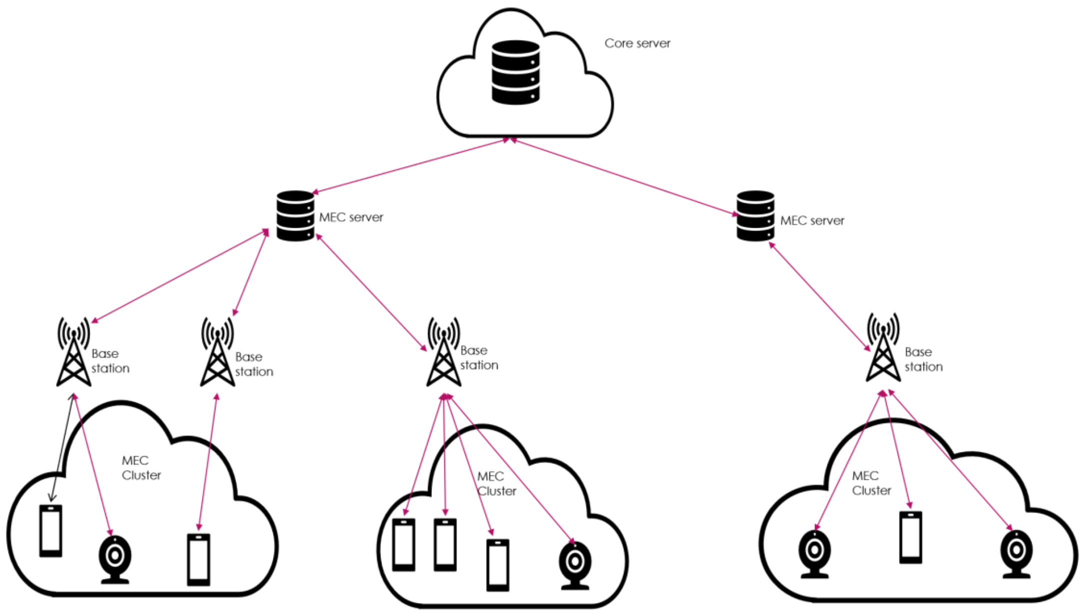


Figure 2:Model of edge cloud placement

Network model architecture

The mobile edge cloud system could be modelled by a Graph G(M ∪ B, L), where M is the set of candidate locations for mobile edge cloud servers, and the number for M is k. L is a set of the link between MEC server and base stations. In addition, each edge cloud could consist of more than one base station.

Each $l \in L$ has the following property:

- Transmission delay: D_l^t is the time it takes for a task *t* be uploaded via network link *l*.
- Connection status: $\rho^{l}_{(v,m)}$ denotes the link between the base station *v* and MEC server *m*.

Task model architecture

Let T be a set of tasks that mobile user-generated. We consider that one or more tasks are deployed in the network, and all of them are independent. Each task $t \in T$ has the following properties:

- Maximum Tolerance Delay: MTT_t is the maximum time allowed for responding a task t. If the network delay plus the computation delay for task t exceed MTT_t, this task would be a false return result.
- Task size: st_t is the volume of the task t generated by a mobile user.

Server model architecture

In this essay, we assume the size of M is k, which means we need k mobile edge server to deploy in the experiment area, and capacity each edge cloud server is equivalent. Note that the proposed approach could also be applied to the different capabilities of each edge cloud server. To simulate different capabilities of edge cloud servers, we just need to change the receive task queue size for each edge cloud server. Each MEC server has the following property:

- Resource: R_m indicates the computation resource capability of MEC server m.
- Resource: R_C indicates the computation resource capability of Core Server C.

For edge cloud placement research issues, we focus on the long-term impact of the edge cloud placement solutions. The two evaluation indexes used in this paper are transmission delay, computation delay.

- The transmission delay, which is the delay for task \$t\$ from the mobile user to the assigned base station, the delay form base station to edge cloud server, and the delay from the edge cloud server to the core server and the delay from the core server to the mobile user.
- The computation delay is the delay for the mobile edge server and core server to finish the computation process.

Notation	Definition
M	set of edge cloud servers
V	set of base stations' location
L	set of links
C	Core server
m	The server in M
v	The base station in V
T	A set of tasks generated by mobile users
D_l^t	The transmission delay for task *t* upload through link *l* to MEC server
D_m^t	The transmission delay for task t uploaded to the core server *C* from MEC server *M*
MTT_t	the maximum delay allowed for task *t*
st	The volume of tasks generated by mobile users
R_m	The total resource capacity of MEC server *m*
R_C	The total resource capacity of MEC server *C*
$R_{m,C}$	Internet transmission ration between MEC server *M* and Cloud *C*

$R_{M,V}$	Internet transmission ration between MEC server M and base stations V
N_m	The number of tasks within the base station *v*
N_v	The number of tasks within the cluster *m*
$\alpha \times st$	The volume of tasks after computation in M
C_s	The cost of edge cloud server
C_c	The cost of edge cloud cloudlet
μ_m	The service rate for MEC server *m*
μ_C	The service rate for core server *C*

Table 2: Parameter table of the system model

Table 1 shows the symbols used in this essay. The mobile edge cloud system architecture G = (M∪V, L) is defined as follows. Let k and |V | is the number of edge cloud servers M, and base stations in V. Let $\{m_1, m_2, . . . , m_n\}$ be the edge cloud server in M. Let $\{v_1, v_2, . . . , v_n\}$ be the base station in V. Let $\{l_{1,1,} l_{1,2}, . . . , l_{k,n}\}$ be the link in L, for example, $l_{1,1}$ means there is a link between m1 and v1. Therefore, the (*M*) is the solution to the edge cloud placement problem. The optimization of the edge cloud server placement problem is the balance of transmission delay and computation delay.

Problem definition

Transmission delay: The transmission delay of offloading mobile users' requests to the core server contains: 1) the transmission delay for sending mobile users' tasks from mobile user device to the assigned nearby base station; 2) the network delay for uploading mobile users' tasks from the serving base station to nearby assigned MEC servers; 3) the communication delay in uploading requests of requests to the core server; 4) the sending delay for return the request results to mobile users. As we can see, those different locations of mobile users in edge networks will have a limited effect on the transmission delay between mobile users and their nearby base stations. Therefore, we will not take the wireless transmission delay between mobile users and their nearby base station into consideration to evaluate placement cases of MEC servers. Moreover, we assume all the tasks generated by mobile users are action recognition tasks. Thus the size of the result is 6 bits, which are so small that we can ignore sending delay for return the request results to mobile users.

Let D_l^t denote the transmission delay from request application $t \in T$ on the link $l \in L$. We use a valuable $\rho_{v,m}^l \in \{0,1\}$ shows whether there is a link $l \in L$ to connect base station $v \in V$ and MEC server $m\ in\ M$ or not. The transmission delay for uploading the application t of the mobile user within base station v and MEC server m is defined as:

$$Ti_m^t = \sum_{l \in L} \left(\rho_{(v,m)}^l \times D_l^t\right)$$

$$\text{s.t.} \quad \rho_{(v,m)}^l \in \{0,1\}, \forall v \in V, \forall m \in M, \quad (1)$$

$$\forall s_{t_j} \in S$$

Based on our model, the mobile edge cloud server could finish part of the requests which are sent by the mobile user. The final result will be processed by the core server. There are several factors could affect the transmission delay between mobile cloud edge server m to Core server C, such as, the request's size, the channel gain, the number of mobile edge cloud server which are sending requests in the same time, the state of CPU, buffer queue size, and parallel computing capacity. Let $Ti_{m,C}^{t}$ be the transmission delay from the base station to the edge cloud server in the edge cloud placement solution, and it could be formulated as follows:

$$\begin{aligned} & Ti_{m,C}^{t} = D_{m,C}^{t} \\ \text{s.t.} \quad & \forall s_{t_j} \in S \end{aligned} \tag{2}$$

Thus, for the overall transmission delay for task t from base station v to MEC server m and from MEC server m to the core server C is defined as follows:

$$\begin{aligned} & T_{m,C}^{t} = \sum_{l\in L}\left(\rho_{(v,m)}^{l} \times D_{l}^{t}\right) + D_{m}^{t} \\ \text{s.t.} \quad & \rho_{(v,m)}^{l} \in \{0,1\}, \forall v \in V, \forall m \in M, \\ & \forall st_t \in S \end{aligned} \tag{3}$$

Computation delay: The computation delay for task $t \in T$contains:

- The computation delay in the MEC server.
- The computation delay in the core server.

Users generate tasks are generated according to a homogeneous Poisson process with a ratio λ_t. We define G_t^m as the average request arrival rate of task t in the MEC server m where $G_t^m = \sum_{t\in T} \lambda_t \times \alpha_t^m$. In the meantime, the service time of MEC server m, for processing task t is equal to $1/\nu_m$ where the ν_m is the CPU rate for MEC server m. Thus the computation delay for task t, which belongs to edge cloud *m* takes the average computation time for x time experiments:

$$Tc_m^t = \frac{1}{\nu_m - \sum_{t\in T} G_t^m}$$

$$\text{s.t.} \qquad \nu_m - \sum_{t\in T} G_t^m > 0, \forall m \in M \tag{4}$$

The constrain graduate the MEC server $m \in M$ won't be overloaded.

In the core server, we define the $G_{t,m}^C$ as the average request arrival rate of task t from the MEC server m in the core server *C*. Thus the computation delay for task t which belongs to edge cloud *m* takes the average computation time for *x* times experiments:

$$Tc_{m,C}^t = \frac{1}{\nu_C - \sum_{t\in T, m\in M} G_{t,m}^C}$$

$$\text{s.t.} \qquad \nu_C - \sum_{t\in T, m\in M} G_{t,m}^C > 0 \tag{5}$$

Based on equation 4 and equation 5, the overall computation delay for task *t* in MEC server *m* and the Core server *C*:

$$Ti_{m,C}^t = \frac{1}{\nu_m - G_t^m} + \frac{1}{\nu_C - \sum_{m\in M} G_{t,m}^m}$$

$$\text{s.t.} \qquad \nu_C - \sum_{t\in T, m\in M} G_{t,m}^C > 0 \tag{6}$$

$$\nu_m - G_t^m \geq 0$$

Total number of mobile user: We denote $\sigma_m$as the mobile user located in the MEC server *m* server range. Since the server range for each base station is the same, we can simplify as follows:

$$\sigma_m = \sum_{l \in L} \rho^l_{(v,m)} \times N_v$$

$$\text{s.t.} \qquad \rho^l_{(v,m)} \in \{0,1\}, \forall v \in V, \forall m \in M, \tag{7}$$

$$\sum_{l \in L} \rho^l_{(v,m)} = |V|$$

Problem formulation

Minimization of overall delay problem: The most significant advantage of MEC is the decrease in the overall transmission delay. Tasks from mobile users will first upload to MEC servers and processed in MEC servers, which will cause transmission delay from the mobile user to the MEC server and computation delay in MEC servers. Second, the MEC servers would send the pre-processed tasks to the core server, which will cause the transmission delay from MEC servers to the core server and computation delay in the core server. As we mentioned above, the transmission delay for sending the result from the core server to the mobile user can be neglected. Thus, we define $\mathrm{T_{avg}(T, k, M)}$ as the average overall delay of deploying $|T|$ tasks with the deployment of k MEC servers. We form the problem of getting the minimum $\mathrm{T_{avg}(T, k)}$ by donating services from MEC servers and the core cloud to end personal users as follows:

$$\begin{aligned}
&\min T_{avg}\,(T,k,M) \\
&= \min\left(\frac{1}{R_{\text{total}}}\sum_{t\in T}\left(T_{m,C}^{t}+Ti_{m,C}^{t}\right)\right)0 \\
&= \min\left(\frac{1}{R_{\text{total}}}\cdot\sum_{t\in T}\left(\sum_{m\in M}\sum_{l\in L}\left(\rho_{(v,m)}^{l}\times D_{l}^{t}\right)+\sum_{m\in M} D_{m}^{t}\right)\right. \\
&\qquad\left.+\sum_{t\in T}\left(\sum_{m\in M}\frac{1}{\nu_m-G_t^m}+\frac{1}{\nu_C-\sum_{m\in M}G_{t,m}^{C}}\right)\right) \\
\text{s.t.}\quad & T_{m,C}^{t}+Ti_{m,C}^{t}\le \mathrm{MTT}_t \quad \forall t\in T,\forall m\in M, \\
&\sum_{t\in T}\sum_{l\in L}\rho_{(v,m)}^{l} f_t^m(st_t)\le R_m \quad \forall t\in T,\forall m\in M, \\
&\nu_m-G_t^m\ge 0, \\
&\nu_C-\sum_{m\in M}G_{t,m}^{C}\ge 0 \\
&\rho_{(v,m)}^{l}=\{0,1\},\forall l\in L, \\
&\sum_{l\in L}\rho_{(m,C)}^{l}=k
\end{aligned} \tag{8}$$

where the present all tasks form mobile users in the edge network.

The constrains state that 1) The response time needs to be smaller or equal than the Maximum tolerance delay in the tasks model, 2) For all the tasks offload in MEC server *m*, the requested resource should not exceed the resources that MEC server could provide. 3) For a new arrival task, the MEC server should assign computation resources to process the task. 4) For a new arrival task, the MEC server should assign computation resources to process the task. 5) The total number of MEC servers should be *k*. Mathematically, the optimal placement problem to enhance the quality of services(QoS) is an integer linear problem(ILP). In the above equation, the goal is to minimize the total transmission and computation delay for task $t \in T$ while minimizing the transmission delay and computation delay of the task from MEC servers and the core server.

The delay constraints in equation $T_{m,C}^{t} + Ti_{m,C}^{t} \le MTT_t$ specify that the total delay for task *t* does not exceed the maximum tolerant delay for all tasks *t* and all MEC server *m*. $\sum_{1}^{\nu_m} f_t^m(st) \le R_m$ is to make sure all tasks offloaded to MEC server m does not exceed the resources that MEC server *m* could provide. as for $\nu_m - \sum_{t\in T, m\in M} A_{t,m}^{C} \ge 0, \nu_C - \sum_{t\in T, m\in M} A_{t,m}^{C} \ge 0$ they are to make sure the MEC server *m* and the core server *C* could provide computation resources to task *t* received by MEC server *m*. The total number of deployed MEC servers is equal to *k*.

The optimal solution of equation 8 can be obtained by the PuLP package, using a branch-and-cut search method[20]. The branch-and-cut approach uses a search tree consisting of nodes, and each node represents a relaxed LP sub-problem of the original problem to solve. The branch-and-cut method includes clipping branches to reducing a sub-problem and dealing with branches to build two new nodes from a parent node and joining extra cutting planes to contract the LP relaxations to solve the original optimal placement algorithm. In general, the branch-and-cut search requires index calculation complexity to achieve the optimal solution in the worst case. Hence, the ILP problem is NP-hard.

Algorithm solutions

In this section, we will introduce the algorithm we designed to solve the placement problem. We will introduce the algorithm we proposed in this thesis. Based on the system architecture, we proposed above, the mobile users serviced by the same one base station will only be assigned into one cluster. Thus, we can simplify our problem to allocate the $|V|$ base stations into k clusters.

Our proposed algorithm is as shown in the algorithm, and it involves the following steps: The initial action is to select k positions for the mobile edge cloud by using the K-means algorithm. In this step, we cluster the set base station V into k cluster by minimizing the transmission delay and computation delay for task T, and the centre of k cluster are stored in X from line 2 to 5. The second step is to make a modification to cluster from line 7 to 10. In this step, we will calculate the density of $\mathrm{O}_{\mathrm{m}}^{\mathrm{i}}$ for the ith cluster, which is equation 8. If the density exceeds the density O_{m}We need to move the base station v to another cluster and re-calculate the transmission delay and computation delay.

Algorithm 1: Density placement Algorithm

```
Input: G: the graph G,
k: the number of MEC servers ,
$O_m$ : the density of mobile user for each MEC
server, T:the set of tasks
Output: $X$: the location for MEC server,
$D$: the total computation delay and transmission delay for task set $T$
1  Initialize cluster centroids $x_1, x_2, \dots, x_k$ randomly and set the density of each cluster
     $O_i = 0, i = 1, 2, 3 \dots k$;
2  for base station $\forall v \in V$ do
3      compute the total delay for all task $t$ in $v$ to each edge server;
4      select the minimal total delay;
5      update the $X$ and the density of each cluster $O$;
6  Compute the density of each cluster for deployment case $X$;
7  for base station $\forall v \in V$ do
8      for each other cluster, by $O_m$, unless already moved do
9          If $v$ requires to leave another cluster and this shift occurs and reduce
             the total delay, then exchange two base station allocation clusters;
10         If $v$ can be assigned without violating density constraints, Update $X$;
11 If no more changes in clusters, return X, D;
```

PERFORMANCE EVALUATION

In this section, we will introduce the experiment setup and result of the Density-based mobile edge cloud server placement algorithm we proposed in this thesis.

Dataset

The data set[5, 12, 13] used in the simulation comes from the Shanghai Telecom data set, which has the following attribute: the base station locations, the mobile users' record of accessing the Internet. There are around 7 million records of sending requests via base stations for a half year in the data set. The distribution of these mobile users is shown in Figure 5.1. Table 5.1 consists of examples of the base station's information, which are chosen randomly. Table 5.1 includes the ID of the base station, the user number in the server range of the base station, the location of the base station.

Experiment setup

We choose AMD Ryzen 5 2600 as the edge server CPU. The main frequency of CPU is 3.3 GHz, the RAM is 32 GB, and the GPU is NVIDIA GeForce RTX 2060. We choose Intel Core ^{TM} 9900K as the core server CPU, and the main frequency is 3.60 GHz, the RAM is 64GB, and the GPU is NVIDIA GeForce RTX 2080 Ti. The operating system for both computers is windows 10 home edition (64 bit). The program is written in python 3.6.1 with the Jupyter Notebook.

All the requests are action recognition task. The total number of the task is 100,000 and the size of tasks via from 300 KB to 1500 KB with maximum tolerance 2 seconds. The actions include the following nine actions: Archery, Long Jump, Baby Crawling, High Jump, Rock Climbing Indoor, Push-Ups, Rafting, Walking with a dog, and Yo Yo[37]. The mobile cloud edge server extracts 20 frames from videos, and then the 20 frames will be sent to the core server to finish the action recognition process. The core server receives the 20 frames from the mobile edge cloud server, does action recognition, and then returns the result to the mobile user directly. For the base station, the bandwidth value is set as 1000MHz; the power to transmit the request is 150mW, the noise for the transmission is the variance of complex white Gaussian channel noise.

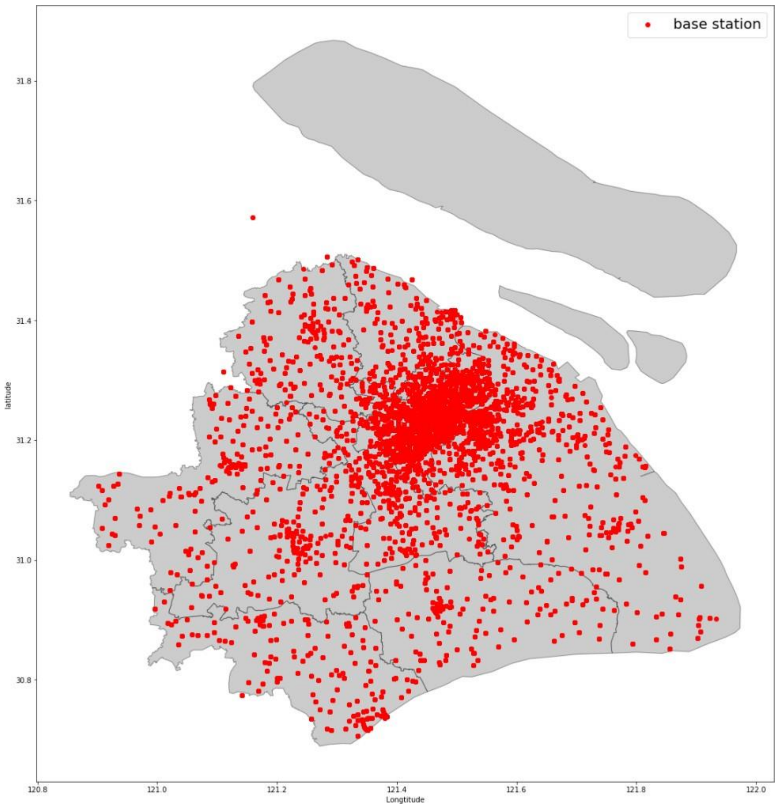


Figure 3:Distribution of base stations in Shanghai telecom dataset

ID	User number	Latitude	Longitude
7	5	31.2377	121.3825
16	165	31.48376	121.2748
193	367	31.01545	121.1548
446	212	31.28157	121.558
593	164	31.30685	121.5196
677	286	31.30113	121.1778
753	202	31.45348	121.2564
833	10	31.25677	121.4387
1005	24	31.39714	121.5077
1141	17	31.18606	121.4487
1325	53	31.24797	121.5134

Table 3: Samples of telecom dataset

Compared approach and performance evaluation

We evaluate the performance of our approach and another placement approach with regards to the average of transmission and computation delay for the tasks set *T* as follows:

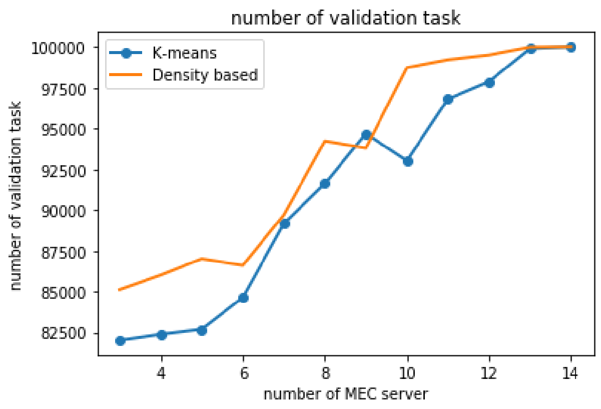
K-means: K-means clustering approach selects *k* random centroids initially. K-means will find the optimal cluster by repeating the following steps: 1) calculate the distance from the point to all centroids; 2) select the smallest distance; 3) update the centroid by taking the average of all points belongs to the same cluster. We apply the K-means approach to cluster the base stations into *k* groups and deploy the *k* MEC servers in the centre of the *k* clusters[38].

To evaluate the performance of our algorithm, we compare it with the K-means algorithm, which is implemented by most researchers. There are two perspectives we need to compare between the Density-based mobile edge cloud server placement algorithm and K-means.

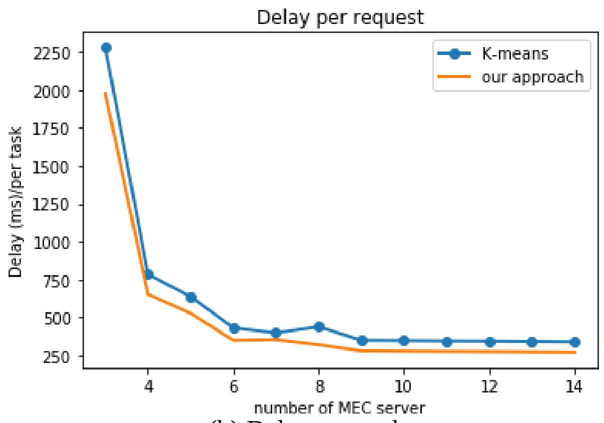
- The number of validation tasks: If the total delay exceeds the maximum tolerance, we will send a false to the mobile user.
- Total delay per task: The total delay for task consist 1) the transmission delay between the mobile user and MEC server; 2) the transmission delay between MEC server and mobile user for 20 frames; 3) the computation delay for the task in MEC server; 4) the computation delay for task in the core server.

Result analysis

The simulation result is presented in 5.2. Figure 5.2 (a) illustrates the number of validated tasks with the number of edge cloud servers from 3 to 15. The total number of tasks is 100,000. As we can see from Figure 5.2 (a), the number of validated tasks increases rapidly. The number of validated tasks in around 82,500 with 3 MEC serves deployed in the Shanghai area using the K-means method. The number of validated tasks in around 85,000 with 3 MEC serves deployed in the Shanghai area using the Density-based MECs deployment method. As shown in Figure 5.2(a) The number of the validated task increase consistently and finally reach to 100,000. The number of validated tasks for our proposed method, the Density-based MECs deployment method, is better than the compared approach, except the number of MEC servers is 9. Figure 5.2(b) states the relationship between the total delay per task and the number of MEC servers. The dealy per task decreases rapidly with the number of MEC server increase. The task delay is around 2250 ms with 3 MEC servers deployed in the simulation experiment with the K-means method. The task delay is around 2000ms, with 3 MEC servers deployed in the simulation experiment with our proposed method. With the number increasing from 3 to 9, the delay per task reduces from around 2000ms to 300 ms. With the number of MEC server reach ten and more, the delay per task reduces much more slowly. The overall performance for our proposed method could reduce around 10% delay than the delay of the other approach.



(a) number of validated tasks



(b) Delay per task

Figure 4: Results for the number of edge clouds

SUMMARY AND CONCLUSIONS

Mobile edge cloud has established as an essential approach which could provide computation, storage resource to enhance QoS for the high latency requirement tasks. In this essay, we work on the deployment of a mobile edge server on the mobile edge cloud. We formulate the deployment problem as an integer linear problem. We proposed a Density-based deployment of the MECs algorithm, which combines the K-means approach and integer linear programming. To evaluate the performance of our proposed approach, we design experiments using Shanghai's telecom data set. From the result of the experiment, we could prove our proposed method could reduce the delay per task around 10% and increase the number of validated tasks with a limited number of MEC servers.

There are several limitations of this research: 1) This thesis mainly focuses on mobile phone users, instead of wearable devices, pads, cars. 2)The complexity of tasks will be limited to action recognition tasks, but the size of the tasks will vary from 150KB to 1.5 MB. 3) The capability of each MEC server deployed is the same. 4)The computation delay is simulated with five times experiments. 5)During the experiment, We observed that the newly received tasks need to wait for a long time to get computation resources in some MEC servers. However, some other MEC server always has available resources for new tasks. Base on these above limitations, We expect to investigate a more efficient action recognition deep learning model and aim to the deployment approach combine with the task offloading strategy.

REFERENCES

1. Ni, J., Zhang, K., Lin, X., & Shen, X. S. (2017). Securing fog computing for Internet of things applications: Challenges and solutions. *IEEE Communications Surveys & Tutorials*, *20*(1), 601-628.
2. Zhang, N., Zhang, S., Zheng, J., Fang, X., Mark, J. W., & Shen, X. (2017). QoE driven decentralized spectrum sharing in 5G networks: Potential game approach. *IEEE Transactions on Vehicular Technology*, *66*(9), 7797-7808.
3. Qian, Y., Hu, L., Chen, J., Guan, X., Hassan, M. M., & Alelaiwi, A. (2019). Privacy-aware service placement for mobile edge computing via federated learning. *Information Sciences*, *505*, 562-570.
4. Jadeja, Y., & Modi, K. (2012, March). Cloud computing-concepts, architecture and challenges. In *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)* (pp. 877-880). IEEE.
5. Guo, Y., Wang, S., Zhou, A., Xu, J., Yuan, J., & Hsu, C. H. (2019). User allocation-aware edge cloud placement in mobile edge computing. *Software: Practice and Experience*.
6. Tortonesi, M., Govoni, M., Morelli, A., Riberto, G., Stefanelli, C., & Suri, N. (2019). Taming the IoT data deluge: An innovative information-centric service model for fog computing applications. *Future Generation Computer Systems*, *93*, 888-902.
7. Kim, H., Lee, H. J., Cho, H., Kim, E., & Hwang, J. (2018). Replacing self-efficacy in physical activity: unconscious intervention of the AR Game, Pokémon GO. *Sustainability*, *10*(6), 1971.

8. He, D., Westphal, C., & Garcia-Luna-Aceves, J. J. (2018, July). Network Support for AR/VR and Immersive Video Application: A Survey. In *ICETE (1)* (pp. 525-535).

9. Xiao, L., Wan, X., Dai, C., Du, X., Chen, X., & Guizani, M. (2018). Security in mobile edge caching with reinforcement learning. *IEEE Wireless Communications*, *25*(3), 116-122.

10. Zhang, N., Cheng, N., Gamage, A. T., Zhang, K., Mark, J. W., & Shen, X. (2015). Cloud assisted HetNets toward 5G wireless networks. *IEEE communications magazine*, *53*(6), 59-65.

11. Mach, P., & Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, *19*(3), 1628-1656.

12. Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A., & Shen, X. S. (2019). Delay-aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach. *IEEE Transactions on Mobile Computing*.

13. Xu, J., Wang, S., Bhargava, B. K., & Yang, F. (2019). A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing. *IEEE Transactions on Industrial Informatics*, *15*(6), 3538-3547.

14. Bisio, I., Delucchi, S., Lavagetto, F., & Marchese, M. (2012, December). Capacity bound of mop-based allocation with packet loss and power metrics in satellite communications systems. In *2012 IEEE Global Communications Conference (GLOBECOM)* (pp. 3311-3316). IEEE.

15. Bouet, M., & Conan, V. (2017, August). Geo-partitioning of mec resources. In *Proceedings of the Workshop on Mobile Edge Communications* (pp. 43-48).

16. Yang, G., Sun, Q., Zhou, A., Wang, S., & Li, J. (2016, October). Access point ranking for cloudlet placement in edge computing environment. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)* (pp. 85-86). IEEE.

17. Fan, Q., & Ansari, N. (2017, May). Cost aware cloudlet placement for big data processing at the edge. In *2017 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE.

18. Xu, Z., Liang, W., Xu, W., Jia, M., & Guo, S. (2015, October). Capacitated cloudlet placements in wireless metropolitan area networks. In *2015 IEEE 40th Conference on Local Computer Networks (LCN)* (pp. 570-578). IEEE.

19. Jia, M., Cao, J., & Liang, W. (2015). Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, *5*(4), 725-737.

20. Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy?. *Computer*, *43*(4), 51-56.

21. He, Y., Yu, F. R., Zhao, N., Leung, V. C., & Yin, H. (2017). Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach. *IEEE Communications Magazine*, *55*(12), 31-37.

22. Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, *13*(18), 1587-1611.

23. Huerta-Canepa, G., & Lee, D. (2010, June). A virtual cloud computing provider for mobile devices. In *proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond* (pp. 1-5).

24. Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. *Future generation computer systems*, *29*(1), 84-106.

25. Gupta, L., Jain, R., & Chan, H. A. (2016). Mobile edge computing—An important ingredient of 5G networks. *IEEE Software Defined Networks Newsletter*.

26. Ahmed, E., & Rehmani, M. H. (2017). Mobile edge computing: opportunities, solutions, and challenges.

27. Cicconetti, C., Conti, M., & Passarella, A. (2020). Architecture and performance evaluation of distributed computation offloading in edge computing. *Simulation Modelling Practice and Theory*, *101*, 102007.

28. Tran, T. X., Hajisami, A., Pandey, P., & Pompili, D. (2017). Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, *55*(4), 54-61.

29. Zhang, C., & Zheng, Z. (2019). Task migration for mobile edge computing using deep reinforcement learning. *Future Generation Computer Systems*, *96*, 111-118.

30. Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K. L. A., Elkhatib, Y., ... & Al-Fuqaha, A. (2019). Unsupervised machine learning for networking: Techniques, applications and research challenges. *IEEE Access*, *7*, 65579-65615.

31. Wang, J., Li, M., Chen, J., & Pan, Y. (2010). A fast hierarchical clustering algorithm for functional modules discovery in protein interaction networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *8*(3), 607-620.

32. Alsabti, K., Ranka, S., & Singh, V. (1997). An efficient k-means clustering algorithm.

33. Maugis, C., Celeux, G., & Martin-Magniette, M. L. (2009). Variable selection for clustering with Gaussian mixture models. *Biometrics*, *65*(3), 701-709.

34. Ciaparrone, G., Sánchez, F. L., Tabik, S., Troiano, L., Tagliaferri, R., & Herrera, F. (2020). Deep learning in video multi-object tracking: A survey. *Neurocomputing*, *381*, 61-88.

35. Gu, L., Zeng, D., Guo, S., Barnawi, A., & Xiang, Y. (2015). Cost efficient resource management in fog computing supported medical cyber-physical system. *IEEE Transactions on Emerging Topics in Computing*, *5*(1), 108-119.

36. Bashier, E. B. (2020). *Practical Numerical and Scientific Computing with MATLAB® and Python*. CRC Press.

37. Soomro, K., Zamir, A. R., & Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

38. Cao, K., Li, L., Cui, Y., Wei, T., & Hu, S. (2020). Exploring Placement of Heterogeneous Edge Servers for Response Time Minimization in Mobile Edge-Cloud Computing. *IEEE Transactions on Industrial Informatics*.