A novel offloading framework for computation service in the Edge Cloud and the Core Cloud: A case study for face recognition



Nasif Muslim,
Id: 012 162 022

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Science in Computer Science and Engineering
United International University
Dhaka, Bangladesh

March 2019

Approval Certificate

This thesis titled "A novel offloading framework for computation service in the Edge Cloud and the Core Cloud: A case study for face recognition" submitted by Nasif Muslim, ID: 012 162 022, has been accepted as satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering.

Board of Examiners

Supervisor

Dr. Salekul Islam
Professor and Head of the Department
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

External examiner

Dr. A.K.M. Ashikur Rahman
Professor
Department of Computer Science and Engineering (CSE)
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000, Bangladesh

Ex-Officio

Dr. Md. Abul Kashem Mia
Professor and Dean
School of Science and Engineering (SoSE)
United International University (UIU)
Dhaka-1212, Bangladesh

Declaration

This is to certify that the work entitled "A novel offloading framework for computation service in the Edge Cloud and the Core Cloud: A case study for face recognition" is the outcome of the research carried out by me under the supervision of Dr. Salekul Islam, Professor and Head of the Department, Department of Computer Science and Engineering (CSE).

Signed: _____

Date: _____

Nasif Muslim, Id: 012 162 022

In my capacity as supervisor of the candidate's thesis, I certify that the above statements are true to the best of my knowledge.

Signed: _____

Date: _____

Dr. Salekul Islam, Head of the Department
Department of Computer Science and Engineering
United International University
Dhaka-1212, Bangladesh.

Abstract

A fast rate of progress has allowed the proliferation of smartphones and ease their extensive presence in people's daily life. However, inadequate processing speed, and limited battery capacity have hindered improvements of the computational capabilities of the smartphone. Offloading computational tasks to the remote Cloud (Edge and Core) could solve this problem by enabling the user to access these services over the Internet. Edge Cloud computing has been recognized as an emerging field within the Cloud computing paradigm, where computation servers are situated at the edge of the Internet to reduce network delay and traffic. Nevertheless, offloading tasks to the remote Cloud is not always beneficial due to variable network conditions and added processing costs.

In this thesis, a framework is proposed that provides smartphones with the ability to make offloading decisions to minimize processing time or cost, energy consumption or any combination of these three parameters. To validate the accuracy of the framework, face recognition is used as an application and is implemented in the remote Cloud infrastructures. Experimental results based on real-life and alternate scenarios demonstrate that the framework can make the correct offloading decision, in a cost effective way.

In memory of Welsh poet and writer Dylan Thomas (1914 - 1953)

Do not go gentle into that good night

Acknowledgements

I would like to express gratitude to my supervisor Prof. Dr. Salekul Islam, Head of the Department, Department of Computer Science and Engineering (CSE), United International University (UIU) for providing me the opportunity to conduct my master's thesis. His spontaneous help, encouragement and advice has guided me throughout the development period of this work.

I would like to acknowledge with much appreciation to Prof. Dr. A.K.M. Ashikur Rahman, Department of Computer Science and Engineering (CSE), Bangladesh University of Engineering and Technology (BUET) for his review comments to improve the quality of the manuscript.

I would like to thank Prof. Dr. Md. Abul Kashem Mia, Dean, School of Science and Engineering (SoSE), United International University (UIU) for his comments and suggestions.

I would like to thank Prof. Dr. Mohammad Nurul Huda, Director - MSCSE Program, Department of Computer Science and Engineering (CSE), United International University (UIU) for his advice.

I would also like to thank Associate Prof. Dr. Swakkhar Shatabda, Undergraduate Program Coordinator, Department of Computer Science and Engineering (CSE), United International University (UIU) for his cooperation and support.

Finally, I am indebted to my mother and my brother for their support, help, and love.

Contents

List of Figures x

List of Tables xii

1 Introduction 1

1.1 Research Motivation and Objectives 1

1.2 Thesis Contributions 4

1.3 Thesis Organization 4

2 Related Work 6

2.1 Cloud computing 6

2.2 Edge Cloud computing 7

2.3 Offloading technique 8

3 A proposed offloading framework 11

3.1 Computation service offloading framework 11

3.1.1 Resource profiling 11

3.1.2 Estimate values using regression analysis 12

3.1.3 Multi-Criteria Decision Making (MCDM) 13

4 Face recognition system 14

4.1 Preamble 14

4.1.1 The local face recognition system 14

4.1.2 The remote face recognition system 15

4.2 Local and remote processing simulation with iFogSim 15

4.2.1 Simulation result 17

5 Mathematical model to estimate the processing time, the energy consumption of the smartphone and the processing cost **20**
5.1 Mathematical model to estimate the processing time 20
5.1.1 Empirical measurement of the local processing time 22
5.1.2 Developing mathematical model for the local processing time 22
5.1.3 Empirical measurement of the remote Cloud processing time 23
5.1.4 Developing mathematical model for the remote Cloud processing time 25
5.2 Mathematical model to estimate the energy consumption of the smartphone . 25
5.2.1 Empirical measurement of the smartphone energy consumption for the local processing 26
5.2.2 Developing mathematical model for energy consumption of the smartphone for the local processing 27
5.2.3 Empirical measurement of smartphone energy consumption for the remote Cloud processing 28
5.2.4 Developing mathematical model for energy consumption of the smartphone for the remote Cloud processing 29
5.3 Mathematical model to estimate the processing cost 30

6 Offloading decision for service selection **31**
6.1 Preamble 31
6.2 Sensitivity analysis 32
6.3 Simple Additive Weighting (SAW) 32

7 Validation of the proposed offloading framework **37**
7.1 Preamble 37
7.2 Edge Cloud (UIU data center) and Core Cloud (Amazon Web Service t2.xlarge instance) 39
7.3 Edge Cloud (UIU data center) and Core Cloud (Microsoft Azure B4ms instance) 41

8 Conclusion **44**

Bibliography **46**

A Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) **52**

A.1 Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) . 52

List of Figures

2.1 Different types of Cloud services 6
2.2 Edge Cloud architecture [1] 7

3.1 The proposed architecture of the computation offloading framework 11
3.2 Steps of MCDM 12

4.1 The local face recognition system 15
4.2 The remote Cloud face recognition system 16
4.3 Sequence diagram showing face image transfer from Smartphone to remote Cloud for recognition 16
4.4 Network topology used for simulation experiment 18
4.5 Face recognition application model deployed in the iFogSim 18
4.6 Network delay measurement 19
4.7 Energy consumption of the smartphone measurement 19

5.1 Face recognition processing time in the smartphone (1.3 GHz quad-core processor) with a different face database size 22
5.2 Face recognition processing time in the smartphone (1.4 GHz quad-core processor) with a different face database size 23
5.3 Polynomial regression relationship (of the 2nd order with SSE = 1.336, R-squared = 0.9901) between the number of faces, face image database size and face recognition processing time in the smartphone (1.3 GHz quad-core processor) 24
5.4 Polynomial regression relationship (of the 2nd order with SSE = 0.0009569, R-squared = 1) between the number of faces, face image database size and face recognition processing time in the smartphone (1.4 GHz quad-core processor) 24

5.5 Computation time T_{server} for face recognition in the remote Cloud with a different face database size 25
5.6 Polynomial regression relationship (of the 1st order with SSE = 0.2309, R-squared = 0.9135) between the number of faces, face image database size and computation time T_{server} for face recognition in the remote Cloud 26
5.7 Energy consumption of the smartphone (1.3 GHz quad-core processor) with a different face database size for face recognition 27
5.8 Energy consumption of the smartphone (1.4 GHz quad-core processor) with a different face database size for face recognition 27
5.9 Polynomial regression relationship (of the 2nd order with SSE = 3.618×10^{-28}) between the number of faces, face image database size, and energy consumption of the smartphone (1.3 GHz quad-core processor) for face recognition 28
5.10 Polynomial regression relationship (of the 2nd order with SSE = 9.079) between the number of faces, face image database size, and energy consumption of the smartphone (1.4 GHz quad-core processor) for face recognition 29
5.11 Energy consumption of the smartphone for face recognition in the remote Cloud (Edge and Core) with a different face database size 29
5.12 Linear regression relationship between the number of faces, face image database size, and energy consumption of the smartphone for remote Cloud processing 30

6.1 A random set of data are generated using the Monte Carlo method 33
6.2 Estimate the preference value of each choice by SAW 33
6.3 Estimate the preference value of each choice by TOPSIS 34
6.4 Ranking of choices by SAW 34
6.5 Ranking of choices by TOPSIS 35

List of Tables

4.1 Experimental configuration of iFogSim for *face recognition* application 17

5.1 Hardware specifications of the smartphone 21
5.2 Hardware specifications of the Cloud 21

6.1 Sample data set for the face recognition system with 1000 face database and number of face to be recognized is 2 32
6.2 RSD of each choice 35

7.1 Resource profile 38
7.2 The weight of each criterion is assigned to five different cases 39
7.3 Framework decision 42
7.4 The actual measurement value for local processing in the smartphone with 1.4 GHz quad-core processor, remote processing in Edge Cloud (UIU data center) and Core Cloud (AWS) 43
7.5 The actual measurement value for local processing in the smartphone with 1.4 GHz quad-core processor, remote processing in Edge Cloud (UIU data center) and Core Cloud (Azure) 43

Chapter 1

Introduction

Smartphones are changing human and society life at breakneck speed with relentless, cutting-edge innovations, the ability to connect with the Internet using various wireless technologies at ever-increasing rates, and mass production-induced affordability. It is estimated that, worldwide, the number of smartphone users will reach 2.5 billion in 2019 [2]. By 2018, over 36 percent of the world's population is projected to use these devices. Smartphones bring the power of computing to the user on the move. For example, a user can capture a photo or video using built-in sensors and camera, edit and upload it on the Internet, play video games, navigate using GPS, listen to built-in FM radio or audio files or manage their personal health (e.g., iPhone Heart Monitor [3]), on top of phone calls and message exchanges. A smartphone can handle these intensive computing tasks by using ever more sophisticated and powerful processors. In this thesis, we use the term smartphone and mobile (phone) interchangeably.

1.1 Research Motivation and Objectives

Following Moore's law, smartphones' processor and connectivity speed have been increasing significantly over the years while staying at the same price point. However, battery technology has not been keeping up as well with the rapid growth of power consumption of the smartphone in terms of increase in energy capacity (i.e., milliampere-hour) of the ubiquitous lithium-ion batteries. To alleviate this problem to some degree, researchers have adopted a variety of methodologies and techniques such as smart batteries, energy-efficient graphical user interface, power efficient communication, proxy-assisted energy saving, etc., [4] to

reduce the amount of energy drained from the battery. Yet computing in general remains energy-expensive.

Task offloading to the remote Cloud (Edge and Core) is a promising solution to overcome many of the limitations of the smartphone with the help of improved network infrastructures. Core Cloud refers to the commercial Cloud computing infrastructures situated further away from the client or user, which can be accessed through the Internet core. Edge Cloud refers to commercial or private Cloud computing infrastructures located in closer proximity to the client or user, near or within the wireless access network.

Energy consumption can be reduced by migrating computation intensive tasks from the mobile to the Cloud [5] [6]. Moreover, by offloading such tasks to the Cloud, these devices can harness the superior computational power of a back-end server [7]. Note, however, that offloading a task to the remote Cloud (Edge and Core) is not always beneficial. For example, processing in the Cloud might take a longer time with compared to the processing in the smartphone when communication delays are taken into account. Additionally, it will increase processing cost. The scenario becomes more complex when we consider the fact that the Edge Cloud is situated nearer to the user, which reduces network delay. Therefore, it is essential to make the right decision to make offloading, which incidentally can also be influenced by user preferences, energy-favourable to the smartphone. We identify here four factors that will be impacted by offloading:

- **Processing time**: We distinguish between local, remote and network processing time. The latter includes not only communication delays, but also the overhead of packing/unpacking requests and answers. In other words, we consider that it includes residual local processing. Note also that we assume that the remote resources are immediately available.

  Processing time can be reduced by offloading if the local processing time is higher than the combination of network and remote processing time [8]. As a rule of thumb, remote processing in the Cloud (Edge and Core) is practical for larger computational tasks. However, unfavourable network conditions could create an adverse effect.

- **Energy consumption**: It can be (globally) reduced by offloading if the energy consumption due to local processing is higher than the remote processing [9]. Energy consumption of the smartphone is an important aspect affecting the offloading decision, as local processing increases battery usage. Remote processing in the remote

Cloud (Edge or Core) reduces the smartphone's energy consumption independently of any consideration on processing time.

- **Processing cost**: Generally, while local processing is free, end users use price as their primary criteria to select a Cloud service provider [10]. Most of existing Cloud service providers such as Amazon Web Services (AWS) [11], Microsoft Azure [12] and Google Cloud [13] use a fixed price model where resource usage is charged at a fixed unit. In this thesis, processing cost defines the resource usage cost of the remote Cloud.

- **All of the above**: When the aim of the user is to save on any combination of these three parameters (i.e. processing time, processing cost and energy consumption) then the decision to offload requires trading off one parameter against other. For example, the processing time could be decreased if a task is processed locally in the smartphone, but it may increase battery consumption. Similarly, remote execution in the Cloud (Edge and Core) could save on the smartphone's energy consumption, to the detriment of processing cost. The trade-off between these three parameters can be considered as a multi-objective optimization problem, and has been treated as such in the literature.

Additionally, the types of application and hardware configuration influence greatly the decision to offload. Kumar et al. [14] suggests that offloading a computation task to the Cloud is not beneficial for every applications. The experimental result shows that computation offloading to the Cloud is more beneficial in the case of a *chess game* application when compared with an *image retrieval* application due to the high amount of computation required by the former. In this thesis, we carry out an in-depth study in *offloading* and how users' preference parameters affect one another. Furthermore, a framework is proposed which establishes the decision to offload based on resource profile information (e.g., processor speed, bandwidth, round trip time (RTT) and computation cost) and user preferences. We consider *face recognition* as a computation- and data-intensive application, which we use to evaluate the soundness of the offloading framework. It should be noted that this framework has been studied for a specific hardware configuration of Cloud (Edge and Core) and smartphone. However, with proper modification, it can be adapted to other hardware configurations as well.

1.2 Thesis Contributions

The main contributions of this thesis are as follows:

- A simulator tool, iFogSim [15] is used to model the *face recognition* application and its deployment in the smartphone and in the Cloud (Edge and Core). With the help of this simulator, we study and compare local processing in the smartphone and remote processing in the Cloud (Edge and Core) with respect to network delay and energy consumption of the smartphone. The simulation results show that local computation can reduce the network delay. However, it will increase the energy consumption of the smartphone.
- Mathematical models have been developed to estimate the processing time, the energy consumption of the smartphone and the processing cost for the *face recognition* application.
- Next, using sensitivity analysis [16] we have shown that Simple Additive Weighting (SAW) [17], a Multi-Criteria Decision Making (MCDM) method, is the preferred technique to decide when it is worth to offload computation for the *face recognition* from local processing to remote processing.
- Most of the research has focused on one or two parameters for offload decision making [18]. Here, any combination of these three parameters (i.e. processing time, energy consumption of the smartphone, processing cost) is considered. Using the experiments, we show that the proposed framework can realize savings in either one or in any combination of these parameters.

1.3 Thesis Organization

The thesis is organized as follows:

Chapter 2 presents an overview of previous work related to Cloud computing, Edge Cloud computing and offloading frameworks.

Chapter 3, the architecture of the proposed offloading framework is discussed.

Chapter 4 explains the local and remote face recognition system. Additionally, the simulation results of the face recognition application in terms of network delay and energy consumption for local processing and remote processing are discussed.

Chapter 5 explains the mathematical model to estimate the processing time, the energy consumption of the smartphone and the processing cost, respectively.

Chapter 6 explains how to determine the appropriate MCDM method for decision making to offload the computation task.

Chapter 7 the experimental results on multiple real-life scenarios are presented to validate the proposed framework.

Chapter 8 describes future avenues of expansion of our framework.

Chapter 2

Related Work

This chapter briefly provides background and literature review on Cloud computing, Edge Cloud computing and different computation offload techniques. Section 2.1 and section 2.2 presents the review of the key concepts, that is, Cloud computing and Edge Cloud computing. Related research work in different computation offload techniques are presented in section 2.3.

2.1 Cloud computing

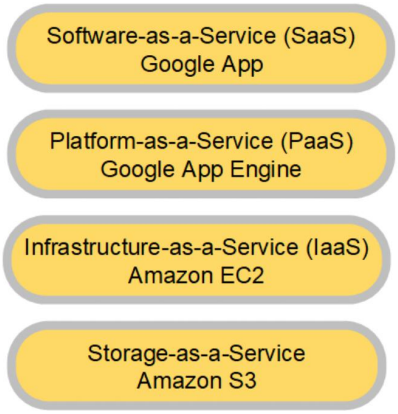


Figure 2.1: Different types of Cloud services

The emergence of Cloud computing was made possible due to large corporate data centres, increased availability of high speed network connectivity and the decreasing cost of storage. Figure 2.1 shows the types of services provided by the Cloud. Storage as a Service includes

Internet-based storage. Infrastructure as a Service (IaaS) includes on-demand computing infrastructures, so that a user can run any software or operating system. Platform-as-a-Service (PaaS) includes a full or partial application development environment that users can access through tools provided by the Cloud service providers. Software-as-a-Service (SaaS) allows the user to use the software in the Cloud either as a subscription based service or in a pay-as-you-go model.

Through server virtualization technology, Cloud computing provides application software, platform or infrastructure as a reliable service [19]. For example, Amazon EC2 [11], and the Rackspace Cloud [20] can support multiple virtual machines (VM) instances on a single physical server by using Xen virtualization technology. Similarly, AT&T Cloud Solutions [21] has used VMware virtualization technology [22].

2.2 Edge Cloud computing

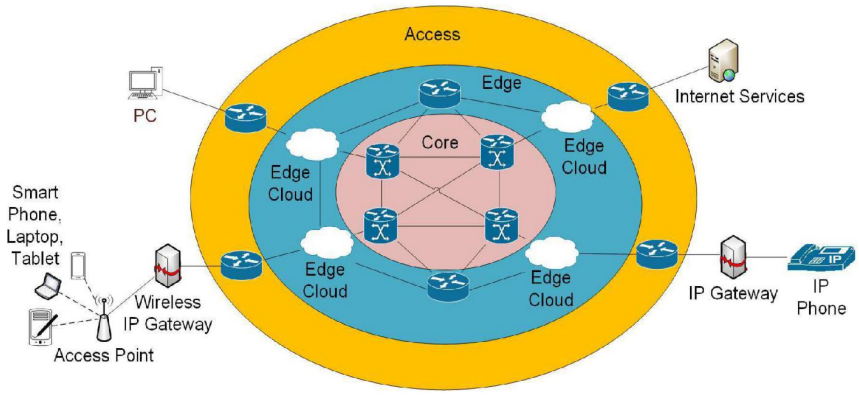


Figure 2.2: Edge Cloud architecture [1]

The Edge Cloud consists of multiple smaller, generic clouds deployed at the edge of the network, in co-operation with the ISPs. The simplified Edge Cloud architecture is presented in figure 2.2. The advantage of the Edge Cloud is that it decreases network latency, breaks service vendor lock-in and enables access to local content which in turn allows users to create and manipulate composite applications with enhanced security and development human-centred novel architecture and application [23]. It also supports client-side virtualization, allowing users to access Virtual Desktop Infrastructure (VDI) using desktops, laptops, and smartphones.

In our previous work, different applications including on-the-fly video transcoding [24], virtual P2P client [25] and face recognition [26], offline video transcoding [27] were deployed in the Edge Cloud to study its feasibility and performance.

Nasif et al. [26] have proposed to transfer the face recognition computation from the smartphone to the Edge Cloud. The experimental results show that face recognition in the Edge Cloud is indubitably faster than the smartphone. It also scales up better as the number of faces increases in the test image data set. Beck et al. [28] discussed several applications like file editing, video streaming, and messaging for the deployment in the Edge Cloud and classified them based on technical metrics: power consumption, delay, bandwidth usage, and scalability. Hu et al. [29] proposed to deploy Cloud servers at the edge of the mobile network to reduce the latency to ensure a highly efficient network operation. Satyanarayanan et al. [30] proposed an architecture in which a smartphone user instantiates customized service software on a nearby Cloudlet by utilizing virtual machine (VM) technology. A Cloudlet is a trusted, sever or cluster of servers with superior hardware configuration, which is available to provide service to smartphone users via the Internet. It will liberate smartphone devices from severe resource constraints. Additionally, it can provide fast request-response between smartphone and Cloudlet due to the Cloudlet's one-hop physical proximity. Soyata et al. [31] proposed a smartphone-Cloudlet-Cloud architecture called *MOCHA* as a platform for real-time face recognition. The Cloudlet receives the computation task from the smartphone and partitions the task among different Cloud service providers which are distributed over different geographical locations. The simulation result shows that the overall processing delay is reduced as the number of Cloud server increases.

2.3 Offloading technique

Offloading of computational tasks frameworks can be classified into two categories based on how the decision to offload is made: either statically or dynamically. The static offloading framework uses a prediction model that utilizes offline network and hardware profiling information.

The static method has the advantage of a low overhead in terms of cost analysis at runtime. Kosta et al. [32] presents a framework *ThinkAir* for offloading computation to the Cloud using smartphone virtualization. It allows on-demand resource allocation and parallelism to increase elasticity and scalability by dynamically creating, resuming, and destroying multiple

virtual machine (VM) images in the Cloud. The decision to offload is based on the prediction of local vs. remote processing energy consumption, which is influenced by the historical values retrieved from previous executions. As a result, the prediction for remote execution energy consumption will be accurate if stable network conditions are maintained. Kemp et al. [33] presents a framework, named *Cuckoo*, which allows Android-based smartphones to offload computation to the Cloud. The decision is based on the network reachability. The objective of this framework is to enhance performance and reduce battery usage. Zhang et al. [34] proposed energy-efficient computation offloading (EECO) algorithm which minimizes the energy consumption of the offloading system under the delay constraints. The simulation results show that EECO is able to lower the energy consumption by up to 15% when compared to the computation without offloading. Liu et al. [35] proposed a computation offloading algorithm based on game theory in vehicular edge networks. It reduces the latency of the computation offloading of vehicles by minimizing computation overhead to select a proper wireless channel for each vehicle.

For the dynamic method, a prediction model exploits real-time network and hardware profiling information, which thus significantly increases the accuracy of the offloading decision. Cuervo et al. [36] presents an energy-aware offloading framework, named *MAUI*. It dynamically decides which section of the program should be offloaded for remote execution to maximize energy saving based on the current network conditions. However, it lacks the capability to predict energy consumption in the case of local and remote processing. Kovachev et al. [5] presents an offloading framework, named Mobile Augmentation Cloud Services (MACS), which allows for Android application partitioning and provides a mechanism for dynamic computation offloading. The decision to partition is transformed into an optimization problem based on the smartphone and Cloud parameters such as CPU load, available memory, remaining battery power on devices, network bandwidth between the smartphone and the Cloud. The main improvement of MACS over *Cuckoo* and *MAUI* is that it allows dynamic partitioning, does extra profiling, and resource monitoring of applications. Namboodiri et al. [37] proposes an analytical model to generalize the energy consumption of the smartphone for executing an application locally in the smartphone and remotely in the Cloud. Based on this model, a decision is made dynamically for local or remote processing. Karim et al. [38] proposes an algorithm to decide to offload a computation from the smartphone to the Cloud based on network conditions, the status of the battery and user inputs. Zhou et al. [39] presents a context-aware offloading framework for the heterogeneous mobile cloud which

consists of mobile devices, nearby cloudlets, and public clouds to improve the performance and availability of mobile cloud computing services. It proposes a context-aware algorithm to make offloading decisions at runtime based on the selection of wireless medium and appropriate cloud resources. Lin et al. [40] proposed a Context-Aware Decision Algorithm (CADA) to improve the accuracy of the offloading framework. The decision engine makes periodic measurements real-time signal strength, transmission time, geographical location to decide at each time slot on the opportunity to offload.

Again we distinguish our work from these earlier investigations by its focus on the use of a combination of three parameters (processing time, energy consumption of the smartphone, processing cost) to decide to offload.

Chapter 3

A proposed offloading framework

This chapter explains the architecture of the proposed computation service offloading framework. Each module of the framework is explained in subsection 3.1.1, 3.1.2 and 3.1.3.

3.1 Computation service offloading framework

The proposed architecture of the system to offload computation services is shown in Figure 3.1. There are three modules: resource profiling, estimate values using regression analysis and multi-criteria service selection. The functions of each module are explained in the following subsections.

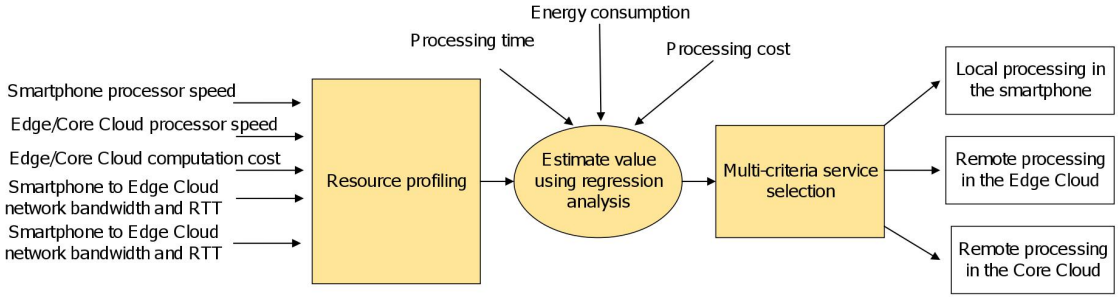


Figure 3.1: The proposed architecture of the computation offloading framework

3.1.1 Resource profiling

The resource profiling module is used to gather information about (i) the processor speed of the smartphone, Edge Cloud and Core Cloud, respectively, (ii) computation costs in Edge

and Core Cloud, respectively, and (iii) bandwidth and RTT between smartphone and Edge Cloud and Core Cloud networks, respectively. Collecting this information with timeliness and accuracy is important to make the correct decisions.

3.1.2 Estimate values using regression analysis

Regression analysis is used for predictive modelling: a prediction equation can be used to predict the value of a dependent variable once regression coefficients are obtained, by fitting a line or curve to measurements. For the multi-variable polynomial regression model, the relationship between the dependent variable with one or more independent variables is a curve. Equation 3.1 shows the general format of the multi-variable polynomial regression equation for prediction.

$$z = a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_4x_2^2 + ... + a_nx_n^n \tag{3.1}$$

where, x: independent variables and a: regression coefficients for the independent variables.

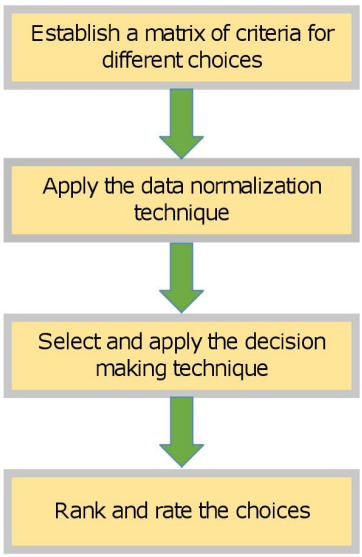


Figure 3.2: Steps of MCDM

The least-square method [41] is used to find the best values of the regression coefficients. In this thesis, we use the sum of squared errors of prediction (SSE), that is, the sum of deviations predicted from the actual empirical values of data, as a measure a quality of our estimate:

the smaller the value, the better the fit of the regression model to the data. Additionally, we use R-Squared, that is, variance of the predicted values divided by the variance of the data to measure the quality of the estimated regression model. R-squared values range from 0 to 1 where values closer to 0 represent a poor fit while values closer to 1 represent a perfect fit of the regression model to the data.

3.1.3 Multi-Criteria Decision Making (MCDM)

MCDM is an important tool for decision making due to its inherent ability to compare different choices based on various criteria [42]. This explicit procedure reduces the number of arguments or conflicts involved. Figure 3.2 shows the steps of MCDM. In the first step, a list of alternatives or choices is prepared and a matrix of criteria for each choice is established. These criteria are captured as sets of independent attributes, which are measured by different units. Next, a data normalization step [43] transforms each criterion into a compatible measurement unit. In the third step, the decision-making technique is selected and applied. Finally, the choices are ranked to identify the best choice.

Chapter 4

Face recognition system

This chapter describes the face recognition system which has been used to evaluate the offloading framework in the section 4.1. Furthermore, the simulation result for local and remote processing with IFogsim is discussed in section 4.2.

4.1 Preamble

Face recognition system consists of mainly two phases: *face detection* and *face recognition*. Human face detection is a complex and challenging task due to the numerous variations of the background, light and a large diversity of facial expressions in images. In the *face detection phase*, the potential location of the face is identified within an image. Google Mobile Vision API [44] is used for face detection. The *face recognition phase* compares the selected face image with stored images in the face image database to find a match. The face recognition application is deployed in the smartphone and the remote Cloud to measure the processing time, the energy consumption of the smartphone and the processing cost. Local binary pattern (LBP) algorithm [45] is used for face recognition. The face image is divided into small regions to extract LBP histograms and concatenated into a single feature vector of that region. These feature vectors form the efficient representation of the face, which can be used to measure the similarity with the selected face image.

4.1.1 The local face recognition system

In the smartphone, the face recognition classifier is first trained using a face dataset stored in the smartphone or from a face recognition database file *facedata.xml* where features, extracted

from the face dataset are stored. Then, the face recognition system takes a photo of a single person or multiple persons using the mobile's camera and starts the face detection process. Ultimately, such recognition can be used for multiple purposes, such as identification to retrieve individuals' personal information like a person's name. When the face image is recognized then the face is marked on the photo with a rectangle and labelled with the *person's name*. Figure 4.1 shows the block diagram of the local face recognition system.

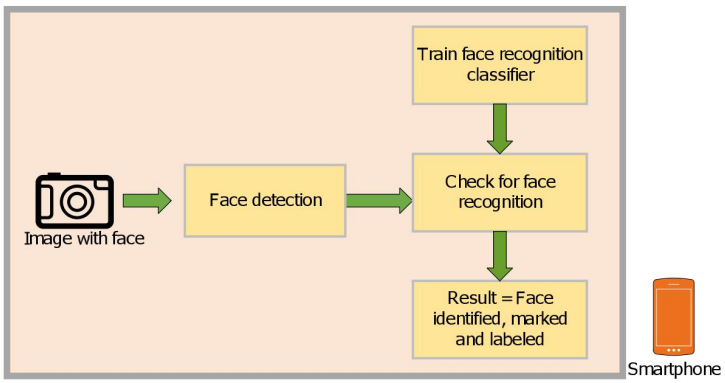


Figure 4.1: The local face recognition system

4.1.2 The remote face recognition system

Rather than processing the face recognition locally, the smartphone could send single or multiple face(s) selected from the image to the remote Cloud. The face recognition classifier is trained in the remote Cloud similarly to the process above. After receiving the face image from the smartphone, it performs the face recognition and sends back individuals' personal information such as a *person's name*. Then, the mobile will label the selected photo with the *person's name*. Figure 4.2 shows the block diagram of the remote Cloud face recognition system. Figure 4.3 shows the sequence diagram showing face image transfer from smartphone to remote Cloud for recognition.

4.2 Local and remote processing simulation with iFogSim

iFogSim [15] is an open source simulation tool written in Java for the evaluation of resource-management techniques and scheduling policies across Edge Cloud and Core Cloud under

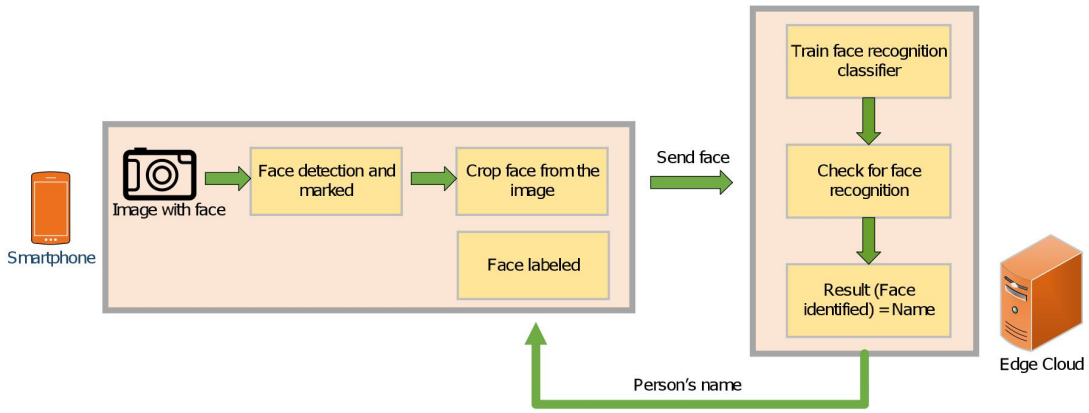


Figure 4.2: The remote Cloud face recognition system

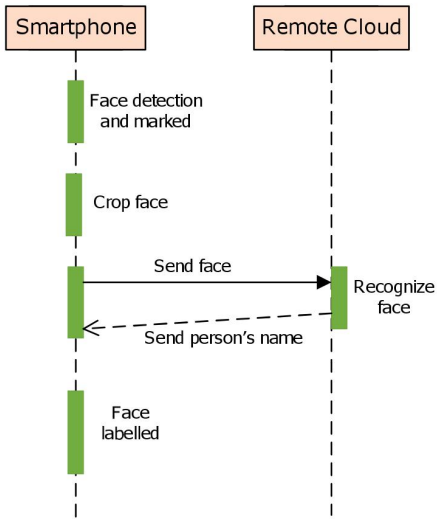


Figure 4.3: Sequence diagram showing face image transfer from Smartphone to remote Cloud for recognition

multiple scenarios. In this simulator, *face recognition* is used as an application to evaluate network delay, energy consumption for local processing in the smartphone and remote processing in the Edge Cloud and Core Cloud.

To start the iFogSim simulation, the first step is to create hardware components with parameters which includes processing capability in million instructions per second (MIPS), volume of RAM, uplink and downlink network bandwidth. Table 4.1a provides the configuration information of different pieces of hardware in the simulation. Figure 4.4 shows the

Table 4.1: Experimental configuration of iFogSim for *face recognition* application

(a) Hardware configuration

Hardware	CPU (MIPS)	RAM (GB)	Network bandwidth uplink (Mbps)	Network bandwidth downlink (Mbps)
Smartphone	2470	1	15	20
Edge Cloud	75000	16	30	30
Core Cloud	75000	16	100	100

(b) Inter-module tuples configuration

Tuple type	CPU count (MIPS)	Data length (byte)
PHOTO	15	262144
_face	50	10240
_name	100	8
_result	40	262144

(c) Network link delay configuration

Source	Destination	Latency (ms)
Camera	Smartphone	0.1
Smartphone	Edge Cloud	6.5
Smartphone	Core Cloud	60
Smartphone	Display	0.1

network topology used for the experiments. The next step is to add application-related logical components such as AppModule and AppEdge. The AppModule consists of three modules: *face_detector*, *face_recognizer* and *result*, while the AppEdge objects include information regarding tuple's type, CPU, networking length and latencies between different devices of the simulator (Table 4.1b, 4.1c). Tuples form the fundamental unit of communication between entities in iFogSim simulator, and are indicated over the edges in figure 4.5.

4.2.1 Simulation result

Figure 4.6 shows the network delay measurement for *face recognition* application local and remote processing. The remote processing time is significantly higher compared to the local processing in the smartphone. Furthermore, remote processing in the Core Cloud has even longer processing delay than Edge Cloud due to the greater network link latency to the Core

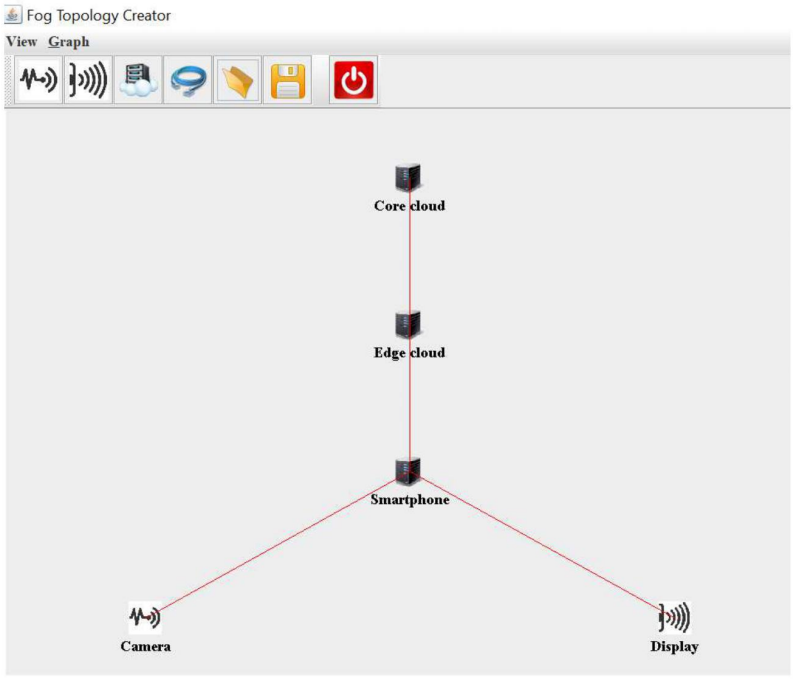


Figure 4.4: Network topology used for simulation experiment

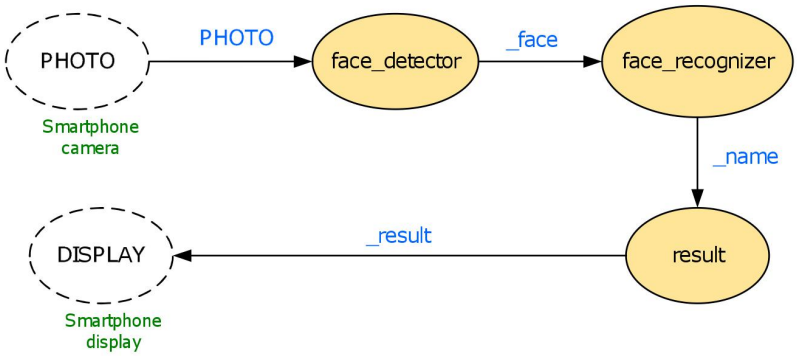


Figure 4.5: Face recognition application model deployed in the iFogSim

Cloud. Figure 4.7 shows smartphone energy consumption for local and remote processing. It shows that more energy is consumed for local processing compared to remote processing.

It is clear from the simulation's results that it is optimal to execute face recognition application locally in the smartphone considering network delay. However, it is also more energy consuming.

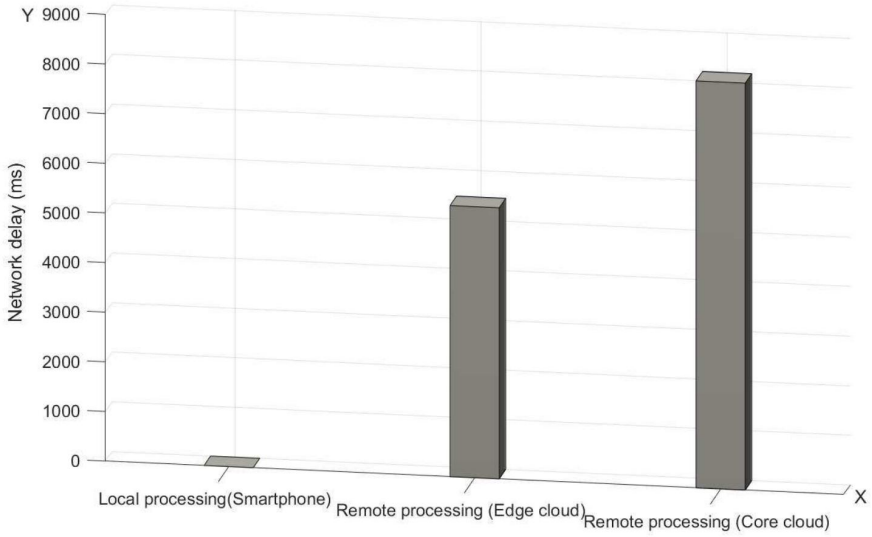


Figure 4.6: Network delay measurement

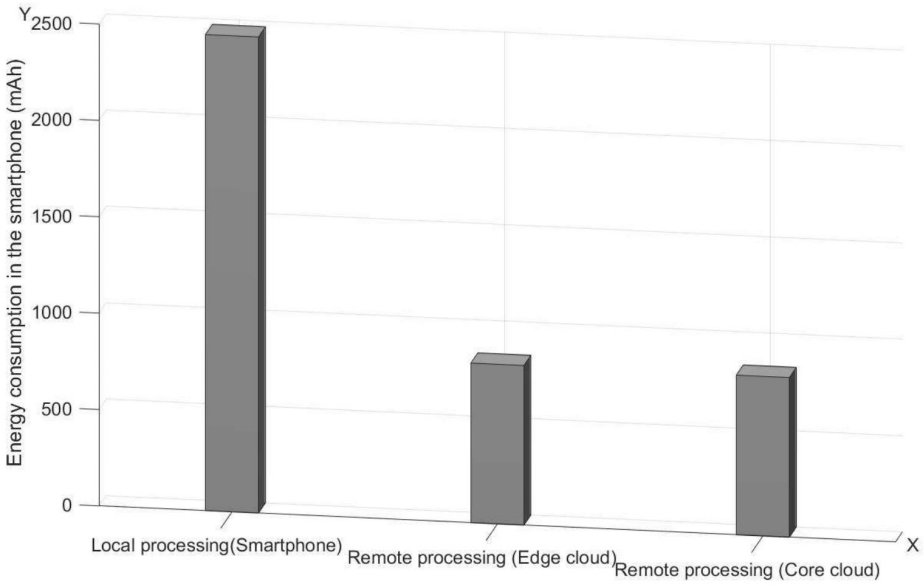


Figure 4.7: Energy consumption of the smartphone measurement

Chapter 5

Mathematical model to estimate the processing time, the energy consumption of the smartphone and the processing cost

This chapter presents the mathematical model to estimate the processing time (section 5.1), the energy consumption of the smartphone (section 5.2) and the processing cost (section 5.3).

5.1 Mathematical model to estimate the processing time

Accurate estimation of the processing time will enable the framework to make the right decision whether or not to perform task offloading. Estimating local processing time in the smartphone is straightforward because it solely depends on the mobile. However, it is always more challenging to estimate processing time in the remote Cloud since it depends on both computation time in the Cloud and network delays.

Our aim is to design an offloading framework as generic as possible. For this reason, we have conducted experimentation on Android smartphones with 1.3 GHz and 1.4 GHz quad-core processors (shown in Table 5.1). The Cloud (Edge and Core) hardware configuration is provided in Table 5.2. Subsection 5.1.1 and 5.1.3 explains the empirical measurement of the local and remote processing time. Mathematical models for estimating the local and the remote processing time are discussed in subsection 5.1.2 and 5.1.4 based on these empirical

measurement.

Table 5.1: Hardware specifications of the smartphone

	Model name	A	B
Hardware specification	Processor	1.3 GHz Quad core Cortex-A7	1.4 GHz Quad-core Cortex-A53
	RAM	1 GB	2 GB
	OS	Android 5.1 Lollipop	Android 7.1.2 Nougat
	Battery	2300 mAh	3120 mAh

Table 5.2: Hardware specifications of the Cloud

		Edge Cloud (UIU data center)	Core Cloud	
			Amazon EC2 t2.xlarge instance	Azure Standard B4ms instance
Hardware specification	Processor	Intel Xeon Processor E5-2630 V3 4 core @ 2.40 GHz	Intel Xeon Processor E5-2686 V4 4 core @ 2.30 GHz	Intel Xeon Processor E5-2673 V4 4 core @ 2.30 GHz
	RAM	16 GB	16 GB	16 GB
	OS	Windows 10	Windows Server 2012	Windows 10

Let us assume that $T_{L1.3}$ and $T_{L1.4}$ are the local face recognition processing times for the smartphones with 1.3 GHz and 1.4 GHz quad-core processors, respectively.

In case of remote face recognition, we can assume the following equation:

$$T_{R-edge/core} = T_{server} + T_{net} \tag{5.1}$$

Where, $T_{R-edge/core}$ = Remote face recognition processing time in the Cloud (Edge/Core),

T_{server} = Cloud (Edge/Core) computation processing time, and

T_{net} = Network delay for transmission of face image from smartphone to Cloud (Edge/Core) and transmission of the person's name from Cloud (Edge/Core) to smartphone.

Note that

$$T_{net} = d_{trans} \times n + d_{prop} \tag{5.2}$$

Where, d_{trans}, d_{prop} and n denote the transmission delay, the propagation delay and number of face to be recognized. For modern network equipment, the node processing delay

and queuing delay are insignificant. As a result, in this thesis, node processing and queuing delays are ignored.

5.1.1 Empirical measurement of the local processing time

Figures 5.1 and 5.2 show face recognition times by the Android smartphones with 1.3 GHz and 1.4 GHz quad-core processor. It is observed that the face recognition time increases as the number of faces and face database size increase. However, face recognition time decreases when the smartphone with 1.4 GHz quad-core processor is used due to the better processor speed. Therefore, it is evident that the processor speed, the number of faces and the face database size can impact on the face recognition time.

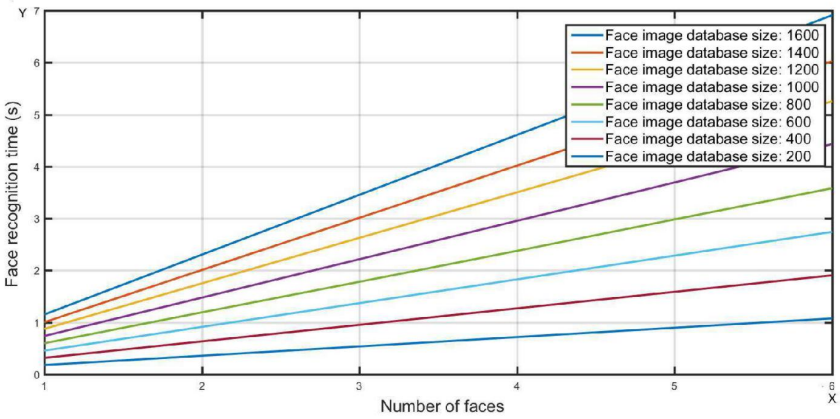


Figure 5.1: Face recognition processing time in the smartphone (1.3 GHz quad-core processor) with a different face database size

5.1.2 Developing mathematical model for the local processing time

From the empirical measurements of the previous sub-section, we can deduce that the multivariable polynomial regression model with the two predictor or independent variables (e.g., number of face and face image database size) and a dependent variable (e.g., face recognition time) can be understood as a two-dimensional surface in space while the processor speed is constant. Observations are points in space and the surface is *fitted* to best approximate the observations using the Matlab curve fitting toolbox [46]. The shape of the surface depends on the structure of the model. The figure 5.3 and 5.4 show the multivariable polynomial

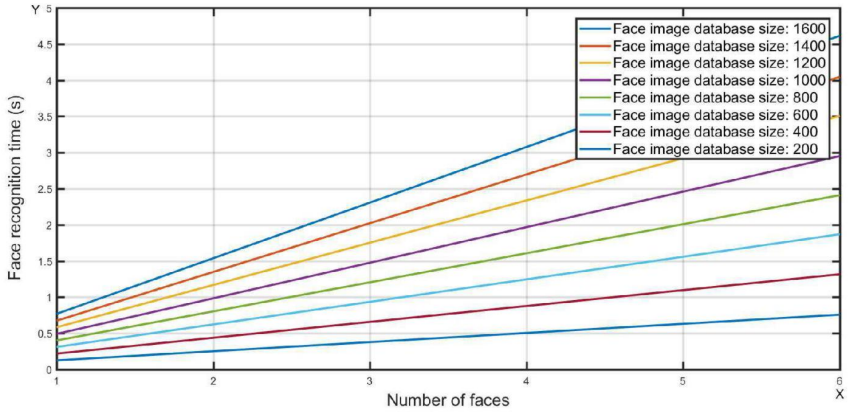


Figure 5.2: Face recognition processing time in the smartphone (1.4 GHz quad-core processor) with a different face database size

regression models between the number of faces, the face image database size and the face recognition time in the two smartphone (1.3 GHz and 1.4 GHz). The two-dimensional polynomial surfaces which are used to predict face recognition times for the two smartphones are shown in Equations 5.3 and 5.4.

$$\begin{aligned} T_{L1.3} =& 0.2111 + 0.01306x - 0.0006146y + 0.0007042xy \\ &+ 3.375 \times 10^{-7} y^2 \end{aligned} \tag{5.3}$$

$$\begin{aligned} T_{L1.4} =& -2.516 \times 10^{-16} + 0.0358x + 1.847 \times 10^{-19} y \\ &+ 5.205 \times 10^{-17} x^2 + 0.0004573xy \end{aligned} \tag{5.4}$$

5.1.3 Empirical measurement of the remote Cloud processing time

We measure the face recognition time in the Edge Cloud (UIU data center) only. Note that due to the similar hardware configuration of Edge and Core Clouds (shown in Table 5.2, this time will be similar in both cases. The computation processing time for face recognition T_{server} in the remote Cloud is significantly faster compared to the smartphone T_L due to the superior processor speed. For example, in 1000 face database, the remote Cloud takes 0.35 second while the smartphones with 1.3 and 1.4 GHz quad-core processors take 1.477 and 0.984 seconds, respectively. Figure 5.5 shows the computation time T_{server} for face recognition in the remote Cloud with a different face database size. Additional network delay T_{net} needs

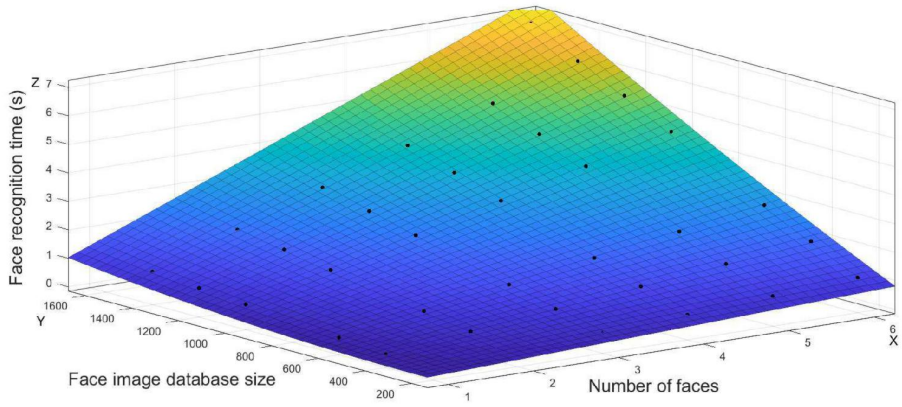


Figure 5.3: Polynomial regression relationship (of the 2nd order with SSE = 1.336, R-squared = 0.9901) between the number of faces, face image database size and face recognition processing time in the smartphone (1.3 GHz quad-core processor)

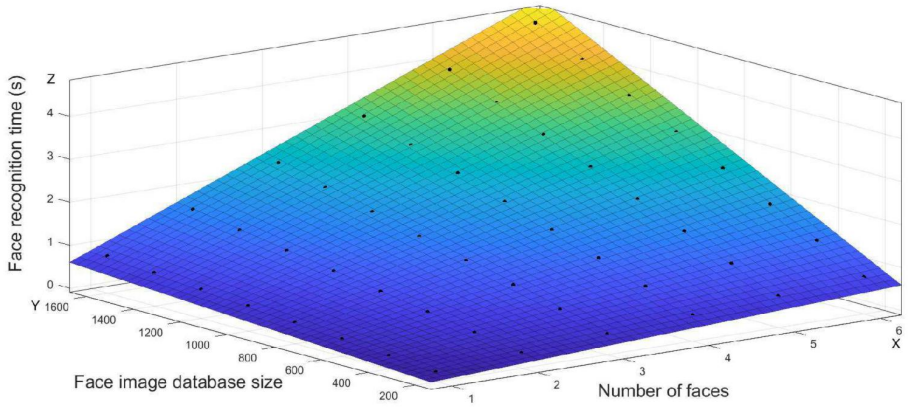


Figure 5.4: Polynomial regression relationship (of the 2nd order with SSE = 0.0009569, R-squared = 1) between the number of faces, face image database size and face recognition processing time in the smartphone (1.4 GHz quad-core processor)

to be considered for face recognition time calculation in the remote Cloud. Transmission delay d_{trans} calculation requires face image file size and network bandwidth value from the smartphone to the Cloud. Using IPerf3 [47] tool the uploading and downloading bandwidth of the network from the smartphone to Cloud is measured. The propagation delay d_{prop} can

be calculated from ICMP ping command.

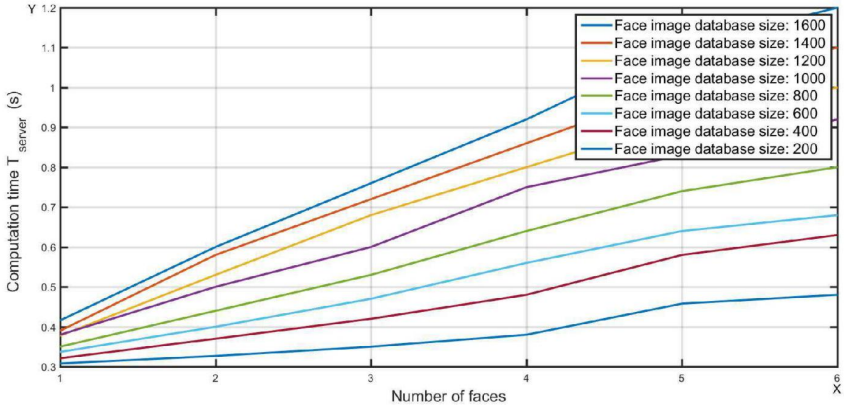


Figure 5.5: Computation time T_{server} for face recognition in the remote Cloud with a different face database size

5.1.4 Developing mathematical model for the remote Cloud processing time

Figure 5.6 shows the multivariable polynomial regression model between the number of faces, face image database size and computation time T_{server} for face recognition in the remote Cloud. The two-dimensional polynomial surface which is used to predict face recognition time in the remote Cloud is shown in Equation 5.5.

$$T_{server} = -0.02121 + 0.1002x + 0.0003198y \tag{5.5}$$

5.2 Mathematical model to estimate the energy consumption of the smartphone

Energy consumption of the smartphone for local or remote processing is an important parameter in the decision of offloading. *Batterystats* [48] is an open-source tool, which is used to estimate energy consumption by collecting battery data of the smartphone. It is included in the Android framework. *Battery Historian* [49] converts the report from *Batterystats* into a HTML that can be viewed in a web browser. Subsection 5.2.1 and 5.2.3 explains empirical measurement of the smartphone energy consumption for local and remote processing.

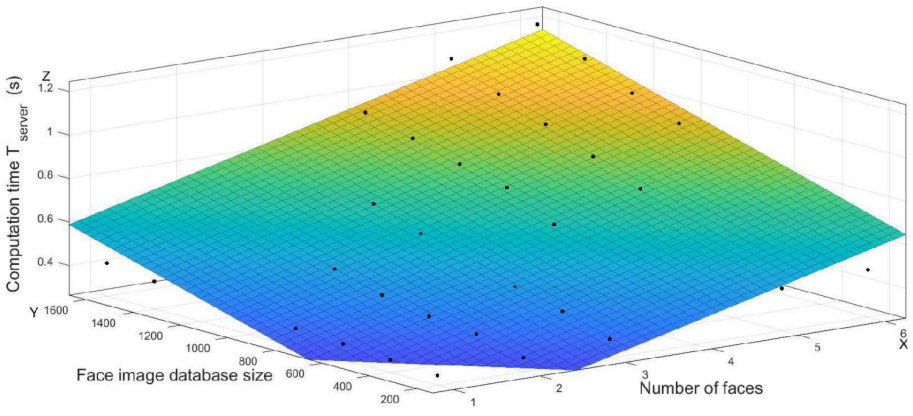


Figure 5.6: Polynomial regression relationship (of the 1st order with SSE = 0.2309, R-squared = 0.9135) between the number of faces, face image database size and computation time T_{server} for face recognition in the remote Cloud

Mathematical models are developed in subsection 5.2.2 and 5.2.4 based on these empirical measurement.

Let us assume that $E_{L1.3}$ and $E_{L1.4}$ denote the energy consumption of smartphone for local face recognition in the smartphone with 1.3 and 1.4 GHz quad-core processors, respectively. Also, $E_{R-edge/core}$ denotes energy consumption of smartphone for remote face recognition in the Cloud.

5.2.1 Empirical measurement of the smartphone energy consumption for the local processing

As for face recognition time, energy consumption of the smartphone is influenced by processor speed, number of faces and face database size. Figure 5.7 shows the energy consumption of the smartphone (1.3 GHz quad-core processor) with different face database size for face recognition. The energy consumption increases steadily as the number of faces and/or the size of the face database increase.

A similar trend is observed in Figure 5.8 which shows the energy consumption of the smartphone (1.4 GHz quad-core processor). However, the energy consumption of the smartphone with a 1.4 GHz quad-core processor than that of the 1.3 GHz quad-core processor's due to the higher processor speed [50]. For example, to recognize three faces in 1400 face

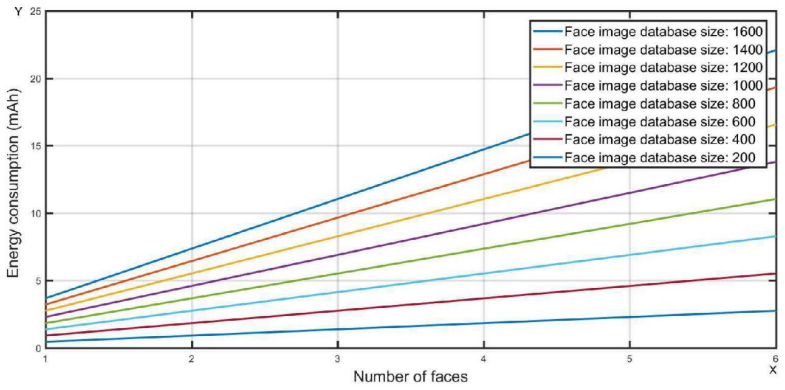


Figure 5.7: Energy consumption of the smartphone (1.3 GHz quad-core processor) with a different face database size for face recognition

database, energy consumption is 11.4 mAh in the case of the a 1.4 GHz quad-core processor while it is only 9.2 mAh in the other case.

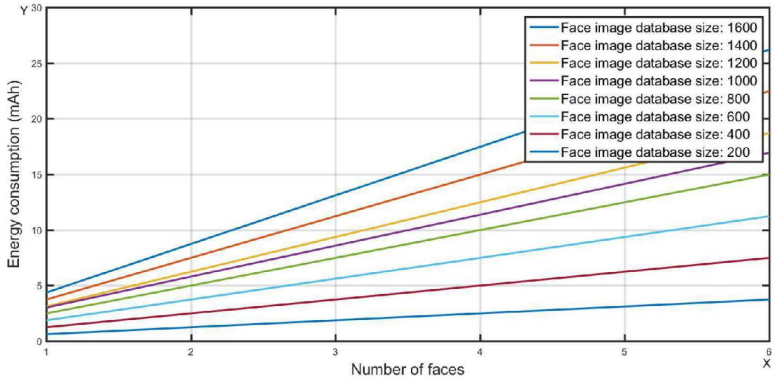


Figure 5.8: Energy consumption of the smartphone (1.4 GHz quad-core processor) with a different face database size for face recognition

5.2.2 Developing mathematical model for energy consumption of the smartphone for the local processing

Figures 5.9 and 5.10 show the multi-variable polynomial regression models between the number of faces, the face image database size and the energy consumption of the smartphone (1.3 GHz and 1.4 GHz quad-core processors) for face recognition.

The two-dimensional polynomial surfaces, which are used to predict the energy consumption of the smartphones (1.3 GHz and 1.4 GHz quad-core) for face recognition are shown in equations 5.6 and 5.7.

$$\begin{aligned} E_{L1.3} =& -3.699 \times 10^{-15} + 9.839 \times 10^{-17}x + 8.32 \times 10^{-18}y \\ &+ 0.0023xy - 6.294 \times 10^{-21}y^2 \end{aligned} \tag{5.6}$$

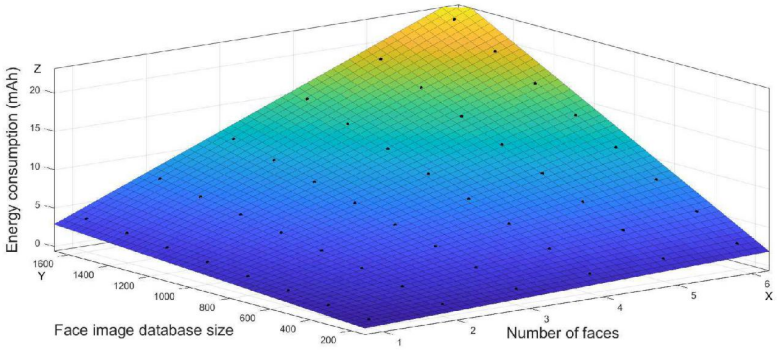


Figure 5.9: Polynomial regression relationship (of the 2nd order with SSE = 3.618×10^{-28}) between the number of faces, face image database size, and energy consumption of the smartphone (1.3 GHz quad-core processor) for face recognition

$$\begin{aligned} E_{L1.4} =& -0.5355 + 0.2432x + 0.001673y + 0.002543xy \\ &- 9.212 \times 10^{-7}y^2 \end{aligned} \tag{5.7}$$

5.2.3 Empirical measurement of smartphone energy consumption for the remote Cloud processing

Energy consumption of the smartphone is measured when the face image is transferred in the remote Cloud for face recognition. For both smartphones (1.3 GHz quad-core processor and 1.4 GHz quad-core processor) the energy consumption does not vary significantly over the number of faces and image database size, as shown in the Figure 5.11).

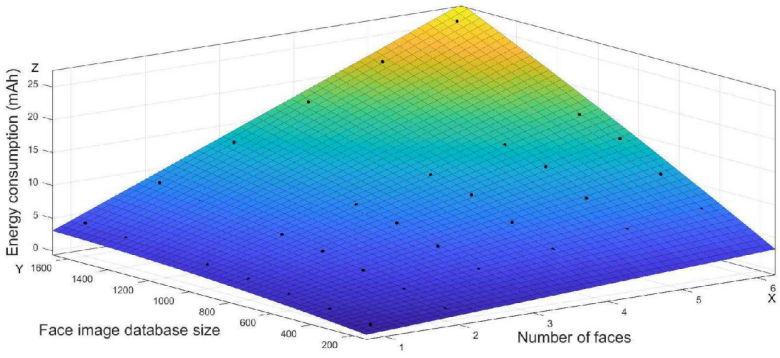


Figure 5.10: Polynomial regression relationship (of the 2nd order with SSE = 9.079) between the number of faces, face image database size, and energy consumption of the smartphone (1.4 GHz quad-core processor) for face recognition

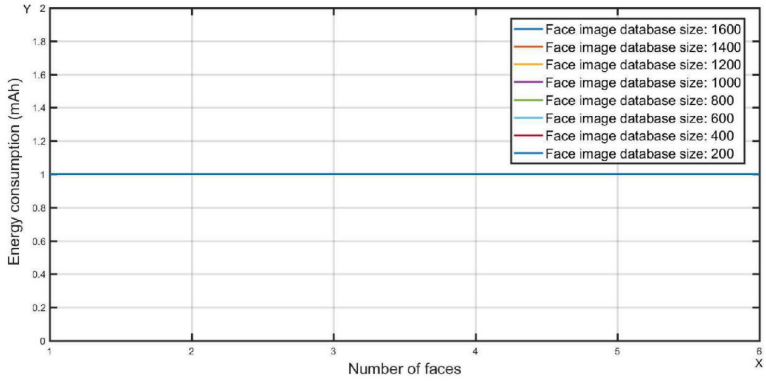


Figure 5.11: Energy consumption of the smartphone for face recognition in the remote Cloud (Edge and Core) with a different face database size

5.2.4 Developing mathematical model for energy consumption of the smartphone for the remote Cloud processing

Figure 5.12 shows the multi-variable linear regression model between the number of faces, face image database size, and energy consumption of the smartphone for remote Cloud processing. The two-dimensional linear surface which is used to predict the energy consumption of the smartphone is shown in equation 5.8.

$$E_{R-edge/core} = 1 \tag{5.8}$$

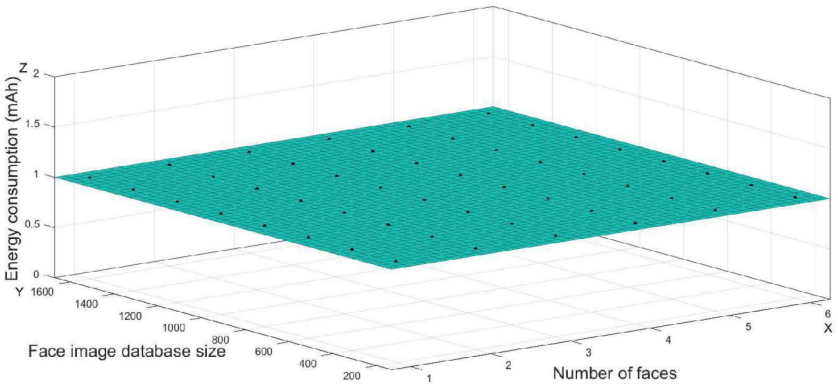


Figure 5.12: Linear regression relationship between the number of faces, face image database size, and energy consumption of the smartphone for remote Cloud processing

5.3 Mathematical model to estimate the processing cost

Estimating the processing cost for face recognition is crucial to decide to offload. In general, processing in the remote Cloud (Core and Edge) is expensive. For example, Amazon EC2's t2.xlarge on-demand service allows users to pay a fixed rate by the hour (shown in Table 7.1b). The cost increases linearly with time. Local processing is preferred since there is no cost associated it (shown in equation 5.9).

$$C_L = 0 \tag{5.9}$$

Assume that C_{R-edge} and C_{R-core} express the processing costs for remote face recognition in the Edge and Core Clouds, respectively. Linear equations are used to predict these costs that are shown in Equations 5.10 and 5.11.

$$C_{R-edge} = 1.66 \times 10^{-5} \times T_{R-edge} \tag{5.10}$$

$$C_{R-core} = 5.55 \times 10^{-5} \times T_{R-core} \tag{5.11}$$

Chapter 6

Offloading decision for service selection

This chapter presents the study of sensitivity analysis between two Multi-Criteria decision-making technique (MCDM) Simple Additive Weighting (SAW) and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS). Section 6.2 explains the steps of sensitivity analysis and discusses the experimental results. Section 6.3 and appendix A provides an overview of the SAW and the TOPSIS method.

6.1 Preamble

In the real world, the input parameters used in the Multi-Criteria decision-making technique (MCDM) methods are unstable and changeable. Therefore, it is a challenge to determine the appropriate MCDM method for decision making to offload the computation task. Approximately seventy MCDM methods are available to solve different decision-making problems [51]. Different MCDM methods require different underlying assumptions, information requirement from decision maker and evaluation principles. As a result, when different MCDM methods are applied to the same problem, at least in 40% of cases, each MCDM method generates dissimilar outcomes from the other methods [52]. In this thesis, Simple Additive Weighting (SAW) [17], the most widely used MCDM method and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [53], the computationally efficient MCDM method are considered for computation offloading decision making.

Table 6.1: Sample data set for the face recognition system with 1000 face database and number of face to be recognized is 2

	Criteria		
Choice	Time (s)	Energy (mAh)	Cost (Tk)
Smartphone processing	1.48	4.6	0
Edge Cloud processing	0.5	0.1	0.21
Core Cloud processing	1	0.1	1
Weight of the criteria	0.33 (w_1)	0.33 (w_2)	0.34 (w_3)

6.2 Sensitivity analysis

Sensitivity analysis can be helpful to find the most appropriate MCDM method regarding a particular problem [54]. It provides an efficient computation technique to study, how the results will change if the input data are changed and whether their decision results are stable. For the sensitivity analysis measurement, we have used the steps explained in Simanaviciene et al. [55]. The first step is to generate an initial set of data using the Monte Carlo method [56] from a sample dataset (shown in Table 6.1). Twenty random sets of data are generated from a *Normal Distribution* with mean represents the value from the decision matrix (shown in Figure 6.1) with standard deviation 10 %. In the next step, the MCDM method is applied to the random data set to generate 20 sets of decisions. Figure 6.2 shows the preference value of each choice estimated by the SAW method. Similarly, Figure 6.3 shows the preference value of each choice estimated by TOPSIS method. Finally, the average Relative Standard Deviation (RSD) on the ranking value of the choices is calculated from the decision data set (shown in Table 6.2). Figure 6.4 shows the ranking of choices by SAW method. Figure 6.5 shows the ranking of choices by TOPSIS method. The average RSD value of the SAW method is lower than the TOPSIS method. Since the sensitivity analysis shows that the SAW method is less sensitive than the TOPSIS method, the SAW method is more appropriate for decision making to offload face recognition processing.

6.3 Simple Additive Weighting (SAW)

The steps of the Simple Additive Weighting (SAW) method are explained below.

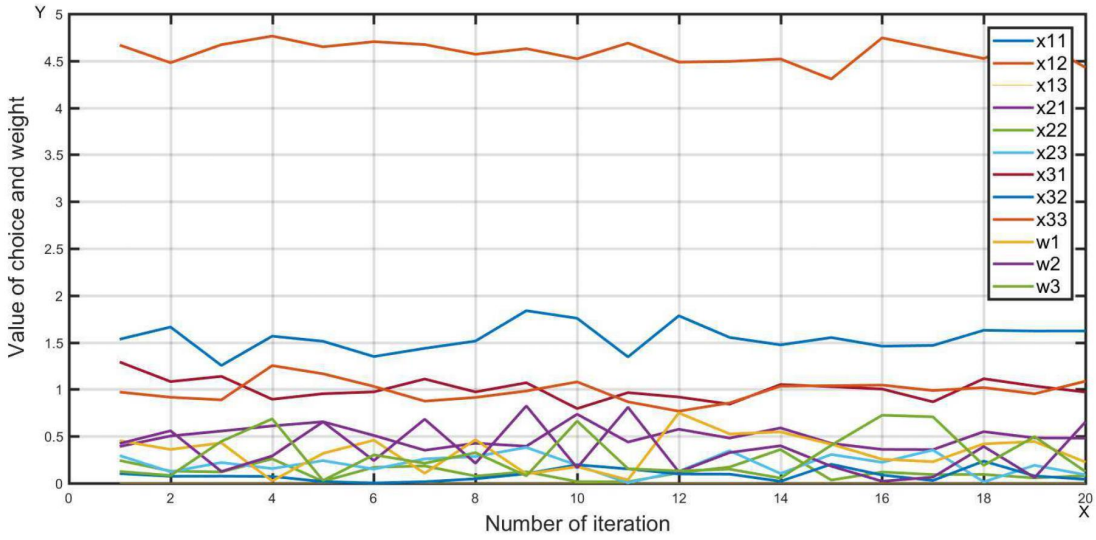


Figure 6.1: A random set of data are generated using the Monte Carlo method

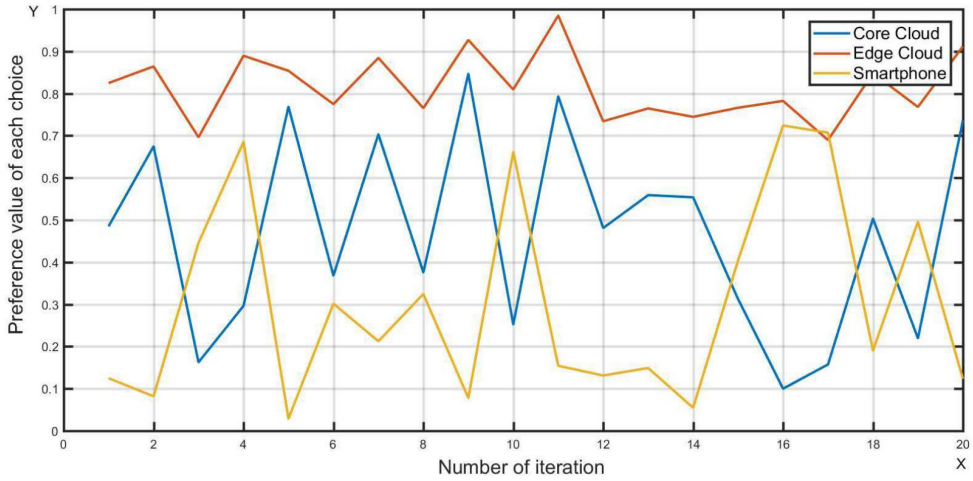


Figure 6.2: Estimate the preference value of each choice by SAW

1. A decision matrix (m × n) is constructed that includes m number of choices and n number of criteria. Then, the normalized decision matrix for criteria (shown in Equation 6.1) is calculated. There are several procedures exist in MADM for normalization (e.g., vector normalization, linear scale transformation (max-min method), linear scale trans-

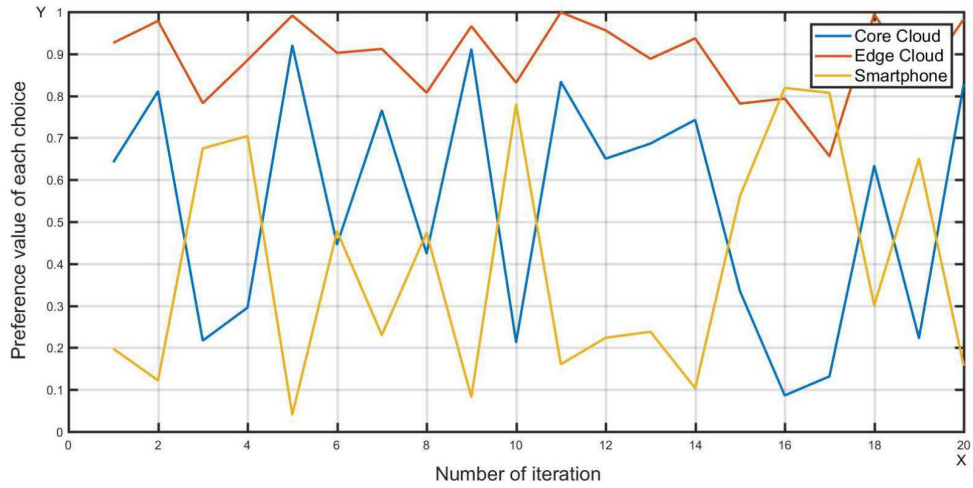


Figure 6.3: Estimate the preference value of each choice by TOPSIS

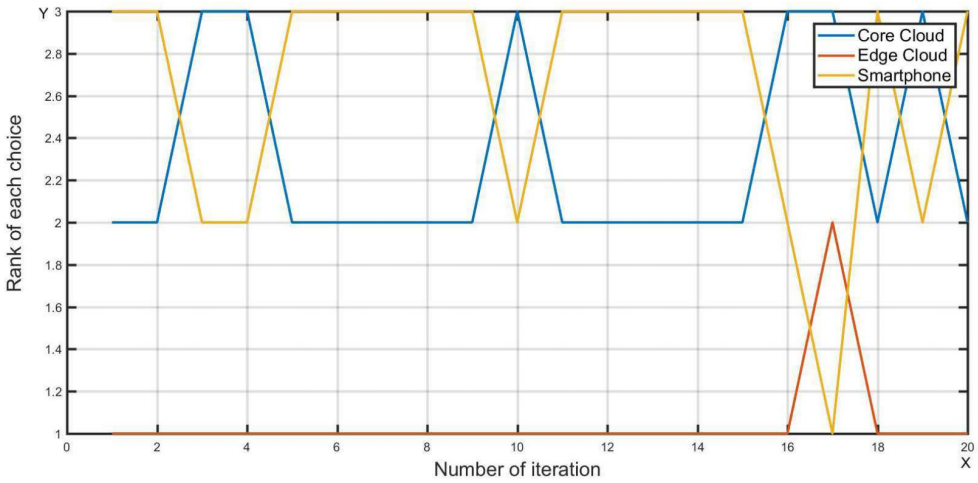


Figure 6.4: Ranking of choices by SAW

formation (max method), linear scale transformation (sum method)). Chakraborty et al. [57] shows that linear scale transformation (max method) exceeds other normaliza-

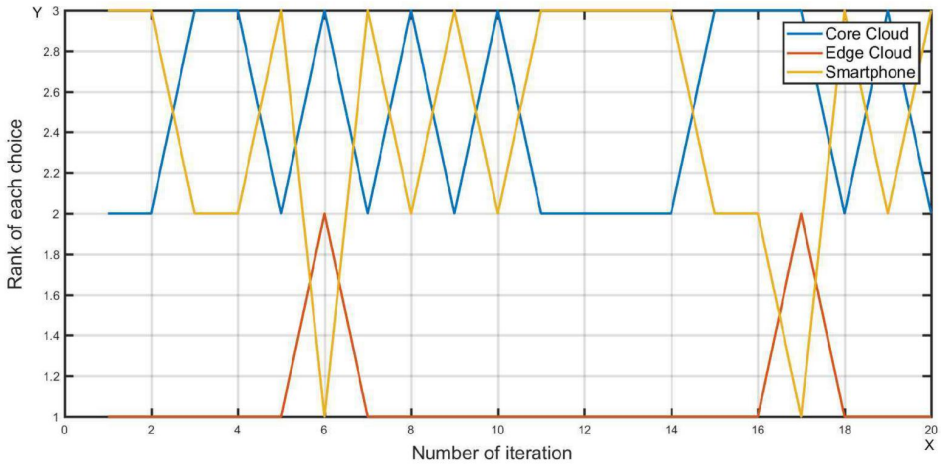


Figure 6.5: Ranking of choices by TOPSIS

Table 6.2: RSD of each choice

MCDM technique	RSD of each choice			Average of RSD
	Smartphone processing	Edge Cloud processing	Core Cloud processing	
SAW	22.16%	21.30%	20.44%	21.30%
TOPSIS	28.01%	27.98%	20.83%	25.61%

tion procedures when used with SAW (shown in Equation 6.2 and 6.3).

$$\begin{bmatrix} r_{11} & r_{12} & .... & r_{1n} \\ r_{21} & r_{22} & .... & r_{2n} \\ .... & .... & .... & .... \\ r_{m1} & r_{m2} & .... & r_{mn} \end{bmatrix} \tag{6.1}$$

For benefit criteria (more is better),

$$r_{ij} = \frac{x_{ij}}{x_j^{Max}} \tag{6.2}$$

For cost criteria (less is better),

$$r_{ij} = 1 - \frac{x_{ij}}{x_j^{Max}} \tag{6.3}$$

Where, x_j^{Max} is the maximum value among choices for criteria $j = 1, 2, \cdots, n$.

2. The overall preference value of each choice P_i is calculated by the multiplication of normalized score of criteria and corresponding criteria weight followed by summing of the products. The choice with maximum preference value is selected (shown in Equation 6.4).

$$P_i = \sum_{j=1}^{n} w_j r_{ij}; \tag{6.4}$$

Where, P_i: Preference value of the i^{th} choice
n: Number of attributes
r: Normalized score of the i-th choice with respect to the j^{th} criteria
w_j: Weight of the j^{th} criteria

Chapter 7

Validation of the proposed offloading framework

This chapter presents evaluation study to examine the accuracy of the offloading framework. Section 7.2 dicusses the experimental results when Edge Cloud and Core Cloud are implemented in the UIU data center and Amazon Web Service with t2.xlarge instance. Section 7.3 dicusses the experimental results when Edge Cloud and Core Cloud are implemented in the UIU data center and Microsoft Azure with B4ms instance. Both experimental result prove the accuracy of the offloading framework.

7.1 Preamble

Task offloading to the remote Cloud (Edge and Core) could extend the capability of the smartphone. Thus, an offloading framework is essential to make the decision whether or not to perform the task offloading. The proposed offloading decision-making algorithm is presented in Algorithm 1 which calculates the preference value of each choice. To validate the accuracy of the offloading framework, it is implemented and tested on the real-life and hypothetical scenario for local processing in the smartphone and remote processing in the Cloud. It is implemented for remote processing in the UIU data center as the Edge Cloud and in the two public Cloud vendors: Amazon Web Service (AWS), Microsoft Azure platform as the Core Cloud. Next, the experimental results are discussed.

Algorithm 1 Offloading decision

1: Known resource profile: Smartphone to Edge Cloud and Smartphone to Core Cloud bandwidth, smartphone to Edge Cloud and smartphone to Core Cloud RTT, smartphone and Edge/Core Cloud processor speed, Edge and Core Cloud processing cost
2: Set weights w_1, w_2, w_3 based on user preferences
3: Estimate T_L, T_{R-edge}, T_{R-Core} (section 5.1)
4: Estimate E_L, E_{R-edge}, E_{R-Core} (section 5.2)
5: Estimate C_L, C_{R-edge}, C_{R-Core} (section 5.3)
6: Compute P_S, P_E, P_C (Equation 6.4)
7: **if** P_S is maximum **then**
8: Execute in the smartphone
9: **else if** P_E is maximum **then**
10: Execute in the Edge Cloud
11: **else if** P_C is maximum **then**
12: Execute in the Core Cloud
13: **end if**

Table 7.1: Resource profile

(a) Network resource parameters

Resource profile	Smartphone to Edge Cloud (UIU)			Smartphone to Core Cloud (AWS)			Smartphone to Core Cloud (Azure)
	WiFi	3G	4G	WiFi	3G	4G	WiFi
Network bandwidth (Mbps)	14.5	3.5	13	3	3.4	14	4.25
Round-trip-time (ms)	12	25	40	112	120	130	88
Hop count	13	12	11	27	20	18	14

(b) Remote Cloud (Edge and Core) cost profile

Types of Cloud service	Computation processing cost
Edge Cloud (UIU)	$0.060 per hour
Core Cloud (Amazon Web Service t2.xlarge instance)	$0.2394 per hour
Core Cloud (Microsoft Azure)	$0.267 per hour

Table 7.2: The weight of each criterion is assigned to five different cases

Case	Weight of the criteria		
	Face recognition time	Energy consumption of the smartphone	Processing cost
Save time	1.0	0.0	0.0
Save time, energy consumption	0.5	0.5	0.0
Save time, cost (priority: cost)	0.2	0	0.8
Save time, energy consumption, cost	0.33	0.33	0.33
Save time, energy consumption, cost (priority: energy consumption)	0.25	0.50	0.25

7.2 Edge Cloud (UIU data center) and Core Cloud (Amazon Web Service t2.xlarge instance)

For this experiment, it is assumed that the total image database size is 1000 and the number of faces to be recognized is 6. Table 7.1 shows the resource profiling parameters for communicating from a smartphone with 1.4 GHz quad-core processor to the remote Cloud (Edge and Core) using the communication method: WiFi, 3G, and 4G. Some example cases are presented in the following discussion.

- **Case 1**

  Target: Save *time*

  Decision by the framework: The framework makes the decision to execute in the Edge Cloud as the value of P_E is maximum.

  Discussion: Actual measurements (shown in Table 7.4a) show that Edge Cloud processing time is 0.90 second, which is faster than local processing in the smartphone (2.95 second) and remote processing in the Core Cloud (1.05 second). The experiment shows that the proposed framework makes the correct decision to offload in the Edge Cloud to minimize processing time.

- **Case 2**

  Target: Save time, energy consumption.

  Decision by the framework: The framework makes the decision to execute in the

Edge Cloud as the value of P_E is maximum.

Discussion: Actual measurements (Table 7.4a) show that the Edge Cloud processing is faster (0.90 second) and save energy consumption of the smartphone (1 mAh) compare with the local processing in the smartphone (2.95 second, 17 mAh) and remote processing in the Core Cloud (1.05 second, 1 mAh). Therefore, the framework makes the correct decision to execute in Edge Cloud to minimize both processing time and energy consumption of the smartphone.

- **Case 3**

  Target: Save time, cost (priority: cost)

  Decision by the framework: The framework makes the decision to execute in the smartphone as the value of P_S is maximum.

  Discussion: Actual measurements (Table 7.4) show that the Edge Cloud processing is faster (0.90 s) but expensive compared with the local processing in the smartphone. There is no cost associated with local processing. Therefore, the framework makes the correct decision to execute in the smartphone to minimize both processing time and cost when priority is given to save cost.

- **Case 4**

  Target: Save time, energy consumption, cost

  Decision by the framework: The framework makes the decision to execute in the Edge Cloud as the value of P_E is maximum.

  Discussion: Actual measurements (Table 7.4a) show that the Edge Cloud processing is faster and save energy consumption of the smartphone compare with the local processing in the smartphone remote processing in Core Cloud. However, Edge Cloud processing is expensive than local processing. To minimize the three criterion (processing time, energy consumption, processing cost) simultaneously, the framework makes the decision to execute in the Edge Cloud.

- **Case 5**

  Target: Save time, energy consumption, cost (priority: energy consumption)

Decision by the framework: The framework makes the decision to execute in the Edge Cloud as the value of P_E is maximum.

Discussion: Actual measurements (Table 7.4) show that remote processing in the Edge Cloud saves processing time and cost compared with the Core Cloud. Although there is no cost associated with local processing, it will increase the processing time and energy consumption of the smartphone. Therefore, when the priority is to save energy consumption and simultaneously minimize processing time and processing cost, the framework makes the correct decision to execute in the Edge Cloud.

Similarly, the offloading framework makes the correct decision for the communication method 3G and 4G (shown in Table 7.3a). Table 7.4b and 7.4c shows the actual measurement value of face recognition processing time, energy consumption and processing cost for the commutation method: 3G and 4G.

One of the limitations of the real-life experimental setup is, it is not possible to vary the network bandwidth, round-trip-time or hop count. Therefore, the bandwidth from a smartphone (1.3 GHz quad-core processor) to Edge Cloud and Core Cloud has been assumed to be 0.05 Mbps and 0.1 Mbps to evaluate the performance of the offloading framework. The decision of the offloading framework (shown in Table 7.3b) validates that the framework can provide a dynamic decision based on the dynamic situation.

7.3 Edge Cloud (UIU data center) and Core Cloud (Microsoft Azure B4ms instance)

For this experiment, it is assumed that the total image database size is 1400 and the number of faces to be recognized is 3. Similar to the three different cases discussed earlier, this experiment also shows that the offloading framework makes the correct decision shown in Table 7.3a. Table 7.5 shows, the actual measurement value of face recognition time, energy consumption and processing cost.

Table 7.3: Framework decision

(a) Based on real-life resource profile measurement

		Configuration	
		Smartphone (1.4 GHz quad-core), Edge Cloud (UIU)	
	Case	Core Cloud (AWS)	Core Cloud (Azure)
Decision	Save time	Execute in the Edge Cloud ($P_E : maximum$)	Execute in the Edge Cloud ($P_E : maximum$)
	Save time, energy consumption	Execute in the Edge Cloud ($P_E : maximum$)	Execute in the Edge Cloud ($P_E : maximum$)
	Save time, cost (priority: cost)	Execute in the smartphone ($P_S : maximum$)	Execute in the smartphone ($P_S : maximum$)
	Save time, energy consumption, cost	Execute in the Edge Cloud ($P_E : maximum$)	Execute in the Edge Cloud ($P_E : maximum$)
	Save time, energy consumption, cost (priority: energy consumption)	Execute in the Edge Cloud ($P_E : maximum$)	Execute in the Edge Cloud ($P_E : maximum$)

(b) Based on the hypothetical resource profile measurement

		Configuration
		Smartphone (1.3 GHz quad-core), Edge Cloud (UIU)
	Case	Core Cloud (AWS)
Decision	Save time	Execute in the smartphone ($P_S : maximum$)
	Save time, energy consumption	Execute in the Core Cloud ($P_C : maximum$)
	Save time, cost (priority: cost)	Execute in the smartphone ($P_S : maximum$)
	Save time, energy consumption, cost	Execute in the Edge Cloud ($P_E : maximum$)
	Save time, energy consumption, cost (priority: energy consumption)	Execute in the Edge Cloud ($P_E : maximum$)

Table 7.4: The actual measurement value for local processing in the smartphone with 1.4 GHz quad-core processor, remote processing in Edge Cloud (UIU data center) and Core Cloud (AWS)

(a) Commutation method: WiFi

Case	Local processing (Smartphone)	Remote processing (Edge Cloud)	Remote processing (Core Cloud)
Face recognition time (s)	2.95	0.90	1.05
Smartphone energy consumption (mAh)	17	1	1
Processing cost ($)	0	1.23×10^{-5}	4.81×10^{-5}

(b) Commutation method: 3G

Case	Local processing (Smartphone)	Remote processing (Edge Cloud)	Remote processing (Core Cloud)
Face recognition time (s)	2.95	1.57	1.60
Smartphone energy consumption (mAh)	17	1	1
Processing cost ($)	0	1.84×10^{-5}	6.21×10^{-5}

(c) Commutation method: 4G

Case	Local processing (Smartphone)	Remote processing (Edge Cloud)	Remote processing (Core Cloud)
Face recognition time (s)	2.95	1.20	1.18
Smartphone energy consumption (mAh)	17	1	1
Processing cost ($)	0	1.44×10^{-5}	4.67×10^{-5}

Table 7.5: The actual measurement value for local processing in the smartphone with 1.4 GHz quad-core processor, remote processing in Edge Cloud (UIU data center) and Core Cloud (Azure)

Case	Commutation method: WiFi		
	Smartphone processing	Edge Cloud Processing	Core Cloud Processing
Face recognition time (s)	3.01	0.73	0.92
Smartphone energy consumption (mAh)	9.66	1	1
Processing cost ($)	0	1.04×10^{-5}	3.99×10^{-5}

Chapter 8

Conclusion

Faster wireless communication (3G, 4G, and WiFi) allows the smartphone to offload processing intensive job to the remote Cloud (Edge and Core) to overcome its limitation like inadequate processing speed, limited battery capacity. In this thesis, an offloading framework is proposed for a trade-off between processing time and energy saving, while considering the economic factor. IFogSim simulation tool is used to verify whether offloading computation task to the remote Cloud (Core and Edge) can save processing time and energy consumption of the smartphone. The simulation results show that local computation can reduce the network delay and network usage but increase the energy consumption of the smartphone. It establishes the fact that to make the offloading decision favorable to the smartphone multiple criteria need to be considered. Simple Additive Weighting (SAW) of Multi-Criteria decision-making technique is used to make the decision for offloading based on the processing time, the energy consumption of the smartphone and the processing cost criteria. Mathematical models are proposed to estimate the value of the processing time, the energy consumption of the smartphone and the processing cost criteria. The offloading framework is experimentally validated on multiple real-life scenarios, which proves that the proposed framework is realistic, effective and efficient.

In future work, we will expand the proposed framework for real-time applications, such as real-time face recognition. Furthermore, this framework could be extended to video streaming with transcoding. Virtual reality/augmented reality (VR/AR) is another research area which could be benefited from task offloading. Immersive VR/AR applications are going to demand more processing power which current headsets are not able to provide due to hardware limitations. Rather than costly upgrades to a headset, a cloud-based service can more

easily enhance the speed of its core infrastructure [58].

Bibliography

[1] S. Islam and J.-C. Grégoire, "Network edge intelligence for the emerging next-generation internet," *Future Internet*, vol. 2, no. 4, pp. 603–623, 2010. x, 7

[2] "Number of smartphone users worldwide 2014-2020 — statista," https://www.statista.com/statistics/330695/number-of -smartphone-users-worldwide/, (Accessed on 08/09/2018). 1

[3] "iphone heart monitor tracks your heartbeat unless you are dead," https://gizmodo.com/5056167/iphone-heart-monitor-tracks-your-heartbeat-unless-you-are-dead, (Accessed on 08/09/2018). 1

[4] K. Naik, *A survey of software based energy saving methodologies for handheld wireless communication devices*. Department of Electrical and Computer Engineering, University of Waterloo, 2010. 1

[5] D. Kovachev, T. Yu, and R. Klamma, "Adaptive computation offloading from mobile devices into the cloud," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*. IEEE, 2012, pp. 784–791. 2, 9

[6] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When mobile terminals meet the cloud: computation offloading as the bridge," *IEEE Network*, vol. 27, no. 5, pp. 28–33, September 2013. 2

[7] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *HotOS*, vol. 9, 2009, pp. 8–11. 2

[8] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: enabling mobile phones as interfaces to cloud applications," in *Proceedings of the 10th*

ACM/IFIP/USENIX International Conference on Middleware. Springer-Verlag New York, Inc., 2009, p. 5. 2

[9] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *Communications Workshops (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 728–732. 2

[10] D. Durkee, "Why cloud computing will never be free," *Communications of the ACM*, vol. 53, no. 5, pp. 62–69, 2010. 3

[11] A. E. C. Cloud, "Amazon web services," *Retrieved November*, vol. 9, p. 2011, 2011. 3, 7

[12] M. Copeland, J. Soh, A. Puca, M. Manning, and D. Gollob, "Microsoft azure and cloud computing," in *Microsoft Azure*. Springer, 2015, pp. 3–26. 3

[13] "Google cloud including gcp & g suite — try free — google cloud," https://cloud.google.com/, (Accessed on 08/07/2018). 3

[14] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010. 3

[15] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017. 4, 15

[16] W. Wolters, B. Mareschal *et al.*, "Novel types of sensitivity analysis for additive mcdm methods," *European Journal of Operational Research*, vol. 81, no. 2, pp. 281–290, 1995. 4

[17] L. Abdullah and C. R. Adawiyah, "Simple additive weighting methods of multi criteria decision making and applications: A decade review," *International Journal of Information Processing and Management*, vol. 5, no. 1, p. 39, 2014. 4, 31

[18] Q. K. Gill and K. Kaur, "A review on energy efficient computation offloading frameworks for mobile cloud computing," *International Journal of Innovations & Advancement in Computer Science*, vol. 5, p. 1, 2016. 4

[19] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith, "Intel virtualization technology," *Computer*, vol. 38, no. 5, pp. 48–56, 2005. 7

[20] U. Rackspace, "Inc.,"the rackspace cloud,"," 2010. 7

[21] "At&t cloud services - business cloud solutions," https://www.business.att.com/portfolios/cloud.html, (Accessed on 11/21/2018). 7

[22] V. I. A. H. Availability, "Services with vmware ha," 2007. 7

[23] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015. 7

[24] S. Islam and J.-C. Grégoire, "Giving users an edge: A flexible cloud model and its application for multimedia," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 823–832, 2012. 8

[25] S. A. Haque, S. Islam, M. J. Islam, and J.-C. Grégoire, "An architecture for client virtualization: A case study," *Computer Networks*, vol. 100, pp. 75–89, 2016. 8

[26] N. Muslim and S. Islam, "Face recognition in the edge cloud," in *Proceedings of the International Conference on Imaging, Signal Processing and Communication.* ACM, 2017, pp. 5–9. 8

[27] A. Al Amin, S. Hossain, M. Uddin, N. Muslim, S. Islam, and J.-C. Grégoire, "Computing at the edge cloud: Performance analysis of an image comparison web service," in *Proceedings of the International Conference on Imaging, Signal Processing and Communication*, 2018, accepted for publication. 8

[28] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet.* Citeseer, 2014, pp. 48–55. 8

[29] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015. 8

[30] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, 2009. 8

[31] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Computers and communications (ISCC), 2012 IEEE symposium on*. IEEE, 2012, pp. 000 059–000 066. 8

[32] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Infocom, 2012 Proceedings IEEE*. IEEE, 2012, pp. 945–953. 8

[33] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2010, pp. 59–79. 9

[34] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016. 9

[35] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6. 9

[36] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62. 9

[37] V. Namboodiri and T. Ghose, "To cloud or not to cloud: A mobile device perspective on energy consumption of applications," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*. IEEE, 2012, pp. 1–9. 9

[38] S. A. Karim and J. J. Prevost, "Efficient mobile computation using the cloud," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*. IEEE, 2015, pp. 620–626. 9

[39] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "mcloud: A context-aware offloading framework for heterogeneous mobile cloud," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 797–810, 2017. 9

[40] T.-Y. Lin, T.-A. Lin, C.-H. Hsu, and C.-T. King, "Context-aware decision engine for mobile cloud offloading," in *Wireless Communications and Networking Conference Workshops (WCNCW), 2013 IEEE*. IEEE, 2013, pp. 111–116. 10

[41] E. W. Weisstein, "Least squares fitting," 2002. 12

[42] G. Leitmann and A. Marzollo, *Multicriteria decision making*. springer, 2014, vol. 211. 13

[43] D. Sabaei, J. Erkoyuncu, and R. Roy, "A review of multi-criteria decision making methods for enhanced maintenance delivery," *Procedia CIRP*, vol. 37, pp. 30–35, 2015. 13

[44] "Mobile vision api," https://developers.google.com/vision/, (Accessed on 03/07/2018). 14

[45] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," *Computer vision-eccv 2004*, pp. 469–481, 2004. 14

[46] I. MathWorks, *Curve fitting toolbox: for use with MATLAB®: user's guide*. MathWorks, 2002. 22

[47] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "iperf: Testing the limits of your network," 2003. 24

[48] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. De Lucia, "Software-based energy profiling of android apps: Simple, efficient and reliable?" in *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*. IEEE, 2017, pp. 103–114. 25

[49] "Profile battery usage with batterystats and battery historian — android developers," https://developer.android.com/studio/profile/battery-historian, (Accessed on 10/20/2018). 25

[50] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007. 26

[51] X. Sun, V. Gollnick, Y. Li, and E. Stumpf, "Intelligent multicriteria decision support system for systems design," *Journal of Aircraft*, vol. 51, no. 1, pp. 216–225, 2014. 31

[52] H. Voogd, *Multicriteria evaluation for urban and regional planning*. Pion London, 1983, vol. 207. 31

[53] C.-L. Hwang and K. Yoon, "Methods for multiple attribute decision making," in *Multiple attribute decision making*. Springer, 1981, pp. 58–191. 31

[54] H. A. Taha, *Operations research: An introduction (for VTU)*. Pearson Education India, 2005. 32

[55] R. Simanaviciene and L. Ustinovichius, "Sensitivity analysis for multiple criteria decision making methods: Topsis and saw," *Procedia-Social and Behavioral Sciences*, vol. 2, no. 6, pp. 7743–7744, 2010. 32

[56] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016, vol. 10. 32

[57] S. Chakraborty and C.-H. Yeh, "A simulation based comparative study of normalization procedures in multiattribute decision making," in *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases*, vol. 6, 2007, pp. 102–109. 34, 52

[58] P. Chiu, P. Tseng, and K. Feng, "Cloud computing based mobile augmented reality interactive system," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2014, pp. 3320–3325. 45

[59] K. P. Yoon and C.-L. Hwang, *Multiple attribute decision making: an introduction*. Sage publications, 1995, vol. 104. 52

[60] H.-S. Shih, H.-J. Shyur, and E. S. Lee, "An extension of topsis for group decision making," *Mathematical and Computer Modelling*, vol. 45, no. 7-8, pp. 801–813, 2007. 52

Appendix A

Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)

A.1 Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)

The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method, which is proposed by Hwang et al. [59], is another well-known MCDM method. This method assumes that the selected choice should have the shortest distance from the ideal solution as well as the longest distance from the non-ideal solution. This method represents the rational of human choice that the best choice should be nearest to the positive ideal solution and farthest from the negative ideal solution [60]. The TOPSIS method consists of these steps:

1. Calculate the normalized decision matrix for each criteria (shown in equation 6.1). Chakraborty et al. [57] shows that vector normalization method outperforms other normalization methods when used with TOPSIS (shown in Equation A.1).

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{A.1}$$

2. Weighted normalized value is calculated for each criteria (shown in Equation A.2). The weighted value v_{ij} is derived from the product of elements in the normalized decision matrix and criteria weights.

$$v_{ij} = w_j \cdot r_{ij}; \tag{A.2}$$

   Where, w_j: The weight of the j-th criteria

r: Normalized score of the i-th choice with respect to the j-th criteria

3. Identify, positive-ideal solutions A* and negative-ideal solutions A^- for each criteria (shown in Equation A.3 and A.4). The positive-ideal solution is the solution that maximizes the benefit criteria and minimizes the cost criteria whereas the negative-ideal solution maximizes the cost criteria and minimizes the benefit criteria.

$$A^+ = \left(\ v_1^+, v_2^+, \cdots, v_n^+ \ \right) \tag{A.3}$$

$$A^- = \left(\ v_1^-, v_2^-, \cdots, v_n^- \ \right) \tag{A.4}$$

where, $v_j^+ = \{max(v_{ij}) \ if \ j \epsilon J; \ min(v_{ij}) \ if \ j \epsilon J'\}$
$v_j^- = \{min(v_{ij}) \ if \ j \epsilon J; \ max(v_{ij}) \ if \ j \epsilon J'\}$
J is associated with the beneficial attributes and J' is associated with the non-beneficial attributes.

4. Calculate the separation distance of each choice from the positive-ideal solutions (shown in Equation A.5. Similarly, calculate the separation distance of each choice from the negative-ideal solutions (shown in Equation A.6).

$$S_i^* = \sqrt{\sum_{j=1}^{n} \left(\ v_{ij} - v_j^* \ \right)^2} \tag{A.5}$$

$$S_i^- = \sqrt{\sum_{j=1}^{n} \left(\ v_{ij} - v_j^- \ \right)^2} \tag{A.6}$$

5. Calculate, the overall preference value (Pi) for each choice (shown in Equation A.7). The choice with maximum preference value is selected.

$$P_i = \frac{S_i^-}{\left(S_i^* + S_i^-\right)} \tag{A.7}$$