**MOBILITY-AWARE ENERGY MINIMAL TASK OFFLOADING WITH DELAY CONSTRAINTS IN MOBILE EDGE COMPUTING ENVIRONMENT**

*Submitted in fulfilment of requirements for the degree of*

*Bachelor of Technology*

*by*

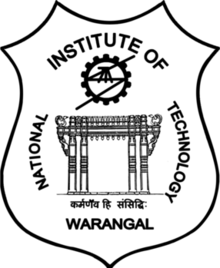
**Vajjhala V V S Srirama Savitru(167264)**

**P.V.Karthik Reddy(167245)**

**Raja Chitawle (167250)**

*Under the esteemed guidance of*

**Dr R.R.Rout**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

2019 – 2020

**APPROVAL SHEET**

The Project Work entitled **Mobility-aware energy minimal task offloading with delay constraints in Mobile Edge Computing Environment** by **Vajjhala V V S Srirama Savitru, P.V.Karthik Reddy, Raja Chitawle** is approved for the degree of Bachelor of Technology in Computer Science and Engineering at National Institute of Technology Warangal during the year 2019-2020.

**Examiners**

**Supervisor**

Dr R.R.ROUT

Associate Professor, CSE Dept.

**Chairman**

Dr P.RADHA KRISHNA

Head of Department, CSE

NIT Warangal

Date:

Place:

**DECLARATION**

We declare that this written submission represents our ideas in our words and where other ideas or words have been included. We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and we have not misrepresented or fabricated or falsified any idea/ data/ fact/ source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Vajjhala V V S Srirama Savitru

167264

Date:

(Signature)

P.V.Karthik Reddy

167245

Date:

(Signature)

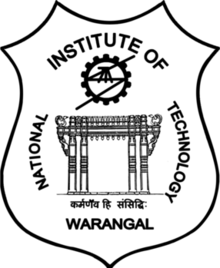
Raja Chitawle

167250

Date:

**NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the project work entitled **“MOBILITY-AWARE ENERGY MINIMAL TASK OFFLOADING WITH DELAY CONSTRAINTS IN MOBILE EDGE COMPUTING ENVIRONMENT”** is a bonafide record of work carried out by Vajjhala V V S Srirama Savitru(167264), P.V.Karthik Reddy(167245) and Raja Chitawle (167250), submitted to the faculty of Computer Science and Engineering Department, in fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering at National Institue of Technology, Warangal during the academic year 2019-2020.

Dr. R.R.Rout

Project guide

Associate Professor,

Department of CSE

NIT Warangal

Dr. P.Radha Krishna

Head of the Department

Department of CSE

NIT Warangal

**ACKNOWLEDGEMENT**

We consider it as a great privilege to express our deep gratitude to many respected personalities who guided, inspired and helped us in the successful completion of our project.

We would like to express our deepest gratitude to our guide, Dr R.R.Rout, Department of Computer Science and Engineering, National Institute of Technology, Warangal, for his constant supervision, guidance, suggestions and invaluable encouragement during this project. He has been a constant source of inspiration and helped us in each stage.

We are grateful to Dr P.Radha Krishna, Head of the Department, Computer Science and Engineering, National Institute of Technology, Warangal for his mortal support to carry out this project.

We are very thankful to the Project Evaluation Committee, for their kind cooperation and support given throughout our project work. We are also thankful to all our friends who have given valuable suggestions and help in all stages of development of the project.

Vajjhala V V S Srirama Savitru(167264)

P.V.Karthik Reddy(167245)

Raja Chitawle (167250)

**ABSTRACT**

Mobile Edge Computing (MEC) has emerged as a prospective computing paradigm to provide pervasive computing and storage services for mobile and big data applications. In MEC, many MEC servers are deployed at base stations to establish a mobile edge network (MEN). Mobile users are allowed to ofﬂoad mobile tasks to nearby mobile edge servers to speed up their mobile applications. Nevertheless, various challenges, especially the quality of such a mobile task ofﬂoading in the edge computing environment, are yet to be properly tackled. Most studies and related ofﬂoading strategies based on the assumption that mobile users are fully stationary when they are ofﬂoading tasks to edge servers. However, this is often not realistic in real-world where edge users are keeping moving, which has a great impact on the success of task ofﬂoading and further affects the response time of mobile applications.

Our work aims at reducing the energy consumption and boost Quality of Services(QoS) by the scheduling of assignment of mobile tasks to MECs in their trajectories predicted by a Random waypoint Model. Specifically, we collectively contemplate the task properties, the user mobility and delay constraints. The problem is formalized as an optimization and constraint satisfaction problem. A heuristic-based solution is proposed for scheduling mobiles tasks to MECs. We used Melbourne CBD dataset consisting of 126 base stations and Random Waypoint Mobility model to study the performance of the proposed work. We compared metrics like Task Acceptance rate and Energy Consumption for the proposed approach and some baseline approaches. The results show that our work can significantly scale back the energy consumption in MENs and also enhance QoS by executing task under constrained time.

Contents

[INTRODUCTION 1](#_Toc43394537)

[1.1 Cloud Computing 1](#_Toc43394538)

[1.1.1 Advantages of cloud computing 1](#_Toc43394539)

[1.1.2 Mobile Cloud Computing 1](#_Toc43394540)

[1.2 Mobile Edge Computing 2](#_Toc43394541)

[1.2.1 Advantages of Using Edge Computing 3](#_Toc43394542)

[1.2.2 Examples of Edge Computing 3](#_Toc43394545)

[1.3 Dealing with User Mobility in MEC 4](#_Toc43394549)

[RELATED WORKS 5](#_Toc43394550)

[REVIEW ON THE PRESENT INVESTIGATION 7](#_Toc43394551)

[3.1 System Model 7](#_Toc43394552)

[3.2 Energy Consumption and Task Delay 7](#_Toc43394553)

[3.2.1 Task Uploading 7](#_Toc43394554)

[3.2.2 Task Transmission 8](#_Toc43394555)

[3.2.3 Task Execution 8](#_Toc43394556)

[3.3 Problem Formulation 9](#_Toc43394557)

[3.4 Algorithm 10](#_Toc43394558)

[EVALUATION & RESULTS 14](#_Toc43394559)

[CONCLUSION 19](#_Toc43394560)

[APPENDIX 20](#_Toc43394561)

[Random Waypoint Model 20](#_Toc43394562)

[REFERENCES 21](#_Toc43394563)

List of Figures

[Figure 1 System Model 7](#_Toc43397876)

[Figure 2 Illustrative Example for Algorithm 1 13](file:///F:\finals\Project_Report.docx#_Toc43397877)

[Figure 3 Melbourne CBD Base stations 14](#_Toc43397878)

[Figure 4 Task Completion% vs Task Deadline 15](#_Toc43397879)

[Figure 5 Energy Consumption vs Number of Users 16](#_Toc43397880)

[Figure 6 Avg Energy Consumption vs Number of Users 17](#_Toc43397881)

[Figure 7 Task Input Size vs Energy Consumption 17](#_Toc43397882)

[Figure 8 Energy Consumption vs Server Computation Capacity 18](#_Toc43397883)

List of Tables

[Table 1 Symbols Description 9](#_Toc43397851)

[Table 2 Simulation Parameters 15](#_Toc43397852)

**CHAPTER 1**

# INTRODUCTION

## Cloud Computing

Cloud computing is a method of delivering technology to the consumer by using Internet servers for processing and data storage, while the client system uses the data.

Cloud Computing provides an alternative to the on-premises datacenter. With an on-premises datacenter, we have to manage everything, such as purchasing and installing hardware, virtualization, installing the operating system, and any other required applications, setting up the network, configuring the firewall, and setting up storage for data. After doing all the set-up, we become responsible for maintaining it through its entire lifecycle.

But in Cloud Computing, a cloud vendor is responsible for the hardware purchase and maintenance. They also provide a wide variety of software and platform as a service. We can take any required services on rent. The cloud computing services will be charged based on usage.

## Advantages of cloud computing

**Cost:** It reduces the huge capital costs of buying hardware and software.

**Speed:** Resources can be accessed in minutes, typically within a few clicks.

**Scalability:** We can increase or decrease the requirement of resources according to the business requirements.

**Productivity:** While using cloud computing, we put less operational effort. We do not need to apply to patch, as well as no need to maintain hardware and software. So, in this way, the IT team can be more productive and focus on achieving business goals.

**Reliability:** Backup and recovery of data are less expensive and very fast for business continuity.

**Security:** Many cloud vendors offer a broad set of policies, technologies, and controls that strengthen our data security.

## Mobile Cloud Computing

The users’ requirements on data rates and quality of service (QoS) are exponentially increasing. Moreover, the technological evolution of smartphones, laptops and tablets enables to emerge new high demanding services and applications. Although new mobile devices are more and more powerful in terms of a central processing unit (CPU), even these may not be able to handle the applications requiring huge processing in a short time. Moreover, high battery consumption still poses a signiﬁcant obstacle restricting the users to fully enjoy highly demanding applications on their own devices. This motivates the development of mobile cloud computing (MCC) concept allowing cloud computing for mobile users. In the MCC, user equipment (UE) may exploit computing and storage resources of powerful distant centralized clouds (CC), which are accessible through a core network (CN) of a mobile operator and the Internet. Tasks requiring huge computation capacity are first offloaded to MEN and then executed in one or more MEC servers in the network. The MCC brings several advantages;

1. extending battery lifetime by ofﬂoading energy-consuming computations of the applications to the cloud,
2. enabling sophisticated applications to mobile users, and
3. providing higher data storage capabilities to the users.

Cloud computing revolves around large, centralized servers stored in data centres. After data is created on an end device, that data travels to that central server for processing. This architecture becomes cumbersome for processes that require intensive computations. Latency becomes the main problem here. So, Cloud Computing provides a low Quality of Service to a mobile user using its services.

To address the problem of a long latency and to boost Quality of Service, the cloud services should be moved to the proximity of the UEs, i.e., to the edge of the mobile network as considered in newly emerged edge computing paradigm.

## Mobile Edge Computing

The term “Edge computing” refers to computing as a distributed paradigm. It brings data storage and computes power closer to the device or data source where it’s most needed. Information is not processed on the cloud filtered through distant data centres; instead, the cloud comes to you. This distribution eliminates lag-time and saves bandwidth and improves the quality of service.

MEC is a network architecture that enables IT and cloud-computing capabilities at the edge of the cellular network. The main idea behind the architecture is to reduce network congestion and improve applications by performing related processing tasks closer to the end-user. The technology is designed to be implemented at cellular base stations, providing rapid deployment of applications and other customer services. Nevertheless, in the conventional MCC, the cloud services are accessed via the Internet connection while in the case of the edge computing, the computing/storage resources are supposed to be in proximity of the UEs (in sense of network topology). Hence, the MEC can offer signiﬁcantly lower latencies and jitter when compared to the MCC. Moreover, while the MCC is a fully centralized approach with farms of computers usually placed at one or few locations, edge computing is supposed to be deployed in a fully distributed manner. On the other hand, edge computing provides only limited computational and storage resources with respect to the MCC.

Computational needs are more efficiently met when using edge computing. Wherever there is a requirement of collecting data or where a user performs a particular action, it can be completed in real-time.

## Advantages of Using Edge Computing

### Improved Performance

Besides collecting data for transmission to the cloud, edge computing also processes analyses and performs necessary actions on the collected data locally. Since these processes are completed in milliseconds, it’s become essential in optimizing technical data, no matter what the operations may be.

Transferring large quantities of data in real-time in a cost-effective way can be a challenge, primarily when conducted from remote industrial sites. This problem is remedied by adding intelligence to devices present at the edge of the network. Edge computing brings analytics capabilities closer to the machine, which cuts out the middle-man. This setup provides for less expensive options for optimizing asset performance.

### Reducing Operational Costs

In the cloud computing model, connectivity, data migration, bandwidth, and latency features are pretty expensive. This inefficiency is remedied by edge computing, which has a significantly less bandwidth requirement and less latency. By applying edge computing, a valuable continuum from the device to the cloud is created, which can handle the massive amounts of data generated. Costly bandwidth additions are no longer required as there is no need to transfer gigabytes of data to the cloud. It also analyses sensitive IoT data within a private network, thereby protecting sensitive data. Enterprises now tend to prefer edge computing. This is because of its optimizable operational performance, address compliance and security protocols, alongside lower costs.

Edge computing can help lower dependence on the cloud and improve the speed of data processing as a result. Besides, there are already many modern IoT devices that have processing power and storage available. The move to edge processing power makes it possible to utilize these devices to their fullest potential.

## Examples of Edge Computing

The best way to demonstrate the use of this method is through some key edge computing examples. Here are a few scenarios where edge computing is most useful:

### Autonomous Vehicles

Self-driven or AI-powered cars and other vehicles require a massive volume of data from their surroundings to work correctly in real-time. Delay would occur if cloud computing were used.

### Streaming Services

Services like Netflix, Hulu, Amazon Prime, and the upcoming Disney+ all create a heavy load on network infrastructure. Edge computing helps create a smoother experience via edge caching. This is when popular content is cached in facilities located closer to end-users for easier and quicker access.

### Smart Homes

Similar to streaming services, the growing popularity of smart homes poses a problem. It’s now too much of a network load to rely on conventional cloud computing alone. Processing information closer to the source means less latency and quicker response times in emergency scenarios. Examples include medical teams, fire, or police deployment. MEC paradigm is also versatile for scheduling resources for mobile tasks.

## Dealing with User Mobility in MEC

Effective and efficient offloading decisions can reduce the delay and energy consumption and improve quality of service (QoS). So far, the research works on MEC often focus on the offloading decision problem which focuses on obtaining the optimal offloading scheme considering different environments and user requirements like the acceleration of computing and the optimization of energy consumption. Conventional methods for making offloading decisions usually consider that user position is static and time-invariant. Such methods may not be appropriate in a real-world scenario since edge users are with high mobility. The optimal offloading solution becomes more challenging when considering user mobility. It has a large impact on task execution. Each mobile user can have multiple MEC servers in their communication range. Assigning tasks to directly connected MEC servers may not lead to an optimal offloading decision. Users may upload tasks to a MEC server and get results via another server which involves communication among the MEC servers and transmission cost.

To effectively address the impact of user mobility on task execution and offloading decision problem, we consider resource allocation to mobile users based on their trajectories predicted by Random waypoint Model. We also constraint task execution by a certain time delay to boost Quality of Service (QoS). The delay constraint is specific to each task. By considering the user mobility, task properties and the resource distribution in the MEN, we formally model the problem as an optimization and constraint satisfaction problem. We then proposed an approximate approach to obtain a near-optimal offloading decision satisfying the delay constraint. We conduct extensive simulation experiments and the results show that the proposed work can significantly reduce the energy consumption of tasks in MEC networks.

**CHAPTER 2**

# RELATED WORKS

With the ubiquity of smart mobile devices like intelligent mobile phones and smartwatches, lots of resource-intensive applications like face recognition augmented reality, interactive gaming is developing rapidly. However, this immense growth introduces significant challenges to the physical limitations of mobile devices, such as computation capability, battery life.

Cloud computing is one of the technologies that can help to save the energy of mobile devices and it is deployed broadly. In cloud computing, devices can ofﬂoad their computationally intensive tasks to the cloud server, where the immense volume of computing power exists to facilitate the work of resource-limited devices. However, the imposed delay by cloud computing is not suitable for many of today’s delay-sensitive applications.

Mobile Edge Computing (MEC) was proposed to overcome the long delay of task ofﬂoading in cloud computing. MEC collects data from the mobile devices and processes it at the edge of the network without sending it to the traditional cloud. Ofﬂoading the computational-intensive tasks to the MEC will lead to saving the energy of mobile devices with a low delay. Although the transmission distance from edge infrastructure to the cloud centre is eliminated, the energy consumption and time delay for the wireless communication and task computation remains, which needs to be managed carefully.

Despite the relatively high computing power at the edge infrastructure compared to each device, it has to be shared by a different type of tasks, such as computation-intensive task, delay-sensitive task, etc. For these scenarios, Collaborative Edge Computing (CEC) was preferred to consider the relationships between servers, such as the hierarchical servers. CEC allows multiple servers to collaboratively ofﬂoad different type of tasks to efﬁciently reduce time delay and energy consumption. In [1], J Wang et al studied energy-efficient task offloading in CEC environment and an offloading scheme based on the Hungarian algorithm was derived. [2] focused on the delay constrained energy minimization problem in D2D-assisted MEC network where firstly feasible tasks were found out based on the delay constraint and then low complexity task switching algorithm was derived for global energy minimization.

In the above mentioned MEC resource management schemes, the mobility of mobile users is not considered. These works assume that users are stationary and the communication between edge servers and users is reliable. However, this is unrealistic in real-world application scenarios, where mobile users are with universal mobility. Once mobile users move out of the transmission range of edge servers, the offloaded tasks will fail, and such failures will extend the response time of mobile applications and lead to a waste of edge computing resources. Therefore, mobility-aware tasks offloading approaches that can capture the mobility of edge users and make smart offloading decisions are in high need. In [3], Chunrong Wu et al discussed mobility aware task offloading to find the most suitable cloud or edge resource for every task in real-time. They used SGAN, a deep-learning-based method for predicting users’ trajectories. In [4], Zhaolin Liu et al worked on reducing task migration by optimal task offloading in MEC environment and a suboptimal algorithm named genetic algorithm based allocation algorithm (GAAA) was proposed to solve this NP-hard problem. These works didn’t make use of cooperation between MECs at multiple BSs to avoid task migration. In [5], Zi wang et al worked-on mobility aware latency optimal task offloading in which tasks can be offloaded to MEC servers situated along the users’ trajectories.

However, our work aims at Mobility aware delay constrained energy minimization problem by optimizing the task offloading decision. The major contributions are:

* Mobility aware delay constrained energy minimal task offloading scheme is proposed by making use of cooperation between MECs at base stations along the users’ trajectories.
* Mobility of users in our work refers to the random waypoint model which is widely used in many other research works related to mobile computing
* The performance of the proposed scheme was evaluated by comparing energy consumption and task completion percentage with certain conventional schemes.

**CHAPTER 3**

# REVIEW ON THE PRESENT INVESTIGATION

## System Model



Figure 1 System Model

As shown in Figure 1 System Model, Mobile Edge Network (MEN) consists of several Base Stations (BS), each equipped with a MEC server which is capable of receiving executing and transmitting computation tasks offloaded by users within the signal range of BS. Central Base Station (CBS) helps in controlling and monitoring all base stations. Mobile users are allowed to offload their tasks to MEC servers to improve the task latency and to reduce energy consumption. Mobile users can move out of the signal range of a BS at any time because of the mobility. In fig 1, the user moved out of after offloading task . In this case, MEC at can transmit the offloaded task to MEC at where the user currently is in. This way, all MECs, in the user’s trajectory are capable of executing the offloaded task.

We consider be the mobile users moving around the area under MEN. Each user is having a task , denoted as a triple where is the size of the computational task, is the required computation resource (in cycles) and is the task deadline. Also, there are a total of base stations each with signal range and the associated MEC having computation capacity .

## Energy Consumption and Task Delay

## Task Uploading

Mobile users first offload the task and the communication rate between user and server can be expressed as

where indicates the channel bandwidth, is the distance between the user and MEC server , is the noise power spectral density, and is the channel fading parameter. The uploading time depends on the communication rate and task size and can be denoted as

and the uploading energy can be calculated by

where is the transmission power of mobile

## Task Transmission

As MECs can communicate with each other by making use of CBS, the transmission rate between the server and can be expressed as

where is the distance between the server and . The transmission time and transmission energy can be formulated as

where is the transmission power of the MEC server .

## Task Execution

The computation time of task in server can be denoted as

and the computation energy can be calculated as

where is the computation power of MEC server .

The download transmission delay and energy are not considered because the size of the result of the task after processing is very less and the energy consumption and delay are negligible.

Table 1 Symbols Description

|  |  |
| --- | --- |
| Symbol | Description |
|  | size of the computational task |
|  | computation resource (in cycles) needed for the task |
|  | deadline for task |
|  | signal range of MEC server |
|  | computation capacity of MEC server |
|  | communication rate between user and server |
|  | transmission rate between servers and |
|  | channel bandwidth |
|  | noise power spectral density |
|  | channel fading parameter |
| , | delay and energy consumption for task uploading from user to server |
|  | delay and energy consumption for task transmission between servers and |
| , | delay and energy consumption for the execution of a task from user in server |
|  | distance between the user and MEC server |
|  | transmission power of user mobile |
|  | transmission power of server |
|  | computation power of server |

## Problem Formulation

We assume that each user is having a task which can be offloaded. As the user is moving, the MECs to which user can offload the task varies with time and we denote as the set of available MECs at time and let be the total time user is moving. We aim to find a MEC server from the set of available servers in user trajectory satisfying task deadline and minimizing the total energy consumption. The task offloading problem can then be modelled as

It’s very clear from the conditions that the task should be computed only once at some server and can’t be divided.

The energy and time in the above formulation can be expressed as

and indicates that total energy and total time includes energy and time spent in uploading, transmission and computation of a task.

## Algorithm

The above problem is a constraint-satisfaction problem and it is NP-complete. We propose a heuristic to solve the problem and it is as follows:

Each user offloads task with the required information: the size of the computational task, required computation resource (in cycles) and the task deadline. The CBS then assigns each task to a MEC server satisfying task deadline and least energy consumption. The delay constrained energy minimal task offloading algorithm shortly referred to as DCEMTO algorithm is shown in Algorithm 1.

For each user , at each time step , available MEC servers is found out. Among the servers in , our goal is to find out two servers, one for execution to which task can be assigned and one for transmission from which task can be offloaded to a server( situated in the user’s locality in next timestep. As the task assignment has to satisfy delay constraint, the time needed for uploading, transmission, execution is also calculated and checked for delay constraint in each timestep.

For each available server , the energy consumed due to transmission (up to timestep ) and execution is calculated. This process is repeated for all time steps and the server with the least energy is assigned to the task for execution.

|  |
| --- |
| **Algorithm 1** DCEMTO algorithm |
| **Input**: mobile users , base stations   1. **for each** user **do** 2. **for each** timestep **do** 3. get available MEC servers at timestep 4. **for each** server **do** 5. **if**  **then** 7. **if**  **then** 8. **else** 10. **endif** 11. **else** 13. **if**  **then** 15. **else** 17. **endif** 18. **end** 20. let be the user ’s position at 21. **for each** server **do** 23. **end** 25. **end** 26. **end** 27. **output** |

The energy consumed due to transmission at timestep can be found out by adding transmission energy up to timestep and energy needed to offload task from server’s location to user’s locality in next time step .

In the end, the algorithm outputs , the energy-optimal server satisfying delay constraint of task .

As an explanation of our scheme, we use an illustrative example in Figure 2. User 1 offloads tasks to available MEC servers in timestep 0. Among the two servers, is having less energy consumption after uploading, and execution. Hence is chosen for execution. As is having less energy for transmission, it is used to offload the task to MEC servers available in next timestep.

|  |  |
| --- | --- |
| **Execution** | **Transmission** |
|  |  |
|  |  |

In the second timestep, available MEC servers are . Among the two servers, is having less energy consumption after uploading and execution. Hence is chosen for

|  |  |
| --- | --- |
| **Execution** | **Transmission** |
|  |  |
|  |  |

execution. As is having less energy for transmission, it is used to offload the task to MEC servers available in next timestep.

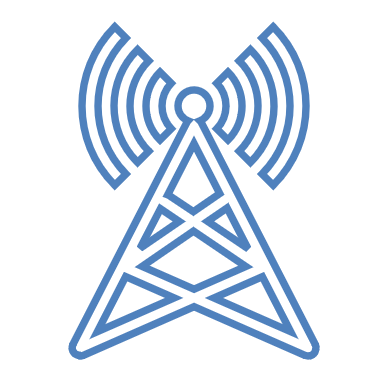
In the last timestep, available MEC servers are . Among the two servers, is having less energy consumption after uploading and execution. Hence is chosen for execution.

|  |
| --- |
| **Execution** |
|  |
|  |

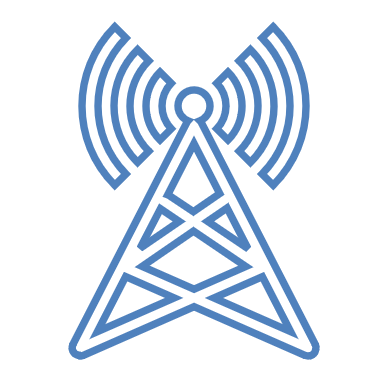
As this is the last timestep, there will be no further transmission from . Now, CBS will assign the task to one among having least execution energy .

t1tr=0.2,

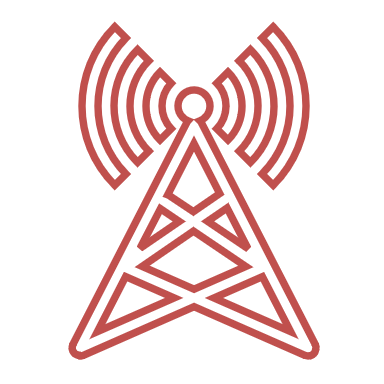
e1tr=0.1

BS5

|  |  |
| --- | --- |
| t1ex | 0.2 |
| e1ex | 0.2 |

BS3

|  |  |
| --- | --- |
| t1ex | 0.5 |
| e1ex | 0.3 |
| t1tr | 0.1 |
| e1tr | 0.5 |

BS1

|  |  |
| --- | --- |
| t1ex | 0.4 |
| e1ex | 0.6 |
| t1tr | 0.1 |
| e1tr | 0.3 |

User 1

Task

Max Delay = 1s

t1u=0.2,

e1u=0.3

t1tr=0.2,

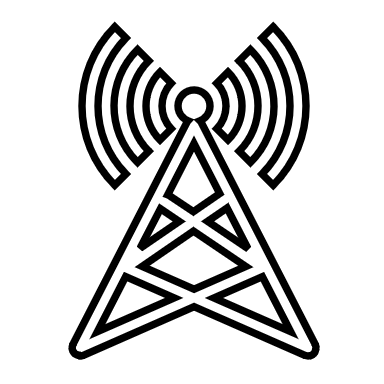
e1tr=0.4

t1u=0.2,

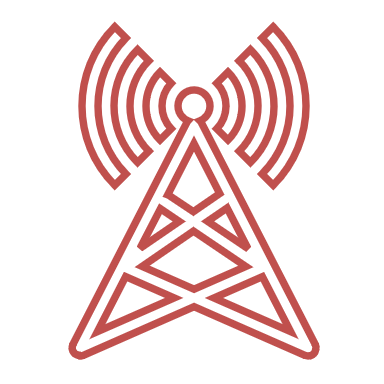
e1u=0.2

t1tr=0.1,

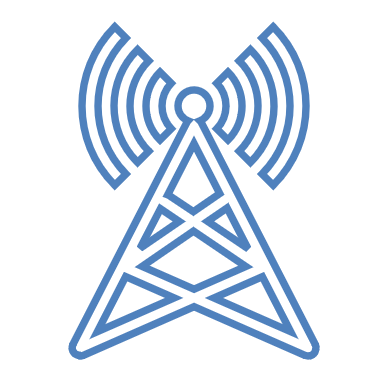
e1tr=0.3

BS6

|  |  |
| --- | --- |
| t1ex | 0.1 |
| e1ex | 0.2 |

BS4

|  |  |
| --- | --- |
| t1ex | 0.4 |
| e1ex | 0.2 |
| t1tr | 0.3 |
| e1tr | 0.2 |

BS2

|  |  |
| --- | --- |
| t1ex | 0.5 |
| e1ex | 0.6 |
| t1tr | 0.2 |
| e1tr | 0.5 |

t1tr=0.2,

e1tr=0.5

timestep 2

timestep 1

timestep 0

Figure 2 Illustrative Example for Algorithm 1

**CHAPTER 4**

**CHAPTER 4**

# EVALUATION & RESULTS

In this section, we evaluated the performance of our proposed algorithm to achieve energy minimal task offloading with users’ mobility and delay constraints into consideration. we used data of base stations within the Melbourne central business district area in Australia Figure 3 Melbourne CBD Base stations, which has a total area of 6.2 km2 provided by the Australian Communications and Media Authority [6]. There are a total of 126 base stations situated inside the Melbourne CBD area.

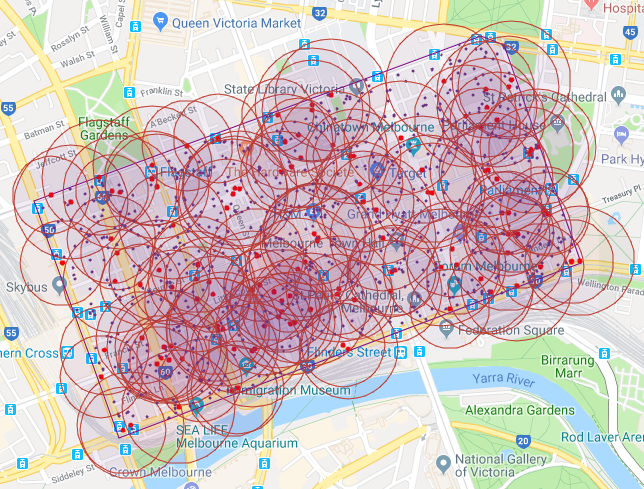


Figure Melbourne CBD Base stations

We follow the Random waypoint mobility model for the users moving in the area covered by the base stations provided in the dataset. The random waypoint model is a random model for the movement of mobile users, and how their location, velocity and acceleration change over time.

Without loss of generality, we refer to the parameter settings in the existing works [2], [7]. The coverage radius of BS is randomly taken varying from 70 to 100 meters and the associated MEC’s CPU capacity is in [7, 20] GHz, computation power is in [3,5] watts and transmission power ranges from 0.1 to 1 watt. The bandwidth of the communication channel is set up as 1MHz. Users are moving with the speed of 1~3 MPH and generate tasks with input data size varies from 1 MB to 3 MB having CPU requirement is in [1,10] GHz and deadline constraint within [0.1,1] seconds. Users’ mobiles are having CPU capacity ranges from 1 to 10 GHz, transmission power is randomly taken from [7,15] watts and computation power from [7,10] watts. We consider mobile-execution and random-allocation as our baseline algorithms:

* Local-Execution: Feasible Tasks satisfying delay constraint are executed within the mobiles without offloading.
* Random-Allocation: Each user’s task is assigned to a server randomly chosen from the servers present along the user’s trajectory.

Table 2 Simulation Parameters

|  |  |
| --- | --- |
| Parameter | Value/Range |
| Coverage radius of BS | 70 to 100 meters |
| MEC’s CPU capacity | [7, 20] GHz |
| MEC’s computation power | [3,5] watt |
| MEC’s transmission power | [0.1,1] watt |
| Channel bandwidth | 1MHz |
| User’s mobility speed | 1~3 MPH |
| Input data size | [1,3] MB |
| Task CPU requirement | [1,10] GHz |
| Task deadline constraint | [0.1,1] sec |

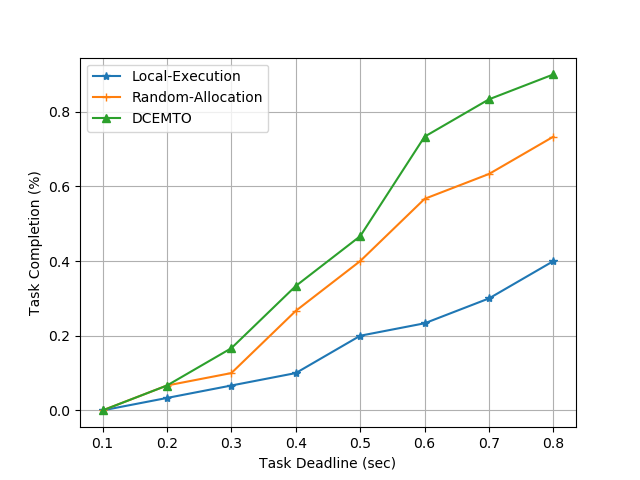


Figure Task Completion% vs Task Deadline

We fixed the number of users to 30 and plotted Task Completion percentage by varying Task deadline constraint in Figure 4. It depicts the higher Task completion percentage in DCEMTO compared to the other two approaches. DCEMTO’s task completion is 9.16% higher than Random-Allocation and 27.08% higher than Local-Execution on average. Also, Task completion percentage increases by increasing Deadline constraint.

Figure 5 shows how energy consumption varies with the change in the number of users. In general, total energy consumption increases with an increase in the number of users. But the plot is not continuously increasing. It is due to less task completion percentage in some cases. Hence, in the next study, Average Energy Consumption is compared to a varying number of users. DCEMTO’s energy consumption is 11.22% efficient than Random-Allocation method.

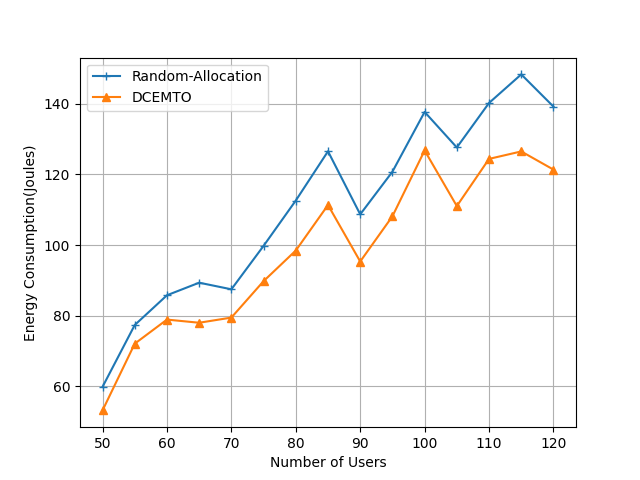


Figure Energy Consumption vs Number of Users

Figure 6 indicates the variation of average energy consumption with a change in the number of users. DCEMTO outperforms other approaches. Average energy consumption in DCEMTO is 10.5% energy efficient than Random-Allocation and 46.5% efficient than Local-Execution.

Figure 7 shows how energy consumption varies with varying task input size. It’s clear that higher the task input size, energy consumption increases. It is due to higher transmission energy needed in task offloading. Energy consumption in DCEMTO is 9.55% efficient than Random-Allocation approach.

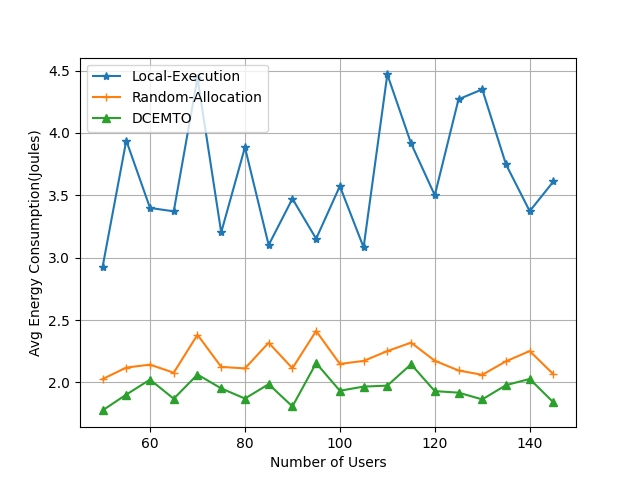


Figure Avg Energy Consumption vs Number of Users

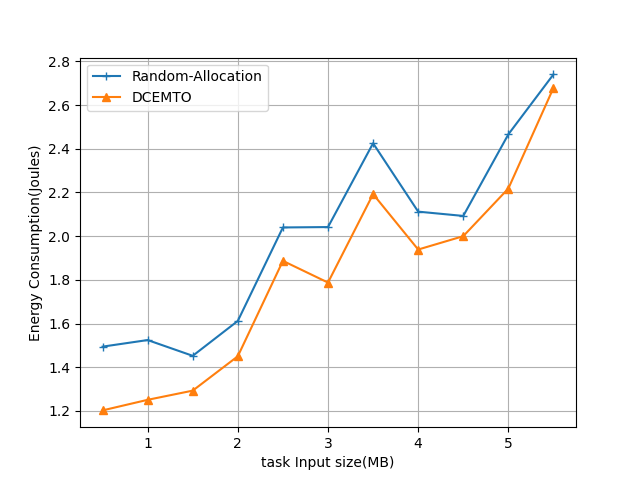


Figure Task Input Size vs Energy Consumption

Figure 8 depicts the difference in energy consumption with the change in MEC server’s computation capacity. Independent of server’s computation capacity, the energy consumption of DCEMTO is lesser compared to Random-Allocation approach. Energy consumption in DCEMTO is 8.22% efficient than Random-Allocation approach.

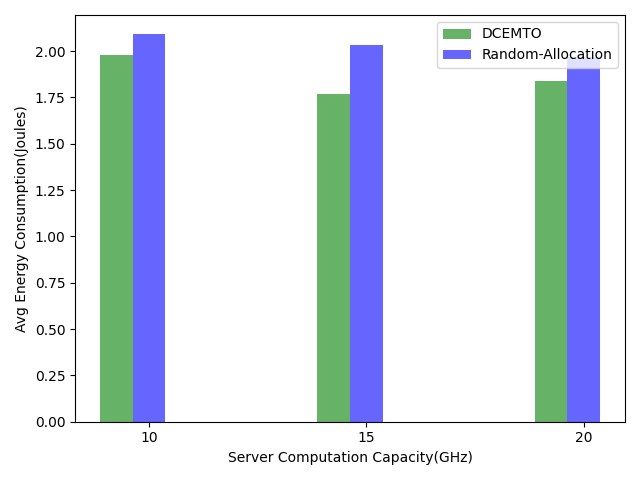


Figure Energy Consumption vs Server Computation Capacity

**CHAPTER 7**

# CONCLUSION

In our project, we explored the problem of Energy-minimal task offloading in the Mobile Edge Computing environment by considering the mobility of users and task deadline constraint. We considered our system model as a Mobile Edge Network (MEN) consisting of several Base Stations (BS), each equipped with a MEC server which is capable of receiving executing and transmitting computation tasks offloaded by users within the signal range of BS. We formulated the problem as an Energy Minimization problem with task constraints and proposed a heuristic algorithm DCEMTO, where user mobility, task deadline and base station information are utilized to optimize the overall energy consumption for task execution.

We evaluated our algorithm on Melbourne CBD base stations dataset consisting of Latitude and Longitude data of 126 Base stations situated in an area of 6.2 Km2. We compared our algorithm with baseline approaches like Random-Allocation and Local-Execution by referring to existing works for parameter settings. The results convey that our DCEMTO algorithm outperforms other approaches in metrics like Task Acceptance rate, Average Energy consumption and overall Energy Consumption. We performed the number of experiments to study how Energy Consumption varies by varying Number of users, MEC server capacity, Task Input size and observed that our algorithm is highly energy-efficient compared to other approaches. We also studied the variation in Task Acceptance rate with different Task Deadlines.

In our current work, we used the Random Waypoint model for the mobility of users. In future, we want to explore Machine Learning based methods like Generative adversarial network for users’ trajectories and use them to evaluate our approach.

# APPENDIX

## Random Waypoint Model

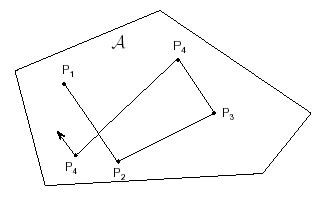
Random Waypoint (RWP) model is a commonly used synthetic model for mobility, e.g., in Ad Hoc Networks. In random-based mobility simulation models, the mobile nodes move randomly and freely without restrictions. To be more specific, the destination, speed and direction are all chosen randomly and independently of other nodes. It is an elementary model which describes the movement pattern of independent nodes by simple terms. This kind of model has been used in many simulation studies.

RWP model is elementary and it is easy to argue about the paths being unnatural. Then again, any practical protocol or mechanism should be robust and give a reasonable performance with a wide range of moving patterns, including movement similar to the RWP model.

The movement of nodes is governed in the following manner: Each node begins by pausing for a fixed number of seconds. The node then selects a random destination in the simulation area and a random speed between 0 (excluded) and some maximum speed. The node moves to this destination and again pauses for a fixed period before another random location and speed. This behaviour is repeated for the length of the simulation.

Briefly, in the RWP model:

* Each node moves along a zigzag line from one waypoint Pi to the next Pi+1.
* The waypoints are uniformly distributed over the given convex area, e.g. unit disk.
* At the start of each leg, a random velocity is drawn from the velocity distribution.
* (in the basic case the velocity is constant 1)
* Optionally, the nodes may have so-called "thinking times" when they reach each waypoint before continuing on the next leg, where durations are independent and identically distributed random variables.



# REFERENCES

1. J. Wang, W. Wu, Z. Liao, A. K. Sangaiah and R. Simon Sherratt, "An Energy-Efficient Off-Loading Scheme for Low Latency in Collaborative Edge Computing," in IEEE Access, vol. 7, pp. 149182-149190, 2019.
2. Y. Kai, J. Wang and H. Zhu, "Energy Minimization for D2D-Assisted Mobile Edge Computing Networks," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-6.
3. C. Wu, Q. Peng, Y. Xia and J. Lee, "Mobility-Aware Tasks Offloading in Mobile Edge Computing Environment," 2019 Seventh International Symposium on Computing and Networking (CANDAR), Nagasaki, Japan, 2019, pp. 204-210.
4. Z. Liu, X. Wang, D. Wang, Y. Lan and J. Hou, "Mobility-aware Task Offloading and Migration Schemes in SCNs with Mobile Edge Computing," 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 2019, pp. 1-6.
5. Zi Wang, Zhiwei Zhao, Geyong Min, Xinyuan Huang, Qiang Ni, Rong Wang, User mobility aware task assignment for Mobile Edge Computing, Future Generation Computer Systems, Volume 85, 2018, Pages 1-8, ISSN 0167-739X,
6. P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, “Optimal edge user allocation in edge computing with variable sized vector bin packing,” in International Conference on Service-Oriented Computing. Springer, 2018, pp. 230–245.
7. Q. Peng et al., "Mobility-Aware and Migration-Enabled Online Edge User Allocation in Mobile Edge Computing," 2019 IEEE International Conference on Web Services (ICWS), Milan, Italy, 2019, pp. 91-98.