LATENCY EFFICIENT CACHE PLACEMENT USING LEARNING TECHNIQUES IN MOBILE EDGE NETWORKS

by

Lubna Badri Mohammed

Ph.D. Computer Engineering, 2000, University of Technology

M.Sc. Computer Engineering, 1996, University of Technology

B.Sc. Control and Systems Engineering, 1994, University of Technology

A dissertation
presented to Ryerson University
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in the
program of Electrical and Computer Engineering

Toronto, Ontario, Canada

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A DISSERTATION

I hereby declare that I am the sole author of this dissertation. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Abstract

Latency Efficient Cache Placement using Learning Techniques in Mobile Edge Networks

Lubna Badri Mohammed, 2022

Doctor of Philosophy

Electrical and Computer Engineering

Ryerson University, Toronto, Ontario, Canada

Future wireless networks provide interesting research challenges with the exponential growth in mobile data traffic, the advent of new high computational and real-time applications that cause many-fold increases in traffic and require low latency from the network. The emerging need to bring data closer to users and minimize the traffic off the macrocell base station (MBS) introduces caches at the edge of the networks. Storing most popular files in user terminals (UTs) and small base stations (SBSs) caches inside the mobile edge networks (MENs) is a promising approach to the challenges that face future data-rich wireless networks. Caching in the mobile UT allows obtaining requested contents directly from its nearby UT caches through device-to-device communication.

This thesis addresses several challenges faced in developing a solution for cache placement at the edge of the network due to continuous changes in content popularity, user mobility, and the number of users within each network. It also considers the challenges related to high computation requirements of future applications that need to satisfy power and delivery time constraints. This dissertation aims to overcome those challenges in developing new solutions by employing intelligence and machine learning techniques (ILT) for mobile edge networks. We formulate the cache placement problem as a latency-efficient cache placement optimization problem that considers four objectives, to place contents in SBSs and UT caches. The multi-objective function takes advantage of user mobility patterns to decide on each SBS and UT cache content. The function is resolved into a weighted fusion decision with four objectives. Three of them are related to user mobility computed from previous data sets, and one objective is related to content popularity. We study the impact of user mobility on increasing the cache hit rate, which decreases the latency of downloading the requested data content. The results show the effect of user mobility on reducing the total energy consumed for transmitting the contents to the UTs. We propose a new cache placement algorithm based on user locations, contact probability, communication range, contact duration, and content popularity, formulating cache placement decisions as a binary classification problem (to cache and not to cache). Artificial neural networks (ANN), support vector machine (SVM), and logistic regression (LR) are used to model cache placement decisions. We investigate the characteristics of the input features (attributes) and the properties of these characteristics that affect supervised machine learning approaches. The performance of the new cache placement models using supervised learning techniques is evaluated to study the sensitivity of classification decisions with the change of system parameters. Finally, we develop a semi-supervised self-training (SSST) classification model for the cache placement problem. We assess the proposed SSST algorithm through experiments with datasets on different learning techniques. The performance comparison of different machine learning models was carried out with the same datasets. For the hit rate, we investigated the sensitivity of the classification by the changes in the environment parameters to show the effectiveness of the proposed theme.

Acknowledgment

I want to express my deepest gratitude to my supervisor Professor Alagan Anpalagan, for his motivating guidance, valuable suggestions, constant encouragement, and support. I have significantly benefited from his wealth of knowledge and meticulous editing. Furthermore, I am highly grateful that Prof. Alagan Anpalagan accepted me as a Ph.D. student and continued to have faith in me over the years. It was honor to learn from Professor Alagan Anpalagan.

My sincere thanks to my co-supervisor, Professor Muhammad Jasimudddin. Professor Muhammad Jasimuddin taught me the wireless networks course that inspired me to work on developing future wireless networks. I am also grateful for his valuable feedback during my research.

I express my sincere gratitude to Dr. Ahmed Shaharyar Khwaja from the WINCORE Lab, Ryerson University, for his feedback and suggestions throughout this research.

My endless gratitude to my parents, Professor Badri Mohammed and Professor Ibtisam Marhon, for their continuous support. You have always supported me. Thank you for your love and for constantly reminding me of the end goal.

Last but not least, my warm and heartfelt thanks go to my loving husband Esam and my lovely boys, Hani, Mohammed, Mustafa, and Bilal, for giving me strength. I love you all, and without you, this dissertation would not have been possible.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
BS	**B**ase **S**tation
CF	**C**ollaborative **F**iltering
CoMP	**C**ooperative **M**ulti **P**oint
CRP	**C**hines **R**ertaurant **P**rocess
CSOs	**C**ache **S**ystem **O**bjectives
CU	**C**loud **U**nit
D2D	**D**evice **2** **D**evice
DBN	**D**eep **B**elief **N**etwork
DNN	**D**eep **N**eural **N**etwork
ELM	**E**xtreme **L**earning **M**achine
EM	**E**xpectation **M**aximization
EN	**E**dge **N**ode
ETSI	**E**uropean **T**elecommunications **S**tandards **I**nstitute
FSS	**F**uzzy **S**oft **S**et
ICRP	**I**ndividual **C**ontent **R**equest **P**robability
IIoT	**I**Industrial **I**nternet **o**f **T**hings
ILT	**I**ntelligent **L**earning **T**echniques
IoT	**I**nternet **o**f **T**hing
ISG	**I**ndustry **S**pecification **G**roup
LFU	**L**east **F**requently **U**sed
LRU	**L**east **R**ecently **U**sed
MCC	**M**obile **C**loud **C**omputing
MDP	**M**arkov **D**ecision **P**rocess
MEC	**M**obile **E**dge **C**omputing
MEN	**M**obile **E**dge **N**etwork

MILP	**M**ixed-**I**nteger **L**inear **P**rogram
MSE	**M**ean **S**quare **E**
PP	**P**ricing **P**roblem
QoE	**Q**uality **o**f **E**xperience
QoS	**Q**uality **o**f **S**ervice
RAN	**R**adio **A**ccess **N**etwork
RMP	**R**estricted **M**aster **P**roblem
SBS	**S**mall **B**ase **S**tation
SCN	**S**mall **C**ell **N**etwork
SDN	**S**oftware **D**efined **N**etwork
SVC	**S**calable **V**ideo **C**oding
nhj **SVD**	**S**ingular **V**alued **D**ecomposition
TL	**T**ransfer **L**earning
TUR	**T**otal number of **U**sers' **R**equests
UT	**U**ser **T**erminal

Symbols

N	Number of SBSs
M	Number of user terminals
u	Set of user terminals
c_s	Set of cache storages in SBSs
c_u	Set of cache storages in user terminals
F	Number of files in MBS storage library
f	Set of files in the library
p	Set of files popularity probability
γ	The skewness of the Zipf distribution popularity
$T_{i,j}^{SBS}$	Contact duration of mobile UT u_i and SBS s_j
$\lambda_{i,j}$	Contact rate of mobile UT u_i and SBS s_j
d^{SBS}	Communication range between SBS and UT
d^{D2D}	Communication range of two UT devices
$T_{i,k}^{UT}$	Contact duration of mobile UT u_i and UT u_k
$h_{i,z}^{UT}$	Cache hit references of UT u_i and file f_z
$h_{i,z}^{SBS}$	Cache hit references of SBS s_j and file f_z
$T_{d_z}^{B}$	Backhaul download time
$R_{i,z}^{s_j}$	Date rate of file f_z of the UT u_i to download from SBS
$R_{k,z}^{u_i}$	Data rate of file f_z of the UT u_i to download from UT u_k
β_{UT}	Inverse of power amplifier efficiency factor
$\mathcal{P}_T^{u_k}$	Mobile device transmission power
$\mathcal{P}_C^{u_k}$	Circuit power consumption of mobile UT u_k

$\mathcal{P}_H^{u_k}$	Energy consumption of of caching hardware devices
$\mathcal{P}^{u_k}$	Power consumption of UT u_k in D2D communication
E^{u_k}	Energy consumption in D2D communication
$W_{i,k}^{UT}$	Channel bandwidth from UT u_k to UT u_i
f_{l_z}	Length of file
$d_{i,k}$	Distance between u_k and u_i
σ_{UT}^2	Average noise power for D2D communication
α_{UT}	Path loss factor for D2D communication
α_{SBS}	Path loss factor for SBS to UT communication
$W_{i,j}^{SBS}$	Downlink transmission bandwidth from SBS s_j to UT u_i
$\mathcal{P}_T^{s_j}$	SBS transmission power
E^{s_j}	Energy consumption in SBS caching communication
$d_{i,j}$	Distance between s_j and u_i
σ_{SBS}^2	Average noise power
L^{UT}	Set of locations of all UTs in one macrocell
L^{SBS}	Set of locations of all SBSs in one macrocell
$\mathcal{T}^{UT,SBS}$	Contact durations between all UTs and other UTs and SBSs within one cell
$\mathcal{T}^{*UT,SBS}$	Normalized contact durations between all UTs and other UTs and SBSs within one cell
$\mathcal{D}^{UT,SBS}$	Communication distances between all UTs and other UTs and SBSs within one cell
$\mathcal{D}^{*UT,SBS}$	Normalized communication distances between all UTs and other UTs and SBSs within one cell
$P_m^{UT,SBS}$	User terminal UT contact probability with all UTs and SBSs within one cell
h	SBS cache hit rate
$\mathcal{V}$	Number of user requests

Chapter 1

Introduction

1.1 Overview

In recent years, an exponential increase in traffic load has been consistently noticed in wireless networks due to multimedia streaming applications and services, mobile video streaming, web browsing, and social network inter-connections. Wireless devices are expected to generate much higher traffic than wired network devices in the future. To handle these traffic explosions, mobile wireless networks require continuous evolution and improvement of the performance in terms of power consumption, data throughput, and utilization of network resources such as backhaul network capacity and bandwidth [2]. Mobile edge networks (MENs) are one of the candidate solutions in wireless networks. These networks include small cell networks (SCNs), among others. Each SCN has one macro base station and several small base stations (SBSs) with short-range communication, low-power, low-cost, and cache storage to store contents closer to users [3]. However, the demand for contents by devices and many different applications and services results in constraints on latency, energy consumption, and quality of services (QoS), which require novel approaches to solutions at both architectural and algorithmic levels.

Considering these problems, researchers investigated the possibility of caching content items locally and proactively at the edge of the mobile edge networks (i.e., caching in SBS and user terminal (UT)) before users request them. The local cache is a promising approach to improve the network backhaul link bottleneck, and quality of service [3] [4] by providing faster connectivity, lower latency, and less power consumption. Furthermore, SBSs, which are also sometimes called

Femto caches, caching helpers, or simply helpers, have generally high storage capabilities that are used to build wireless distributed caching infrastructure [5]. Utilizing the advantages of storing contents closer to UTs at the edge of the mobile edge networks allows users to download requested contents from neighboring SBS caches or UT caches located within the SBS cell through device-SBS or device-to-device (D2D) communication, respectively, which can reduce the latency while saving power consumption and the network backhaul resources.

The performance of future services and applications are highly dependent upon user location, data rate, and network resources. Internet with massive mobile traffic, many mobile users with different moving speeds suffer from poor support for such services. Thus, user mobility patterns should be considered while designing caching in MENs. Due to dynamic updates of user demands, content popularity, and user mobility, it is difficult to decide which contents to cache, where to cache them, and from where to deliver them using traditional decision-making techniques. Moreover, the large amount of data needed to develop algorithms for cache systems estimates cache contents, access, and delivery a complex and challenging task. To meet all these challenges, we explore in this dissertation machine learning and intelligent decision-making techniques for storing, accessing, and delivering the vast amount of data generated within the MENs through caching.

1.2 Objectives

This thesis investigates the problem of cache placement in MEN for mobile uses. Its focus is to formulate the problem as an optimization problem solvable through machine learning techniques. It presents the development of algorithms for cache placement by utilizing the data storage and computing capabilities of mobile edge networks. Furthermore, it studies the impact of mobility input attributes on the cache placement performance and the relationship between them and system attributes' effect on the performance of cache placement. The main focus is on using machine learning and intelligent decision-making techniques to implement latency-efficient cache placement algorithms.

1.3 Contributions

This thesis shows the effective use of cache placement in mobile edge networks depending on mobility input attributes of the wireless network, content popularity, and the cache placement model. We analyze, predict, evaluate and explain the issues related to cache content placement performance and its impact on wireless network quality of service. The contributions of this thesis are:

- We introduce a mobile edge network model, including network, mobility, content popularity, transmission delay, and energy models. Finally, these models develop and solve the multi-objective cache placement optimization problem and support performance analysis and new algorithm design.

- We first propose a new formulation of the cache placement problem that considers four objectives to place contents in SBSs and UT caches. The multi-objective function sets the advantages of user mobility patterns to decide on each SBS and UT cache content. To study the effect of user mobility on the cache placement performance, we exploit how user speed, user mobility pattern through different paths within a cell, user contact probability with other users and SBSs, and communication range between previous users and SBSs locations along previously used paths, could affect the cache placement decisions.

- We formulate the cache placement problem as a weighted fusion decision with four objectives. Three of them related to user mobility computed from recorded data sets, and one about content popularity. The results describe the impact of user mobility attributes on increasing the cache hit rate, which decreases the latency of downloading the requested data contents. Furthermore, the results show the impact of user mobility attributes on reducing the consumed energy for transmitting the contents to the UT.

- Following the formulation of a multi-objective function that maximizes the cache hit rate, we propose to formulate the cache placement as a binary classification problem that is solved using machine learning techniques. The input attributes of four objectives form the caching decision space. We find the hyperplane that separates the space into two regions corresponding to the binary classification problem, to cache and not to cache.

- We propose a semi-supervised self-trained learning technique (SSST) to design a cache placement binary classifier in mobile edge networks. We first train a supervised classifier based on the logistic regression (LR) learning technique using a small number of labeled samples. Next, we use the label propagation algorithm to self-train the classifier and generate pseudo labels using the trained classifier to predict class labels for all the unlabeled data. Then, pseudo labels with the highest probability of being correct are added to the labeled training set. Finally, we retrain the classifier to predict cache contents for the labeled test dataset and use the results to evaluate the prediction accuracy.

1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 discusses mobile edge networks (MENs) main models, a literature survey of solutions for mobile edge computing and caching challenges in terms of energy and latency, mobile edge computing applications and services, and their target requirements. It introduces caching in MENs, caching schemes, and caching performance metrics. Comparisons between different caching techniques in wireless networks and MENs. Then, we summarize the challenges that are faced in the design of caching systems in MENs. Finally, we discuss various types of machine learning and intelligent decision-making techniques that can be used in the design of a caching system at the edge of the network. An illustration of the research in cache developments for wireless networks that apply intelligently and learning techniques (ILTs) in a specific domain in their design is also presented.

In Chapter 3, the proposed caching system model in MENs based on machine learning and intelligent decision-making techniques is presented. The main system models, components, and relationship of system models are defined. Problem formulation for latency efficient and energy-efficient caching optimization are discussed. Illustrative examples for possible user requests, access, and delivery are also discussed.

In Chapter 4, a new formulation of the cache placement problem is investigated that considers four objectives to place contents in SBSs and UT caches. The multi-objective function sets the advantages of user mobility patterns to decide on each SBS and UT cache content. The problem is formulated as a weighted fusion decision with four objectives, three of them related to user mobility computed from recorded data sets and one objective related to content popularity.

In Chapter 5, we propose a new cache placement algorithm formulated as a binary classification problem (to cache and not to cache) based on user locations, contact probability, communication range, contact duration, and content popularity. Artificial neural networks (ANN), support vector machine (SVM), and logistic regression (LR) are used to model cache placement decisions. We investigate the characteristics of the input features (attributes) and the properties of these characteristics that affect supervised machine learning approaches. The performance of new cache placement models is investigated using supervised learning techniques by placing the proposed models in work with different system parameters. These parameters are varied to study the sensitivity of classification decisions with the change of system parameters.

In Chapter 6, we develop a semi-supervised self-training (SSST) classification model for cache placement problem. We assess the proposed SSST algorithm through experiments with datasets on different learning techniques. The performance comparison of different machine learning models was carried out with the same datasets. The test was made by changing one of the system parameters, fixing the other parameters, and computing the hit rate to investigate the sensitivity of the classification by the changes in the environmental parameters.

Finally, Chapter 7 presents the conclusion and future work.

1.5 Summary

The first chapter presents an overview of cache placement in mobile edge networks and the main objectives of this thesis. This is followed by the research objectives, research questions, and significance of the study. Finally, the chapter concludes with the organization of the thesis. Figure 1.1 shows the block diagram flow chart of the thesis outline.

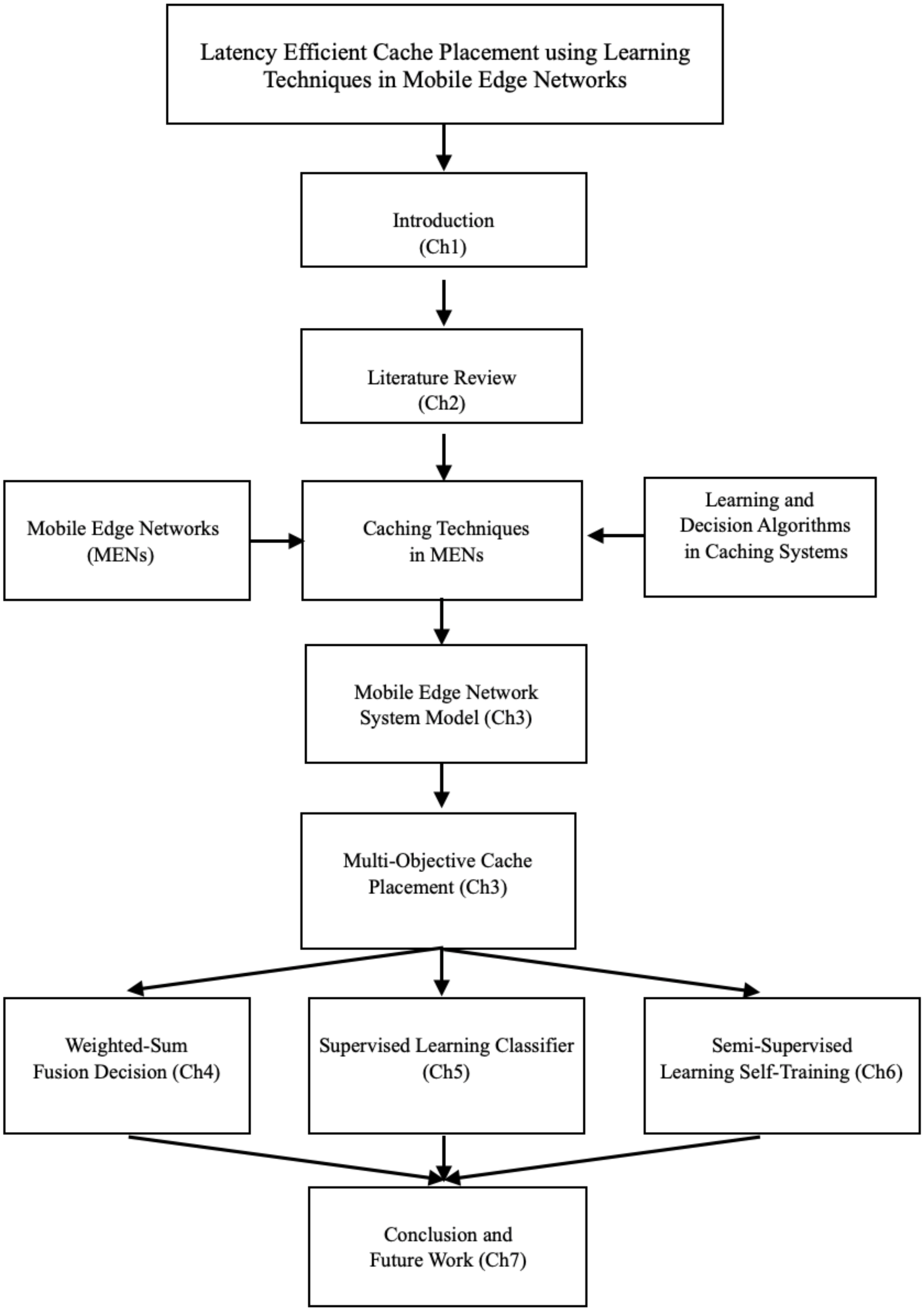

Latency Efficient Cache Placement using Learning Techniques in Mobile Edge Networks
Introduction (Ch1)
Literature Review (Ch2)
Mobile Edge Networks (MENs)
Caching Techniques in MENs
Learning and Decision Algorithms in Caching Systems
Mobile Edge Network System Model (Ch3)
Multi-Objective Cache Placement (Ch3)
Weighted-Sum Fusion Decision (Ch4)
Supervised Learning Classifier (Ch5)
Semi-Supervised Learning Self-Training (Ch6)
Conclusion and Future Work (Ch7)

Figure 1.1: Block diagram flow chart of thesis outline.

Chapter 2

Literature Review

2.1 Introduction

The increasing growth in mobile data traffic and new mobile applications lead to limitations on end-user demands and communications at mobile devices. For example, end-users require quality of services (QoS), service reliability, service availability, lower latency, and efficient energy usage. To overcome limitations such as computation capabilities, storage capacity, latency, and energy consumption, a new wireless network paradigm is needed [6]. Mobile edge network (MEN) architecture has been presented as a promising solution for future wireless networks. Proposals for MEN architecture have been presented from industry and academia. They are evolved from mobile cloud computing by utilizing the computing power and data storage away from the mobile devices into the cloud. MEN architectures are summarized in four main models depending on their services and operations. They are mobile cloud computing (MCC), mobile edge computing (MEC), fog computing, and caching [7].

The fundamental concept of MEN is to make network contents, services, and resources closer to the network edge. This can be implemented through the architecture design of MEN that deploys flexible computing and utilizes storage resources at the mobile network edges. Table 2.1 illustrates the network layers for the main four MEN architectures [7].

MEC is a network architecture concept standardized by the European telecommunications standards institute (ETSI) and the industry specification group (ISG). MEC was acknowledged as a prime emerging technology for 5G networks [8]. At the edge of the network, the IT service environment and cloud-computing capabilities are provided within the radio access network (RAN) [9].

Cisco proposed fog computing as an extension of cloud computing to wireless network edges. The aim is to accommodate the Internet of Things (IoT) applications closer to users. At the same time, fog computing nodes are distributed in a wide area and collaborate among multiple end-users to provide processing and storage [10].

Academic researchers at Carnegie Mellon University proposed a cloudlet that extends the mobile device-cloud architecture. Cloudlet is defined as a resource-rich computer or cluster of computers that are connected to the Internet, and nearby mobile devices [11]. Both Wi-Fi and mobile networks are deployed to provide near-real-time applications and handoff of virtual machine images among edge nodes when a device moves. In addition, Cloudlet can reduce the end-to-end latency between the mobile device and the cloud [12].

Future mobile networks will challenge being heterogeneous because of different types of base stations and the massive and diverse contents. This enables the MEC to proactively exploit many low-cost storage devices at various network edges to cache popular contents during off-peak periods. Caching can be deployed at different levels in mobile networks instead of fetching them from the core network [7]. Reducing the number of network hops between the location of the contents and the user requesting the contents will reduce the latency for retrieving the contents [13]. The caching places which are considered as caching levels in MEN architectures are a macro base station (MBS), a small base station (SBS), and user terminals (UTs), allowing for the device to device (D2D) communications.

2.2 Mobile Edge Computing and Caching

This section presents the benefits of MEC and caching as a technology for the future wireless network. There is a number of research work done to show the efficiency of MEC in terms of energy consumption, latency requirements, and storage capacity for different applications and services.

According to Cisco visual networking index on the global mobile data traffic forecast updated report for the years 2016 to 2021, there will be 11.6 billion mobile devices, and the 5G connection will generate 4.7 times more traffic than the average 4G connection by 2021 [45]. In addition to an expansion in the number of user terminals and a considerable traffic increase, the develop-

Table 2.1: Different mobile edge network architectures.

MEN Architecture	**Hierarchy (Layers)**	**References**
MEC	Core network, MEC server, and mobile devices	[14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], and [25]
Fog Computing	Cloud, fog, and mobile devices	[26], [27], [28], [29], [30], [31], [32], [33], and [34]
Cloudlet	Cloud, Cloudlet, and mobile devices and IoT	[35], [36], [37], [12], [38], [39], and [40]
Edge Caching	Core network, MBS, SBS, and D2D	[41], [42], [25], [43], and [44]

ments of new applications demand high computation capabilities, low latency, energy consumption, and increased storage capacity. These challenges require future mobile networks to satisfy these constraints [46].

The main idea of MEC is to bridge the gap between the capability limitations of storage and computation in user terminals and their increasing demands. It is done by placing storage and computation resources at the edge of the network closer to user terminals. Thus, MEC can reduce latency and energy consumption. In addition, there are various techniques in the literature proposed to process data locally at the edge of the network and accelerate data streams, which will reduce the traffic bottleneck toward the network [47].

Table 2.2 and Table 2.3 present an overview of the main research work areas in the literature that proposed possible solutions for mobile edge computing and caching network in terms of energy and latency, respectively. The research areas can be categorized into the following main streams:

1. **Computation offloading**: The advances in computing technology and various applications that require high computation power and resources to run complex programs have increased lately. These applications use wireless networks and run on mobile devices with limited capabilities to support the needed resources [48]. One of the solutions to solve this problem is computation offloading. In computation offloading, the mobile devices transfer tasks to an

external edge cloud and receive the results from the edge cloud. Offloading increases mobile terminal capabilities by migrating the computation to more resourceful computers (servers) at the edge of the network [49].

2. **Task caching**: Computation offloading considers computing capabilities at the edge of the network by assuming enough hardware and software resources to execute the tasks. Enough storage capacity for computation offloading is another critical challenge that faces future wireless networks. In [50], the authors proposed task caching, which refers to caching the task (application computation task) and its related data at the edge of the network.

3. **Content caching**: Contents requested by end-users in massive multimedia services over mobile networks face network capacity limitations and increase backhaul link loads. Requesting the same content by different users also causes network congestion and a waste of network resources. The development of mobile edge caching techniques is another solution for wireless networks. Content caching of the most popular files can prevent duplicate transmission of the same contents and improve end-user QoS. Quality of service can be improved since downloading the contents from network edge (for example, base stations or end-user terminals) reduces latency compared to downloading the contents from Internet content providers (core network) [7].

4. **Resource allocation**: With multiple user terminals, MEC servers have much fewer computational resources. Therefore, one of the main issues in the design of MEC is to consider resource allocation. It is the process of allocating finite radio and computational resources to multiple mobiles under resource constraints. There are two categories of resource allocation schemes for MEC: centralized and distributed. The centralized resource allocation, the MEC server is responsible for all mobile information, makes the resource allocation decisions, and sends the decisions to mobile devices. While in distributed resource allocation, many techniques, including game theory and decomposition techniques, are used to develop distributed algorithms [51].

5. **Multicast caching**: To reduce the load of wireless links in traditional unicast connection-based transmission and to avoid transmitting the same file multiple times to multiple users,

multicast caching is proposed for base stations in 5G mobile networks [52]. In multicast caching, the popular content is brought close to the users. The optimization objectives are to minimize the average latency for all content requests and minimize the average energy consumption [53].

6. **Service chain management**: Service chaining policy refers to the term that describes executing multiple service functions in an ordered list to guarantee performance and security requirements [54]. In MEC networks, lightweight data centers can be used at the edge of the network. In these centers, operators deploy service chaining to steer traffic by managing a set of service functions. Service chaining is realized by using the software-defined network and network function virtualization technologies. Service chain management can reduce network latency by offloading the workload or bandwidth from the core network service [55].

Mobile edge computing and caching is considered a promising solution that supports many emerging applications and services with specific constraints of latency, energy, and reliability [56]. In the following section, the applications and services designed to use the MEC architecture characteristics will be presented. In addition, latency which is considered as the primary critical constraint to these services will be analyzed.

Table 2.2: Overview of techniques based energy efficient in mobile edge computing and caching.

Area	**Ref.**	**Approach**
Computation offloading	[48]	Markov decision process (MDP)
	[57]	Minority games theory
	[58]	Multi-label classification and deep supervised learning
	[59]	An approximation collaborative computation offloading
	[60]	Karush Kuhn Tucker Lagrangian multiplier method
	[61]	Dynamic sequential game theory
Task caching	[50]	Mixed integer non-linear programming
Content caching	[62]	Poisson point processes modelling

Continued on next page

Table 2.2 – continued from previous page

Area	Ref.	Approach
	[63]	Lagrange multiplier and duality
	[64]	Social-tie factor modelling
	[65]	Dual decomposition
Resource allocation	[66]	Mixed discrete-continuous optimization
	[67]	Dual-decomposition method and alternating direction method of multipliers
Multicast caching	[68]	Graph theory
	[69]	Distributed potential game model
	[70]	Clustering based on game theory

Table 2.3: Overview of techniques based latency efficient in mobile edge computing and caching.

Area	Ref.	Approach
Computation offloading	[71]	Heuristic search, reformulation linearization technique, and semi-definite relaxation
	[72]	Lyapunov stochastic optimization
	[73]	Lyapunov optimization
	[61]	Dynamic sequential game theory
	[74]	Markov decision process
Service chain management	[55]	Hash-based group management
Content caching	[75]	Transfer learning
	[41]	Assessment tests of caching solution based on open source open air interface
	[56]	Submodular optimization

Continued on next page

Table 2.3 – continued from previous page

Area	Ref.	Approach
Resource allocation	[76]	Auction theory
	[77]	Nash bargaining game

2.2.1 MEC Applications and Services

2.2.1.1 Services Categories

The characteristics of MEC networks offer opportunities to a wide range of applications and services to be beneficial to users, operators, and service providers [19]. Although low latency and high data rates are essential to many applications, in some 5G services and applications, they must have as low as 1 ms latency. Service scenarios supported by 5G can be characterized by the following categories:

1. **Consumer-oriented services** : Computation offloading in MEC enables users to run new applications on their end terminals. These applications tend to take full advantage of new technologies that satisfy their restricted requirements. Examples of consumer-oriented services are listed below:

   - **Augmented Reality (AR) and Virtual Reality (VR)**: Requires tracking of moving objects in real time, such as driver-assistance system, telemedicine, flight training, remote education, city, sport or musical event guide, or gallery guide [9], [78], [46], and [79].

   - **Image/Video Editing**: Requires considerable computational power and energy. With increasing performance requirements and image/video editing algorithms complexity, and the mobile terminals short battery lifetime, the users need to offload these tasks to be computed and implemented at the network edge [80], [19], [81], and [82]

   - **Gaming**: Considered as a delay sensitive application that requires computing and storage resources for on-line data processing and reliable decision making [83], [8], [47], and [84].

- **Web Accelerated Browser**: Accelerate the web applications execution by offloading intensive computation from the end devices. Most browsing functions such as web content evaluation, optimized transmission, and application execution. Besides offloading computation, this service aims to maximize the battery life of end-user devices [85], and [86].

- **Image/Voice/Video Recognition**: Object detection, recognition, and classification have many applications, from face recognition, vehicular license plate recognition, voice/images classification, and home security. These algorithms implement complex computations that can be offloaded away from the capturing devices to simplify the design of these devices. Executing applications at network edges, the network can avoid congestion caused by video stream uploading and result in lower latency in the overall application execution [87], [51], [88], and [89].

2. **Operator and Third Party Services**: MEC offers new ways for information technology to store, process, and analyze data and create new services that solve complex systems [90]. Operators of third parties start to recognize MEC technologies and transfer mobile network functionality to the edges of the network [19]. Examples of this type of service are given below:

   - **Internet of Thing**: IoT connects everyday objects such as physical devices, vehicles, home appliances, and other embedded items with electronics, software, sensors, actuators, and network connectivity and brings the connected world into reality. IoT requires intensive computation, massive storage, and data sharing capabilities. MEC enables the environment for these requirements to allow objects to connect, execute tasks through computation offloading, and exchange data through caching on edge, [91]. Example of these services are: e-health [92], automatic driving [93], industrial automation [94], etc [95] and [96].

   - **Big Data**: It requires connecting billions of devices, collecting a massive amount of information, analyzing the collected data, and executing intelligent techniques to understand the datasets. It requires implementing deep learning or similar techniques to extract knowledge from a huge amount of data efficiently. MEC architecture provides the infrastructure to collect, store, and execute complex algorithms at the edge of the network closer to user terminals [97]. Examples of services that can satisfy their requirements through adopting MEC architecture are: smart cities infrastructures [98] and transportation [99], solving traffic congestion [100], waste management [101], electronic vehicle charging [102], etc.

3. **Network Performance and QoS Improvements Services**: MEC assists network performance improvement and/or improving quality of service (QoS) through radio analytics applications and RAN-aware content optimization [19]. Examples of such services are:

   - **Radio/Backhaul Optimization**: Radio/backhaul network traffic requirements can be monitored, collected, analyzed in MEC. Then, running an optimization application on MEC that can optimize overall network performance and improve QoS [67].
   - **Traffic Monitoring/Shaping**: This type of service is used to control and re-route the traffic in the MEC and perform the coordination between radio and backhaul links [67].
   - **Content Caching**: The considerable increase in mobile data traffics, which leads to increased data requests from the core network, emerges MEC design to include new services that improve network performance and QoS by adding caching technology at the edge of the network. Caching the most popular contents during off-peak traffic periods is a solution to minimize duplicate transmission of the same files requested by user terminals in temporal and spatial demands. Also, caching contents at the edge of the network through wireless connection reduces the download latency of the same data downloaded through backhaul links from core network [75] and [7].

2.2.1.2 Applications and Services Target Requirements

The previous section presented the most popular applications and services that current research progresses promise to include in future wireless networks. We characterized them according to the benefits from these services. The target requirements for these applications and services can be measured in terms of bandwidth, latency, reliability, storage, and mobility. Table 2.4 illustrates the target requirements for different services and applications [103], [46], [104], and [105]. The reliability can be defined as the probability of successful transmission of a certain amount of data from one peer to another peer within a given deadline or time frame [106]. Additionally, storage indicates if the target service requires storage capacity for its manipulated data, and mobility indicates if the service needs processing of user terminals locations. Based on the requirements given in Table 2.4, latency is highly critical in most of these applications and services.

In the next section, the sources of the latency in a mobile network and current approaches to achieve low latency are discussed.

Table 2.4: Target requirements for different services and applications in future wireless networks.

App./Services	**Bandwidth**	**Latency**	**Reliability**	**Storage**	**Mobility**
AR/VR	1 Mbps - 16 Mbps	$< 1\text{ms} \leq 10^-5$	High	High	High
Image/Video Editing	10 Mbps	$(5 - 10)$ ms	$\leq 10^-7$	High	High
Gaming	10Mbps	$< 1\text{ms}$	$\leq 10^-5$	High	Low
Image/Voice /Image Recognition	1 Mbps - 1Gbps	¡1ms	$\leq 10^-5$	High	High
IoT	$(1 - 100)$ Mbps	< 20 ms	$\leq 10^-9$	High	High
Big Data	$(1 - 100)$ Mbps	< 20 ms	$\leq 10^-9$	High	Low
Radio/Backhaul Optimization	> 1 Mbps	$(100 - 1000)$ ms	$\leq 10^-7$	Medium	Low
Traffic Monitoring /Shaping	$(1 - 10)$ Mbps	1000 ms	$\leq 10^-9$	Medium	High

2.2.2 Latency Sources in Mobile Networks

The total one way latency is the delay of a packet transmission in a mobile network which is contributed by the RAN, backhaul , core network, and data center/Internet. Figure 2.1 illustrates the contribution of latency through one way packet transmission. Total latency can be written as follows [46]:

$$T = T_{Radio} + T_{Backhaul} + T_{Core} + T_{Transport}, \tag{2.1}$$

where

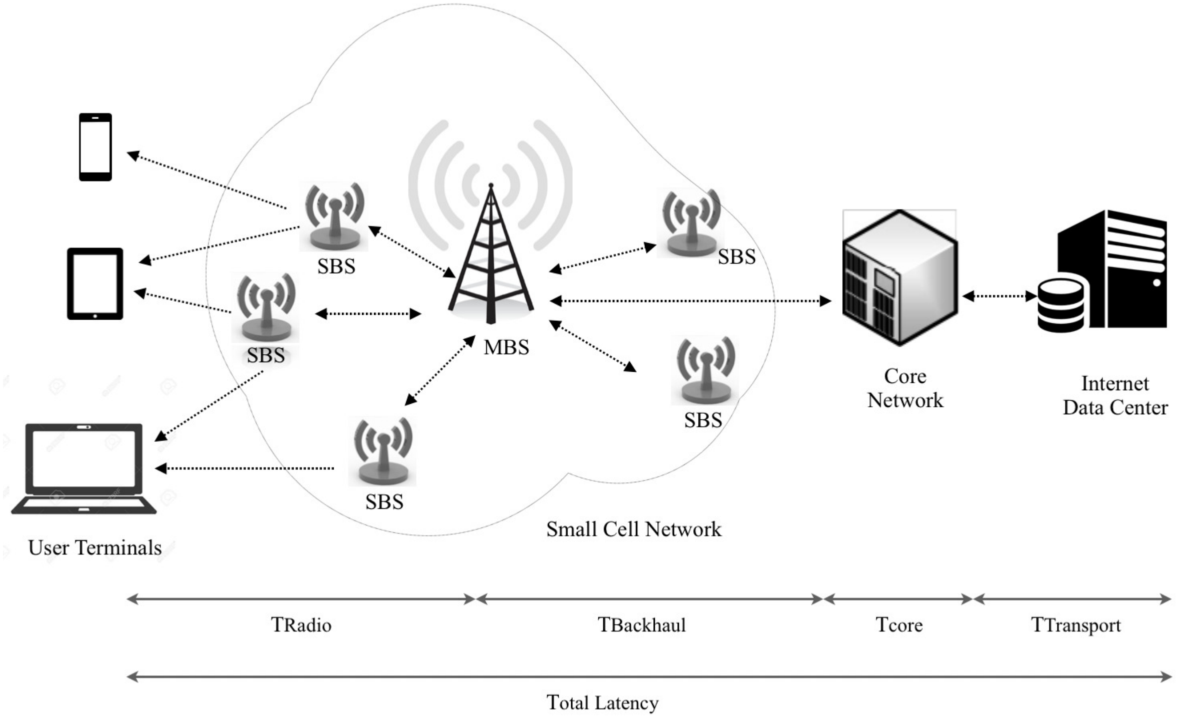


Figure 2.1: Total one way latency contribution in packet transmission.

- T_{Radio} is the transmission time from the user terminal and eNB,
- $T_{Backhaul}$ is the time to build the connection between eNB and the core network,
- T_{Core} is the core network processing time,
- $T_{Transport}$ is the data communication time between the core network and the data center/Internet.

The total end-to-end latency can be computed approximately as $2 \times T$. In the following section, an overview of the approaches to achieve low latency is presented.

2.2.3 Approaches to Achieve Low Latency

To achieve low latency, we need to enhance the four main latency contributors. In literature, various techniques have been proposed to minimize each one of those latencies [46]. Figure 2.2 illustrates the latency sources and the possible approaches for achieving low latency. These approaches can be categorized as:

- RAN solutions such as frame/packet structure, advanced multiple access techniques, waveforms designs, modulation, channel coding, etc.

- Caching includes device caching, small base station SBS caching, and macro base station MBS caching.

- Core network development includes new core network entities such as software-defined network (SDN), MEC, and network function virtualization (NFV).

- MEC that enables caching such as service caching and data caching [51].

The focus of this research (the shaded area in Figure 2.2) is to find new solutions related to different levels of caching (device caching, SBS caching, MBS caching) for MEC networks based on learning techniques.

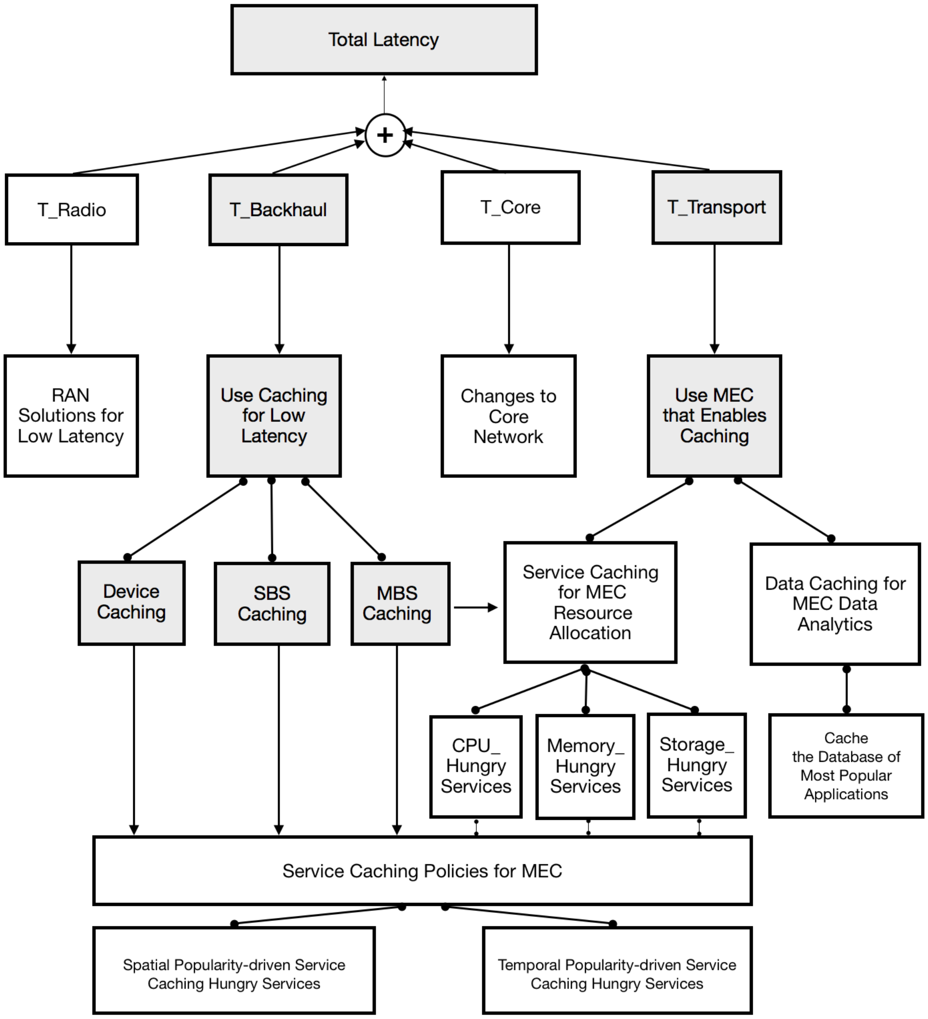

Total Latency
+
T_Radio
T_Backhaul
T_Core
T_Transport
RAN Solutions for Low Latency
Use Caching for Low Latency
Changes to Core Network
Use MEC that Enables Caching
Device Caching
SBS Caching
MBS Caching
Service Caching for MEC Resource Allocation
Data Caching for MEC Data Analytics
CPU_ Hungry Services
Memory_ Hungry Services
Storage_ Hungry Services
Cache the Database of Most Popular Applications
Service Caching Policies for MEC
Spatial Popularity-driven Service Caching Hungry Services
Temporal Popularity-driven Service Caching Hungry Services

Figure 2.2: Latency sources and approaches for achieving low latency.

2.3 Caching in Mobile Edge Networks

Caching is presented as a promising technology solution to the challenges facing future wireless networks and subsequent generations. Having many requests from connected users during peak traffic hours, the massive increase in the amount of data exchanged within the network, and the insufficient capacity of backhaul links can result in a long delay. The applications and services described in the previous chapter required low latency communication. Caching concepts, different caching schemes, comparisons between different caching techniques for different network topologies and mobile edge computing (MEC) networks, and the limitations of caching techniques are presented and discussed.

Caching refers to storing a certain amount of data at network edges, base stations, and user terminals in the case of communication systems. Caching ability is measured by the size of stored information whose unit is byte [107]. Caching supports the massive traffic in mobile networks during peak time by enhancing information delivery capability and improving network throughput and QoS for users over the long term. Enhancing user experience by improving throughput, latency, and link capacity yields more significant network operators' profits. According to CISCO, by 2021, video traffic will be 82 % of all consumer Internet traffic, and the estimated total size would be exceeding 3.3 Zettabyte per year [45].

The major applications and services that were discussed in Chapter 2 are based on audio and video streaming. Mobile devices provide limited computation resources and storage capacity that will not be able to meet the requirements of these services (see Table 2.4). The powerful technologies available in MEC are needed to reduce and/or offer higher computation capabilities on network edges. To reduce the latency, caching most popular videos at network edges through caching functionality such that users effectively obtain their requested contents directly from caches instead of using backhaul connections [107] and [108].

In each cell network, there is one MBS and several SBSs that are located close to the user terminal (UTs). Although spectral efficiency will be enhanced because of increased spatial reuse, such enhancements lead to intense interference and an increase in backhaul cost. For ultra-dense small cell networks, inter-cell interference will highly reduce the spectral efficiency. SBSs share the payload and serve multiple users based on the cooperative multi-point (CoMP) technique to solve cross-cell interference. Due to payload sharing between SBSs, the backhaul load will increase and scale up with the increase of SBSs. A potential solution is by serving users' requests locally from caches located in SBSs and UTs that can eliminate the backhaul cost [109].

2.3.1 Types of Caching Schemes

The places that can be used to cache most popular contents within MENs are shown in Figure 2.3 and described below:

1. **UT caching:** Exploiting the storage resources in UTs is one of the key technology in 5G networks [7]. In addition, caching in user devices allows the improvements of caching strategies to allow D2D communications.

2. **SBS caching:** Each cell in MEN employs a large number of SBSs. SBS includes a higher storage capacity than UT cache capacity. They are closer to the end-users and usually provide higher data rates [7]. Therefore, utilizing caching in SBSs is a promising solution to improve QoS in next-generation heterogeneous networks.

3. **MBS caching:** MBS covers a larger area within the heterogeneous network and can serve more users. In addition, the storage capacity in MBS is higher than other caches within the cell, leading to a higher probability of finding the requested file (hit).

To design a cache system, there are two separate phases: (1) cache placement phase and (2) content delivery phase [110]. The contents are stored in cache memories within UTs, SBSs, and MBS in the cache placement phase. Then, when users request the files during the delivery phase, the contents are delivered from caches according to the caching delivery approach used.

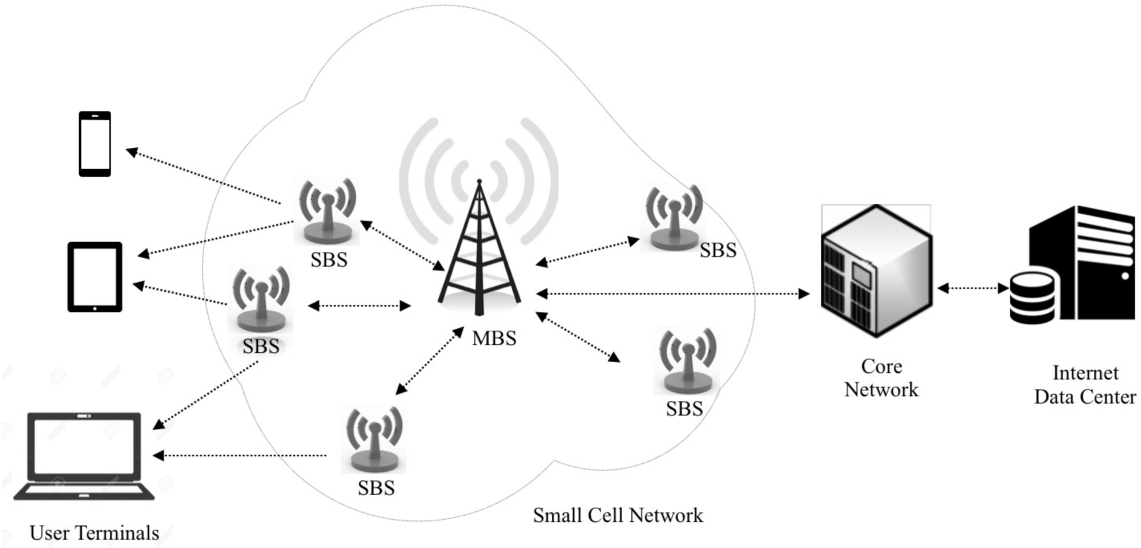


Figure 2.3: Architecture of mobile edge caching network.

2.3.2 Caching Performance Metrics

This research focuses on the performance metrics related to QoS and the energy efficiency of enabling caching in MEC networks.

1. **Latency** In caching systems, the average content delivery latency is taken by a request delivered to the end-user. There are different types of latency generated in the caching system during the delivery phase, depending on the source of the contents (user device, SB, or MBS). Therefore, the latency metric is also referred to as delay, download time, or delivery time.

2. **Cache Hit Rate** Cache hit happened when the requested content found a copy in the cache. Then, the cache hit rate indicates the number of cache hits divided by the total number of requests. Another term can be derived from cache hit rate, which is the complement of cache hits called cache miss rate. Cache miss rate indicates the number of cache misses divided by the total number of requests. Thus, the total number of user requests equals the total number of cache hits and cache misses. Depending on the cache type, cache hit rate can be referred to as a local cache hit rate (UT cache hit rate), BS cache hit rate, and MBS cache hit rate, respectively.

3. **User Satisfaction** When the request arrives at the SBS, and the content delivery starts, if

the rate of content delivery is equal or higher than the bit rate of the content at the end of service, this request is said to be satisfied. If the rate of the content delivery does not equal or less than the bit rate of the requested contents, then the download of the requested delivery will be interrupted, and the user will have less QoS [111].

2.3.3 Caching in Wireless Networks

Most research work in the literature cache either uncoded contents or uncoded parts of files during the placement phase. In the case of coded file caching, the base station broadcasts the coded files (linear combination of multiple files) to user terminals during the content delivery phase. Then, the users can decode their files simultaneously [112].

During the delivery phase, the cache memory contents of the requested user are updated to store new files. There are different techniques to implement cache replacement. Researchers have proposed caching algorithms for wireless systems that range from simple algorithms to more advanced intelligent methods. These algorithms are divided into two main streams. The first is cache replacement algorithms based on prior knowledge about content popularity, while the second is cache replacement without prior knowledge about content popularity [113].

Table 2.5 shows a comparison of different caching techniques in literature in terms of their dependency on content popularity, online learning, training phase, context-awareness, social awareness, mobility awareness, and prediction ability.

In [114] and [115] least recently used (LRU) and least frequently used (LFU) algorithms are used, respectively. These techniques are simple cache replacement algorithms that do not consider future content popularity and update the caches continuously during the delivery phase. In the LRU algorithm, the cache includes an ordered list which is updated to follow the recent access of all cached contents. When the cache is full, the new content is placed in the least recently accessed cache content. In the LFU algorithm, the cache includes an ordered list that is updated to follow the access numbers to all contents. When the cache is full, the new content is placed in the least frequently used cache content.

The content of cache can be changed based on prior knowledge of content popularity. The research work in [116] and [117] use popularity statistics of different video files modeled using a Zipf distribution. The cache replacement algorithm by tracking variations in the popularity distribution and updating cache content at user terminals and collaborative device-to-device communication is combined with increasing the efficiency of content delivery. There is a trade-off between having an optimal content replacement that predicts future requests efficiently and the speed of computing content popularity. Also, in these methods, there is no personalization of user context and preferences.

In [118] and [119], authors exploited storing of video files closer to users in Femto caches. They formulate the problem to increase the throughput by unloading the traffic from the main cellular network. The work presented in [2] proposes caching and multicasting techniques, and caching aims to allow popular content files at network edges to shorten the distance between contents and requesters. At the same time, multicasting aims to serve identical requests happening at nearby locations through common multicast streaming.

The exploiting of proactive caching contents based on file popularity and correlations among users and files patterns are proposed in [3]. Files can be proactively cached during off-peak demands using a machine learning algorithm and collaborative filtering with context awareness. The procedure aims to predict the set of influential users and social structure and to proactively cache strategic content on those UTs to utilize device-to-device communications. This approach requires a training set of known content popularities and can learn to decide which content to place in the cache during a training phase.

In [120], the cache strategy is modeled in a heterogeneous small cell network using a reinforcement learning-based coded caching framework. Authors have designed an optimal cache placement policy that uses the learned file popularity to find the optimal cache contents. The cache placement policy takes into account the users' connectivity to the SBS. At regular intervals, the cache pre-fetches segments of the popular files to serve users' requests.

The caching algorithm is presented in [121] based on contextual multi-armed bandits optimization. In this algorithm, the SBS updates its contents regularly by observing the demand of cached files and learning the popularity profile contexts over time. The objective of the multi-armed bandit optimization is to maximize the number of cache hits. The objective of the multi-armed bandit optimization is to maximize the number of cache hits.

In [122], a different extension of the multi-armed bandit framework is proposed. In this framework, the authors have exploited the topology of user connections by incorporating coded cache contents. An optimal cache content placement strategy can be devised based on observations of instantaneous demands that assume content popularity distribution.

While previous algorithms do not consider the future prediction of popular contents in the design of cache replacement algorithm, the work in [123] and [124] aim to learn popularity trends. Their works include the design of context-aware proactive caching. There is no prior knowledge about content popularity in [123], while in [124] the cache replacement method learns the popularity of contents and uses it to determine which contents to place in the cache and which contents to evict from the cache.

2.3.4 Comparison of Different Caching Techniques in MEC

Different studies formulate the caching problem at the edge of the network. These proposals examine the problem from different perspectives. In each study, the optimization problem is developed based on input attributes manipulated by the optimization algorithm and the scheme of caching used in the model. The performance indices in these proposals are overall delay, user satisfaction ratio, offloading probability, and total throughput. However, they have one general objective: redirecting user requests from the expensive and limited backhaul links to local cache storage at the edge of MEC networks. Table 2.6 illustrates a comparison between different caching techniques in MEC networks.

In [128], the authors studied the association between UTs and SBSs in small cell networks. Based on file availability in SBS and the backhaul congestion state, the SBSs decide which users they should serve. The problem is formulated using one-to-many matching game theory. In [129], two proactive caching scenarios are examined. The goal in both cases is to keep the user satisfaction ratio above the required limit. In the first case, the contents are cached proactively at the SBSs during low-peak demands. The cache procedure is built on a supervised machine learning algorithm using regularized singular valued decomposition (SVD). This technique includes two parts. Training the input matrix representing the users-to-files rating association and predicting what files each user will request (file popularity matrix). In the second case, the contents are cached proactively in users' devices. The centrality metric measures the social influence and its connection with other nodes (social community). Then, the k-means clustering method forms a set of influential users within a community (users' social ties). The proactive caching procedure considers each user's number of times each file is downloaded to form a user-to-files history matrix. The beta distribution is assumed to denote the probability that a given user selects content. The popular contents cached in influential users' devices are chosen based on the Chinese restaurant process (CRP). By caching at UT and enabling D2D communication, the load on SBS and backhaul loads are reduced [129].

In [130], optimal two one-tier caching placement is presented based on the difference of convex programming. The objective of the optimization problem is to maximize the offloading probability. The offloading happens in three cases:

1. Self offloading when the requested contents are found on UT (local cache),
2. D2D-offloading when the requested contents are found from near devices, and
3. SBS-offloading when the requested contents are found in near SBS.

Table 2.5: Comparison of different caching techniques.

Cache Technique	Content Popularity	Online Learning	Training Phase	Context aware	Social Aware	Mobility Aware	Predict Ability
LRU and LFU [114] [115]	Yes	No	No	No	No	No	No
PopCaching [124]	Yes	Yes	No	No	No	Yes	No
Femtocaching (coded cache)	Yes	No	No	No	No	No	No
Femtocaching (uncoded cache) [116]	Yes	No	No	No	No	No	No
QoE-oriented resource efficiency [117]	Yes	No	No	No	No	No	No
Multicast aware caching [2]	Yes	No	No	No	No	No	No
Proactive caching Based ML [3]	No	No	Yes	Yes	Yes	Yes	No
Distributed caching [120]	Yes	Yes	No	No	No	No	No
Context-aware cache [113] [121]	No	Yes	No	Yes	No	Yes	No
Optimal uncoded cache [122]	Yes	Yes	No	No	No	No	No
Decentralized cache [125]	Yes	No	No	No	Yes	Yes	No
Stackelberg game caching [126]	Yes	No	No	No	No	No	No
Popularity prediction caching [127]	Yes	No	Yes	Yes	No	Yes	No
Cache-aware user association [128]	Yes	No	Yes	No	No	No	No
Social and spatial caching [129]	Yes	No	Yes	Yes	Yes	Yes	No
Optimal caching placement [130]	Yes	No	No	No	No	No	No
Clustered D2D caching [131]	Yes	No	No	No	No	No	No
Joint caching [132]	Yes	No	No	No	No	No	Yes
Cell-site-aware caching [133]	Yes	No	No	No	No	No	No
Adaptive caching [134]	Yes	No	Yes	No	No	No	Yes
Mobility-aware caching [109]	Yes	No	No	No	No	Yes	No

Their results show that popular contents must be cached under relatively low node density while other contents must be cached evenly under relatively high node density.

In [131], the authors formulate the caching problem as a video recommendation system. Files and users are clustered depending on video files preferences. After clustering all files and users, the cache scheme is formulated in two phases: the D2D cooperative phase, where files are stored in UTs, and caching phase, where files are stored in SBS. The optimal caching is designed based on the greedy intra-cluster algorithm to obtain a minimum average file download time. The results show that clustering files before applying the optimal caching algorithm can reduce the computational complexity of many involved users and files.

The work in [132] proposed joint caching, routing, and channel assignment for video files in a collaborative small cell cellular network. Their objective is to maximize network throughput by using a conflict graph to characterize the communication links interference. The optimization problem is modeled as a large-scale linear programming problem that is solved using the column generation method. The algorithm selects a subset of variables that have potential improvements to the objective function to minimize the optimization problem's complexity. The optimization problem is then divided into two sub-problems: restricted master problem (RMP) and pricing problem (PP). Their results show that the overall throughput of the video data delivered to users is considerably increased over the state-of-the-art Femto caching models.

In [125], proactive caching is designed based on traditional collaborative filtering by regularized singular value decomposition to estimate the popularity matrix. Then transfer learning is used to improve the estimation accuracy by transferring and learning hidden knowledge from other domains such as social networks. Finally, an optimal caching strategy is implemented as a distributed iterative algorithm to update the cache. Results show that the user satisfaction ratio increases with the number of SBS compared to other caching approaches.

Authors in [126] investigate proactive caching for service providers to reduce redundant backhaul transmission to edge nodes (ENs). The Stackelberg game is used to formulate the problem, and it was decomposed into two sub-games, a storage allocation game and a number of user allocation games. The service provider is modeled as a leader that decides the prices for the storage and backhaul resources on ENs. ENs are modeled as the followers. Their results show lower backhaul resources can be achieved with proactive caching-based game theory than centralized popularity-based and random caching. The authors discussed the complexity and scalability of edge caching in wireless communication networks where there will be a large number of ENs, users, and service demands and will involve a considerable amount of data. The complexity is defined as the number of iteration steps of the caching algorithm, and the amount of information exchanged between network edges. The performance of caching algorithm with the increase of network size is addressed as the scalability of the caching algorithm.

The popularity estimation used in caching techniques for video files in work presented previously is based on user request probability or the number of views of videos. The work in [127] computes video popularity for published and unpublished videos using intelligence-based content-awareness. The prediction of video popularity enhances cache placement decisions as well as the QoS in cellular networks.

In [134], an adaptive caching scheme is proposed that takes into account users' behavior, content popularity, request statistics from users, and operating characteristics of the cellular network. The network operating characteristics include network topology, link capacity, routing strategy, cache size, and energy usage to read/write files from hardware storage (called cache deployment cost). In addition, content popularity is predicted using the extreme learning machine (ELM) based on contents features. The features of the contents are computed using a combination of human perception models and network parameters. The adaptive caching scheme uses a mixed-integer linear program (MILP) to cache the results of popularity estimators. The caching decision is made while the network is not heavily utilized. Without affecting network quality of service, popular content can be transferred between BSs.

The impact of mobility awareness in the cache placement algorithm is discussed in [109]. The authors formulate the problem of caching coded segments at BSs and UTs, considering user mobility and the content amount per transmission. The optimization problem is formulated as an integer programming problem that can be solved by sub-modular optimization. Figure 2.4 illustrates different caching techniques' objectives in MEC.

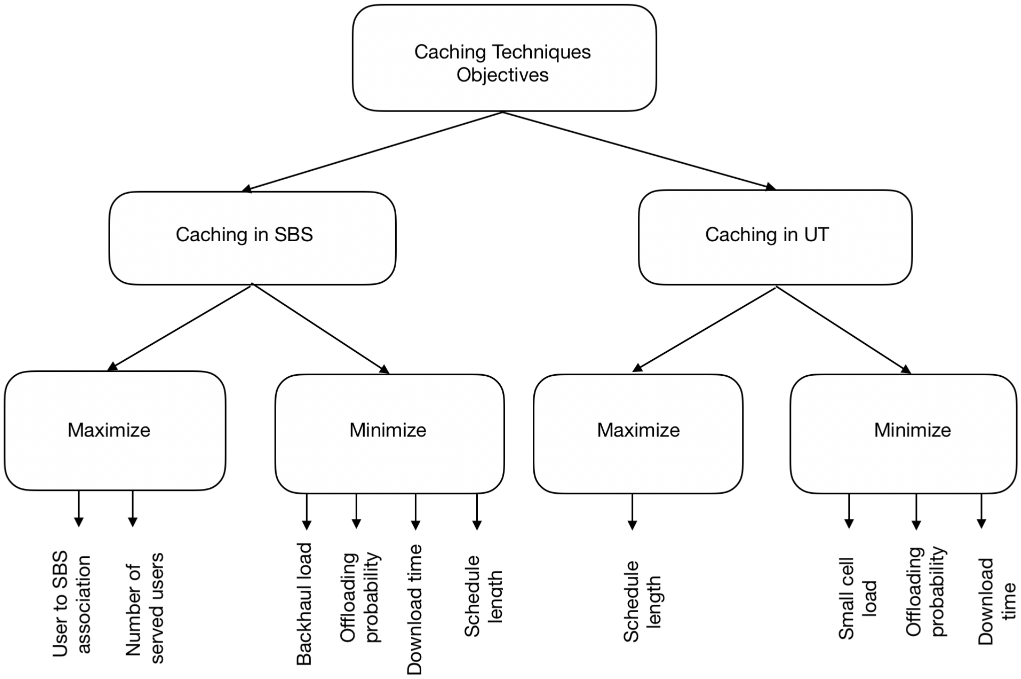


Figure 2.4: Different caching technique's objectives in MEC.

Table 2.6: Comparison of different caching techniques in MEC.

Ref.	Methodology	Caching Location	Input Attributes	Objective	Constraints	Performance Index
[128]	Game Theory	SBS	Content availability Delivery data rate	Maximize UT to SBS association utility	Backhaul capacity Storage size	Overall delay
[129]	Regularized SVD	SBS	User to files rating	Minimize backhaul load	Backhaul rate Cache storage	User satisfaction ratio
[129]	Eigenvector centrality K-means clustering Chinese restaurant process	UT	Users' social ties Users to files history	Minimize small cell load	SBS data rate D2D data rate UT storage	User satisfaction ratio
[130]	Difference of Convex programming	UT SBS	Content popularity D2D transmission range SBS transmission range Users and SBS densities	Minimize total offloading probability	UT storage SBS cache storage	Offloading probability
[131]	Greedy intra-cluster algorithm	UT SBS	D2D download time BS download time	Minimize total average file download time	storage capacity	Delay loss
[132]	Column generation method	SBS	User requests File popularity	Minimize schedule length	Cache capacity Received file fraction Files cached in SBS Channel capacity and active time	Total throughput
[125]	Regularized singular value decomposition based collaborative filtering	SBS	The popularity matrix	Maximize number of users served by neighbouring SBS	Cache size	User Satisfaction ratio
[126]	Stackelberg game	SBS	File demand probability EN storage cost Backhaul bandwidth Backhaul cost	Minimize the total backhaul resources Maximize EN utility	Cache Capacity SBS coverage	Total backhaul resources and Outage
[127]	Intelligence based content-aware	BS	Video popularity	Minimizes the backhaul load	Cache capacity	offload ratio
[134]	Mixed-Integer linear program	BS	Content popularity Cache deployment	Minimize content downloading delay	Initial file transferring cost Cache deployment budge	Predictive Type-I and Type-II error
[109]	Greedy Algorithm	BS UT	Set of files Set of coded segments Cache strategy matrix	Maximize cache hit rate	BS capacity UT capacity	Cache hit rate

2.3.5 Challenges in Designing Caching Techniques

Following the study of different caching techniques in literature, we can summarize the following challenges:

2.3.5.1 Content Popularity Modelling

To improve the performance of caching strategies, it is required to incorporate content popularity in caching decision making [124]. Table 2.7 shows the methods used to model content popularity in literature. In many works, the popularity of files is generally modeled using the Zipf distribution of all files. Zipf distribution gives a fixed popularity profile, and it is assumed that content popularity is known in advance. Based on Zipf distribution, a small portion of Internet content is highly popular while the rest is rarely requested [127]. In reality, content popularity needs to be estimated depending on a number of related factors and not only on the content popularity distribution. Examples of these factors are files' preferences, users' preferences, context, social network characteristics, previous requests, etc. Also, content popularity must not be fixed, and it is expected to change continuously with time, date, and location.

The approach in [135] assumes that the popularity of video contents is changing slowly, and the popularity distribution of all files can be considered as fixed and known before the cache placement algorithm. Therefore, they defined the popularity distribution of video files depending on a number of views vs. the rank of videos in terms of views.

In [121], context-dependent popularity profiles are learned online while observing users' demands and their context information. The placement algorithm does not depend on prior knowledge of content popularity, but it models users' context-dependent demands of files following Zipf distribution. The context information used in modeling the content popularity is the maximum number of users served by SBS, the fraction of female users, and the fraction of underage users. The total number of files used in the Zipf distribution formula is divided according to the context information of users.

In [127], the authors proposed a popularity prediction model for video files. The popularity of video content is estimated from both published (statistical information) and unpublished video (newly uploaded videos). The process consists of the following stages, (1) Feature extraction from

Table 2.7: Content popularity modelling in caching techniques.

References	Content Popularity Method
[114], [115], [118], [119], [2], [3], [122] [126], [128]- [132], and [109]	Zipf distribution
[135]	Number of views vs rank of videos in terms of views
[121]	Context-dependent Zipf-like distribution
[127]	Feature extraction and popularity prediction for unpublished videos
[125], [120], and [134]	Popularity estimation based on learning methods

unpublished videos based on deep neural network technique, (2) Clustering the features resulting from stage 1 based on collaborative filtering technique, and (3) Fit a regression model to predict the popularity of unpublished videos while using the statistical information of the published videos as a training set to the regression model.

The approach in [125] used Zipf distribution as the training set to design a learning-based approach. Their model estimates content popularity using regularized singular value decomposition (RSVD)-based collaborative filtering (CF), and they improve estimation accuracy using the transfer learning (TL) technique.

2.3.5.2 User Mobility

Most of the work discussed in previous sections assumes that users remain stationary while requesting and obtaining files. Unfortunately, research with this assumption does not include mobility as an effective parameter while taking cache placement decisions.

A user may be served by any SBS located in the user communication range. Thus caching technique that does not rely on user mobility may duplicate copies of the same contents at multiple nearby SBSs to serve a user request. This results in a waste of cache storage capacity and limits the varieties of cache contents

Considering mobility on the caching design in future wireless networks, caching can be classified into three categories based on cache location:

1. **Cache in SBS**:

   In [136], [137], [138], and [139], file caching in SBS while user is in mobility is considered.

2. **Cache in mobile UT**:

   In [140], [141], [142], [143], [144], and [145], user mobility-aware caching designs are proposed by utilizing D2D communication links.

3. **Cache in SBS and mobile UT**:

   In [138], [142], [139], and [146], the researchers proposed user mobility-aware cache placement in SBS and mobile UT.

Caching efficiency can be improved by exploiting user mobility-aware cache placement in SBS, and UT [142]. However, few cache placement techniques have taken the impact of user mobility [109].

Most existing approaches that estimate cache contents proactively and store the contents in SBSs face redundancy problems. This has happened in some caching strategies that have neighboring SBSs storing the same popular contents. Redundancy results in wastage of cache resources and minimizes the cache storage capacity that is available for users. The interactions between network edges should be taken into account while optimizing the caching strategy [147].

2.3.5.3 Power Constraint

Energy consumption becomes a more challenging problem in the design of wireless communications due to the increase in energy price, the number of broadband wireless network users, and growing demand of the contents in the future networks [148]. Delivering contents from SBS to UTs and from UT to another UT will consume power and drain energy at both the network and UT. Cache system should be designed to find an optimal transmit power and sustain the continuous growth of power consumption [149]. Using caching could reduce the user requests redundancy and thus decrease energy consumption.

Table 2.8: Power consumption-aware caching systems.

References	Contribution
[150]	Proposed content caching for smart grid enabled wireless multimedia transmission system with optimal power allocation to users.
[109]	Proposed an optimal transmit power of SBSs and UT in order to reduce the delivery energy cost.
[151]	Developed a framework to minimize the total network power consumption by a joint design of adaptive BS selection, backhaul content assignment and multicast beamforming.
[152]	Proposed optimal allocation cooperative caching scheme for Industrial Internet of Things (IIoT) in 5G heterogeneous networks with minimum energy consumption.
[153]	Formulate the optimal caching placement at the wireless edge that maximize the energy efficiency of heterogeneous wireless networks.
[154]	Design a green content caching and mobile userbase station (MU-BS) association mechanism in the SCNs.
[155]	Propose two energy-efficient caching in heterogeneous networks: scalable video coding (SVC)-based fractional caching and SVC-based random caching.

The challenging problem is that by adapting the caching system to reduce power transmission, sometimes the cached contents are never requested by users, which causes additional waste of energy. Therefore, formulating a cache system requires involving this trade-off. Table 2.8 illustrates power consumption-aware caching algorithms proposed for wireless networks.

2.3.5.4 Quality of Service (QoS)

The Quality of Service (QoS) is a network performance characteristic that is experienced by the end-user. Two metrics can refer to the quality of service, and they are latency and throughput. These constraints are two important metrics that need to be considered while formulating the problem of caching at the edge of MEN, as illustrated earlier on caching in future wireless networks.

1. **Latency**: In caching systems, latency refers to the average content delivery latency experienced by the end-use [107]. According to cache types, latency can be classified into three types:

    (a) Average latency of delivering the requested content from another nearby UT cache through D2D communication.

    (b) Average latency of delivering the requested content from nearby BS cache.

    (c) Average latency of delivering the requested content from nearby MBS cache.

    The latency is also referred to as delay, download time, and content delivery deadline.

    New services and applications will appear in future wireless networks, such as augmented reality (AR) and virtual reality (VR), with tight latency requirements than typical video streaming. This adds challenge to the design of the caching system to optimize the source of content to deliver it to the end-user within the delivery deadline. Caching at the edge of the network promises to reduce the latency required for requested data access and delivery.

2. **Throughput**: In caching systems, the throughput refers to the data units that can be delivered through the network per unit time interval [107]. In MEC, this metric is used as a joint indicator of network transmission capabilities. Authors in [163] discuss throughput capability in decentralized coded and uncoded caching in a multihop D2D communication for next-generation cellular networks. In addition, they illustrate the effect of using the UT cache placement strategy on the increase of throughput capabilities.

Table 2.9: Latency computation in caching schemes.

References	Contribution
[156]	Proposed latency-centric placement and delivery strategies for cloud and cache-aided wireless networks.
[143]	Propose cooperative vehicle-aided content edge caching scheme to minimize the content delivery latency.
[157]	Proposed hybrid content caching algorithms for joint content caching control in BSs and cloud units (CUs) subject to finite service latency.
[158]	Proposed a joint caching and association strategy to minimize the average requested content download delay.
[159]	Proposed an optimal cooperative content caching and delivery policy aiming to minimize the average downloading latency.
[160]	Proposed two content caching policies: caching popular files and greedy caching in BS and D2D with the aim to minimize transmission delay.
[161]	Proposed probabilistic caching placement-aided throughput in stochastic wireless D2D caching networks to measure the density of successfully served requests by local device caches.
[162]	Proposed deterministic caching algorithm and enable D2D connections based on reinforcement learning to minimizing the download latency.

Table 2.10: Throughput computation in caching schemes.

Reference	Contribution
[160]	Throughput scaling depends on video content popularity when the number of files grows asymptotically large.
[164]	Proposed femtocaching and D2D collaboration to improve video throughput.
[160]	Proposed two content caching policies: caching popular files and greedy caching in BS and D2D and investigate the behavior of the average throughput per request.
[165]	Proposed optimal file placement for deterministic and random caching with the aim to increase throughput for high user density wireless video network.
[162]	Proposed deterministic caching algorithm based on reinforcement learning to maximize system throughput.

2.4 Machine Learning and Intelligent Decision-Making Techniques Based Caching Systems

The next generations of wireless networks need to be designed to satisfy the highly dynamic and massive growth in capacity demand, the number of users, and the higher traffic density. Caching in mobile edge networks (MENs) targets the increase of network throughput and improvement of the QoS that enables energy and latency efficient performance of future applications and services. New challenges to consider in the design of caching in MENs are: estimating content popularity, users mobility, providing near-zero latency, and power-efficient communication through a higher level of cache system management and automation. To meet all these challenges, a new approach in the design of a cache system is needed. The merging of machine learning (ML) and artificial intelligence (AI) techniques with the applications of big data analytics can monitor the considerable amount of data generated within the MENs. This can provide a potential tool to model networks, and users' behavior at a large scale [166]. Modern ML and AI systems depend on observed data to recognize patterns that can help learn and make decisions on unseen data sets. Therefore, the development in data collection methods and computation capabilities have potential effects on building ML systems [167].

In this section, various types of machine learning and intelligent decision-making techniques are discussed. These techniques can be used to design an optimal caching system at the edge of the network. This is done by interacting with the environment and learning the strategies of optimal cache placement and cache delivery to minimize energy consumption and latency of delivering the requested content to the users.

2.4.1 Intelligent Learning Techniques (ILTs)

Machine learning arises as a subfield approach to Artificial Intelligence (AI) that makes machines able to access data and extract knowledge and learn from the data. Adding more data to the training set, ML algorithms can model the data and automate for learning new data patterns and updating its algorithms, called incremental learning.

ML techniques can be programmed to optimize some performance criteria by using training data sets from experience. ML techniques apply to problems that can not be solved algorithmically or expressed directly but can be identified through examples that reflect the appropriate solution. ML algorithms find patterns in a given data set (data mining), and then the program makes a decision based on what it learns from previous events [168]. AI, ML, and deep learning (DL) techniques are overlapped terms. As they go from rule-based AI systems to ML and then to DL approaches, more complex features and input-output relationships can be modeled and learned. AI solves problems that can be described by a list of formal and mathematical rules. ML algorithm creates a model based on learning the features extracted from input data and the output labels. Thus, it enables models to understand the features in the data and extract intelligence that improves system performance. DL is a subset of ML that can be used to create more complex models. DL models consist of multiple processing layers that can learn representations of data within various levels, and abstractions [169].

The authors in [170] give a higher level of description to the relationships between AI, ML, and DL, showing the flowchart of the main components as illustrated in Figure 2.5.

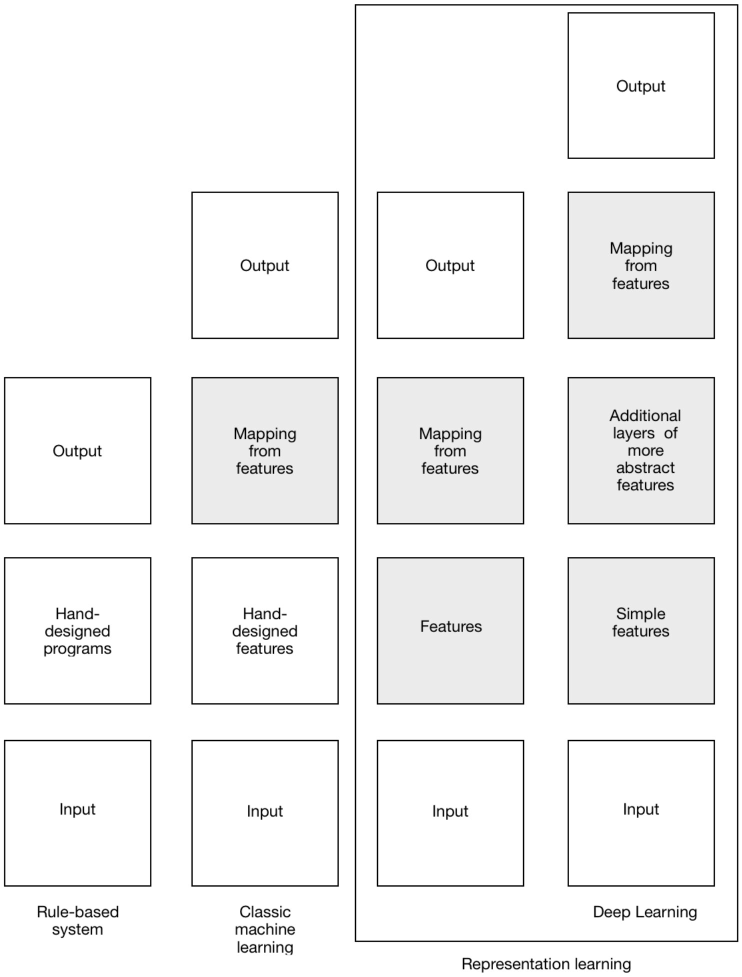


Figure 2.5: Flowchart of flow of operations in AI, ML, and DL

2.4.2 Machine Learning and Intelligent Decision Making Techniques for Optimal Caching Design

In next-generation wireless networks, ultra-dense heterogeneous networks, which are highly dynamic and complex, will add many challenges for network design and management. In addition, the wireless network will face massive data consumption from users and machines, which adds more complexity and challenges, as explained in section 2.2. The design of MENs that includes distribution of computational resources and storage devices in the form of local caches enables the utilization of decision theory, complex ML, and AI approaches to providing possible solutions for the growing challenges. Developing an optimal caching system with frequent changes of input parameters (users' mobility, file requests probability, and contents popularity) to maximize network throughput, minimize power consumption, and minimize content download time, is a highly computational complexity problem. Machine learning and intelligent decision-making techniques-based approach allow combining reasoning, learning, prediction, and decision-making algorithms to efficiently find solutions for optimal cache design.

In the literature, there is a number of research work in cache developments for future wireless networks that applied learning and/or decision approaches in a specific domain in their design. Table 2.11 illustrates a summary of these research and the solution they provided. In addition, a brief review of some of these techniques is given in the following.

2.4.2.1 Decision Theory

When the problem requires selecting one action from several possibilities, we will need to formulate a decision-making problem. In statistical theory, the branch that deals with such issues is called statistical decision theory or hypothesis testing [171].

In probability theory, there are a variety of information fusion operators. Mainly they can be classified into three groups [172] as follows:

1. **Conjunctive operators:** Can be used for merging agreeing sources, and they search for values when all the sources agree.

2. **Disjunctive operators:** Can be used for merging conflicting sources.

3. **Trade-off operators:** Can be used for partially in conflict sources.

2.4.2.2 Machine Learning

ML techniques model the functional relationship between input datasets and output actions to optimize some parameters. The resulted model can estimate an output as close as possible to the actual value. ML techniques can be categorized into two main groups: supervised and unsupervised learning depending on whether the data is labeled or not. Supervised learning aims to model input and output datasets (labeled data) while unsupervised learning aims to model the hidden structure from unlabelled data sets.

ML can explore and extract knowledge from users and network characteristics in caching systems to build an intelligent decision-making system to make decisions for cache placement, cache access, and cache delivery options.

Some ML techniques have been applied in a caching system, such as Reinforcement Learning (RL) and Deep Learning.

1. **Reinforcement Learning:** In reinforcement learning, the machine interacts with its dynamic environment through trial and error interactions. As a result of the interactions, the agent learns actions by receiving input of the current state of the environment and chooses the following action based on possible actions. The agent action affects (changes) the state of the environment. The machine receives a value of the transition state, which can be rewards or penalties. The goal is to learn a trajectory of actions that maximize the rewards (or minimize the penalties) over its lifetime. Reinforcement learning learns the optimal policy that models environment states and actions that will maximize (or minimize) its objectives [173]. Figure 2.6 shows an illustration of the agent and the environment interaction [174].

   In [175], SBSs prefetch popular contents during off-peak traffic hours and send the contents to the edge of the network during peak periods. The cache control unit in the SBS is designed to learn, track, and adapt to the work dynamics. The authors proposed an optimal online caching policy by developing a Q-learning algorithm. The Q-learning scheme is introduced with a linear function approximation to offer fast convergence, reduce complexity, and obtain scalability over large networks

Table 2.11: ILTs for caching system

Learning Technique	**Paper**	**Technical Solution**
Regularized SVD K-means clustering	[4]	Proactively cache files based on file popularity, correlation among users, and exploit influential users.
RSVD based CF and TL	[125]	Estimate the content popularity and improve the estimation accuracy.
Deep learning	[176]	Predict content popularity.
Extreme learning machine (ELM)	[134]	Estimate the popularity of cache content based on the features of the content
Deep belief network (DBN)	[177]	Extract semantic information of user playback pattern.
CF	[111]	Predict the content popularity distribution.
ML on Hadoop framework	[178]	Estimate content popularity.
Clustering technique	[179]	Track the evolution of content popularity over time.
Clustering technique	[180]	Content popularity based users clustering.
Reinforcement learning	[181]	Enabling access points to learn the optimal fetching-caching decisions.
Q-learning	[175]	Learn, track, and adopt optimal policy when the underlying transition probabilities are unknown.
Rank-Directed Sparse Bayesian Learning	[182]	Estimate content popularity.
TL	[183]	Estimate content popularity.
Deep neural network (DNN)	[184]	Propose caching placement and content delivery.
Bayesian and reinforcement learning	[162]	Predict individual content request probability
Genetic algorithm	[185]	Proposed near optimal hierarchical collaborative caching.
FSS	[186]	Proposed fuzzy soft-set decision cache placement.
Q-Learning	[175]	Proposed an optimal online caching policy.

Continued on next page

Table 2.11 – continued from previous page

Deep Learning	[184]	Proposed cache placement, user association, and content delivery.

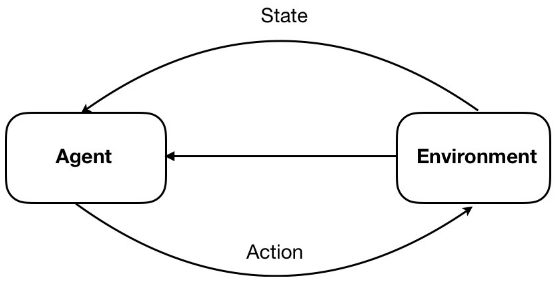


Figure 2.6: The agent-environment interaction system.

2. **Deep Learning:** Deep learning represents the form of learning that creates complex features by using multiple transformation steps. Much larger quantities of data are used during learning steps. Deep learning techniques show the ability to explore information included in massive data sets more effectively than traditional ML techniques. Deep learning implies learning complex artificial neural networks (ANN) that extract patterns progressively in the datasets. In traditional ANN, the three-layer perceptron (input, hidden, and output layers) learns by training the hidden and output layers to adapt to the interesting task. In deep learning, more hidden layers are added to the network to subject features to the sequence of transformations. Each layer's transformation represents an inference. Thus, modeling complex inferences can be made easier using a series of computational steps. The depth of the ANN represents the complexity of the learning algorithm. Some ANN learning algorithms include feedback loops. There are several ANN deep learning techniques such as deep multilayer perceptrons, deep convolutional neural networks (CNNs), and recurrent neural networks [187].

   Authors in [184] proposed a deep neural network (DNN) to train an optimization problem for cache placement, user association, and content delivery in advance and before applying these optimization algorithms in real-time caching.

2.5 Summary

This chapter presents mobile edge computing and caching and the literature survey of solutions for mobile edge computing and caching challenges in terms of energy and latency. This is followed by discussing caching schemes, caching performance metrics, and comparing different caching techniques in mobile edge networks. Next, a summary of the challenges that face the design of the caching system in MENs is discussed. Finally, we present various types of machine learning and intelligent decision-making techniques that can be used to design a caching system at the edge of the network. The following chapter will present our mobile edge network system models and formulate the cache placement problem as a multi-objective optimization problem.

Chapter 3

Mobile Edge Network System Model

3.1 Introduction

This chapter introduces our cache system model of the mobile edge networks (MENs) to provide cache content placement, cache content access, and cache content delivery. It considers different storage capacities, users' mobility, content popularity distribution, latency, and power consumption. The system model includes the following models: network model, cache content model, transmission model, mobility model, and power consumption model. Then, the problem formulations related to finding the optimal solution for cache content placement with minimum latency and cache content delivery with minimum power consumption are presented and discussed.

3.2 System Model

In the following sections, we define the main models, identify essential components, and address the role of each element in the whole system.

3.2.1 Network Model

As illustrated in Figure 3.1, we consider a mobile edge network with multiple small base stations (SBSs) and user terminals (UTs). Each macrocell consists of one MBS connected to a gateway of the core network via a high-speed interface, N SBSs, which are connected to the MBS through backhaul links, and M UTs connected to neighboring devices through D2D communication and to SBSs. The sets of SBSs and UTs are denoted by $s=\{s_1, s_2, \cdots, s_N\}$ and $u=\{u_1, u_2, \cdots, u_M\}$, respectively. There are cache storage in each MBS, SBS, and UT with different storage capacities. Within one macro cell, cache storage capacities can be defined by two sets $c_s=\{c_{s_1}, c_{s_2}, \cdots, c_{s_N}\}$ and $c_u=\{c_{u_1}, c_{u_2}, \cdots, c_{u_N}\}$ for SBSs and UTs caches, respectively. Assume that MBS has enough cache capacity to store F files defined by the set $f=\{f_1, f_2, \cdots, f_F\}$.

As discussed in Chapter 2, MEN provides the communication and caching abilities that enable SBSs and UTs to store and transmit contents at different network levels. This framework is called multilayer caching-networking framework [107]. In our work, we assume four possible cache content delivery approaches as shown in Figure 3.1 and described below:

(a) **Local cache content delivery:** Delivering the requested content from UT local cache.

(b) **D2D cache content delivery:** Delivering the requested content from neighboring UT cache when the other UT device is within the D2D communication range.

(c) **SBS cache content delivery:** Delivering the requested content from SBS cache that is associated with the UT.

(d) **MBS cache content delivery:** Delivering the requested content from MBS in the macro-cell. This happens when the requested content is missed, resulting in ”cache miss” in local cache, D2D cache, and SBS cache, respectively.

3.2.2 Cache Content Model

The content library consists of F files and stored at the MBS cache. Each file f_z has the same size. The size of file f_z is denoted as f_{l_z}. It is assumed that each mobile UT receives one file, and each SBS receives a certain amount of files depending on the cache's file size and storage capacity. The files are requested from the main library based on their popularity distribution. The popularity of the F files are denoted by the set p, where $p=\{p_1, p_2, \cdots, p_F\}$. The set p can be characterized by Zipf popularity distribution. If the files are arranged from the highest popular file to the lowest popular file, the popularity of the $i-$ th ranked content can be shown by Eq. (3.1) [188]:

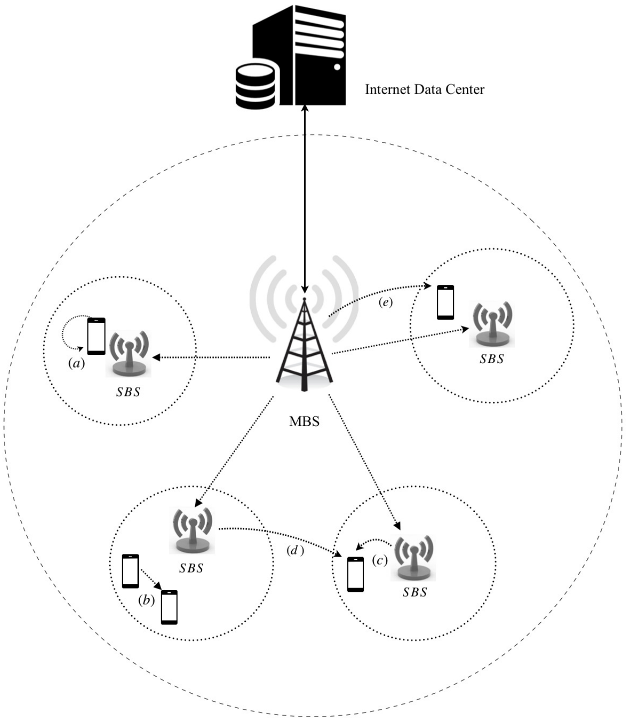


Figure 3.1: An illustration of the network model showing the five cache content delivery approaches: (a) local cache, (b) D2D cache, (c) SBS cache, and (d) MBS cache.

$$p_i = \frac{\frac{1}{f_i^\gamma}}{\sum_{i=1}^{F} \frac{1}{i^\gamma}}. \tag{3.1}$$

The distribution for file f_i is characterized by the exponent factor γ also called the skewness of the popularity. When $\gamma = 0$, the popularity is uniform over contents. As γ grows, the popularity becomes more skewed. Zipf popularity distribution has been widely used in the research literature, and we also use it in our model.

3.2.3 Mobility Model

Assume that the connectivity of users in the D2D communication and the small cell network (communication between mobile UT and SBS) is a peer-to-peer connectivity model [109]. Modeling user mobility depends on spatial and temporal properties. The spatial properties provide physical location information of user mobility patterns, while the temporal properties provide time-related information [142]. User mobility can be modeled by assuming a pairwise contact process that follows an independent Poisson process. The work in [189] implies that the occurrence time of pairwise contact events can be predicted on a large time scale. The Poisson process is used to count contacts between UTs and between UTs and SBSs occurring at a specific rate.

To establish successful communication between mobile UT u_i and SBS s_j, u_i must be within the communication radius of SBS s_j. For independent Poisson process, the pairwise contact duration $T_{i,j}^{SBS}$ between mobile UT u_i and SBS s_j follows the exponential distribution with parameter $\lambda_{i,j}^{SBS}$. Here, $\lambda_{i,j}^{SBS}$ represents the contact rate between mobile UT u_i and SBS s_j. The contact duration $T_{i,j}^{SBS}$ when mobile UT u_i is within the communication range of SBS s_j can be defined as follows [109]:

$$T_{i,j}^{SBS} = \{(t - t_0) : ||l_j^t - l_i^t|| < d^{SBS}, t > t_0\}, \tag{3.2}$$

where t_0 represents the most recent time when mobile UT u_i enters the communication range d^{SBS} of SBS s_j. The locations of SBS s_j and mobile UT u_i at time t are represented by l_j^t and l_i^t, respectively.

Similarly, to establish successful communication between mobile UT u_i and mobile UT u_k, the shortest distance between the two devices must be within the communication range d^{D2D}. The contact rate between mobile UT u_i and mobile UT u_k is represented by $\lambda_{i,k}^{UT}$. The contact duration $T_{i,k}^{UT}$ when mobile UT u_i and mobile UT u_k are within the communication range of d^{D2D} can be defined as follows [109]:

$$T_{i,k}^{UT} = \{(t - t_0) : ||l_i^t - l_k^t|| < d^{D2D}, t > t_0\}, \tag{3.3}$$

where t_0 represents the most recent time when mobile UT u_i enters the communication range d^{D2D} of mobile UT u_k, and l_i^t and l_k^t represent the locations of UT u_i and UT u_k at time t, respectively.

3.2.4 Transmission Model

Let the matrices $(h^{UT})_{M \times F}$ and $(h^{SBS})_{N \times F}$ represent the cache hit references for M UTs and N SBSs, respectively, such that:

$$h_{i,z}^{UT} = \begin{cases} 1, & \text{when the file } f_z \text{ is cached in UT } u_i \text{ (hit)} \\ 0, & \text{otherwise (miss),} \end{cases} \tag{3.4}$$

$$h_{j,z}^{SBS} = \begin{cases} 1, & \text{when the file } f_z \text{ is cached in SBS } s_j \text{ (hit)} \\ 0, & \text{otherwise (miss),} \end{cases} \tag{3.5}$$

where $z = 1, \cdots, F$, $i = 1, \cdots, M$, and $j = 1, \cdots, N$.

Let $T_{d_{k,z}}^{UT}$ denote the time required for UT u_i to download the file f_z from UT u_k. Assume that the file length is a random variable that follows a log-normal distribution [190] [191] and is denoted by f_{l_z} for file z. Then, $T_{d_{k,z}}^{UT}$ is given by [192]:

$$T_{d_{k,z}}^{UT} = \frac{f_{l_z}}{R_{k,z}^{u_i}}, \tag{3.6}$$

where $k = 1, \cdots, M$ and $R_{k,z}^{u_i}$ is the data rate at which file f_z is downloaded by UT u_i from UT u_k through D2D link. Similarly, $T_{d_{j,z}}^{SBS}$ denotes the download time required for UT u_i to download the file f_z from SBS s_j. Thus, $T_{d_{j,z}}^{SBS}$ is given by [192]:

$$T_{d_{j,z}}^{SBS} = \frac{f_{l_z}}{R_{i,z}^{s_j}}, \tag{3.7}$$

where $R_{i,z}^{s_j}$ is the rate of file f_z of the UT u_i from SBS s_j link. When the requested file is not found in nearby UTs and SBSs, then the file will be downloaded from MBS through backhaul links.

Assume that $T_{d_z}^{B}$ is the backhaul download time from MBS for file f_z. The latency of the backhaul download depends on the used technical solution. For 5G, the latency can be 200 μsec or 65-350 μsec if the technology used is mmWave 60 GHz or 70-80 GHz, respectively [193]. The download time of the file f_z can be derived by combining (3.7) and (3.6) with the backhaul download time, as follows:

$$\begin{gathered} T_{d_z} = \sum_{v=1}^{g_z} \left\{ T_{d_{k,v}}^{UT} \times h_{i,v}^{UT} + T_{d_{j,v}}^{SBS} \times h_{j,v}^{SBS} + T_{d_v}^{B}[(h_{i,v}^{UT} = 0) \cap (h_{j,v}^{SBS} = 0)] \right\} \\ \forall z \in \{1, \cdots, F\}, \forall k \in \{1, \cdots, M\}, \forall j \in \{1, \cdots, N\}, \forall i \in \{1, \cdots, M\}. \end{gathered} \tag{3.8}$$

Equation (3.8) means that the download time for file f_z is computed by the summation of the download time required for file f_z. Files may be cached in UT device and/or SBS. If the file is not cached in UTs ($h_{i,v}^{UT} = 0$) and SBSs ($h_{j,v}^{SBS} = 0$) then it will be downloaded from MBS through backhaul link.

3.2.5 Energy Consumption Model

In this subsection, the power consumption model is adapted from [109]. Two different models are considered:

1. **Energy consumption for D2D caching:**

   We assume that the interference of D2D communication is not considered. When UT u_k transmits the cached contents to UT u_i, the components of power consumption of UT u_k are given as follows:

   - β_{UT} is the inverse of power amplifier efficiency factor of mobile UT u_k, ,
   - $\mathcal{P}_T^{u_k}$ is the mobile device transmission power of mobile UT u_k,
   - $\mathcal{P}_C^{u_k}$ is the circuit power consumption of mobile UT u_k,
   - $\mathcal{P}_H^{u_k}$ is the energy consumption of caching hardware devices of mobile UT u_k.

   Then, power consumption of UT u_k, $\forall k$ is given by:

$$\mathcal{P}^{u_k} = \beta_{UT} \times \mathcal{P}_T^{u_k} + \mathcal{P}_C^{u_k} + \mathcal{P}_H^{u_k}, \forall k \in \{1, \cdots, M\}.$$

   Neglecting the energy consumed for delivering the cache contents, the power consumption can be written as:

$$\mathcal{P}^{u_k} = \beta_{UT} \times \mathcal{P}_T^{u_k} + \mathcal{P}_C^{u_k}, \forall k \in \{1, \cdots, M\}. \tag{3.9}$$

   When UT u_k transmits file f_z of length f_{l_z} to UT u_i, the energy consumption can be computed as [109]:

$$E^{u_k}{}_z = \frac{f_{l_z}}{\mathcal{R}_{i,k}}.(\beta_{u_k} \times \mathcal{P}_T^{u_k} + \mathcal{P}_C^{u_k}), \forall k \in \{1, \cdots, M\}, \tag{3.10}$$

   where the data rate $\mathcal{R}_{i,k}$ of D2D communication between UT u_i and UT u_k can be calculated as follows:

$$\mathcal{R}_{i,k} = W_{i,k}^{UT} \log_2 \left(1 + \frac{\mathcal{P}_T^{u_k} . {d_{i,k}}^{-\alpha_{UT}}}{\sigma_{UT}^2} \right), \forall i \in \{1, \cdots, M\}, \forall k \in \{1, \cdots, M\}, \tag{3.11}$$

   and

   - $W_{i,k}^{UT}$ is the channel bandwidth from UT u_k to UT u_i,

- $d_{i,k}$ is the distance between u_i and u_k,
- σ^2_{UT} is the average noise power for D2D communication,
- α_{UT} is the path loss factor.

2. **Energy consumption for SBS caching**

Similarly, to compute the energy consumption to transfer file f_z from SBS s_j to UT u_i, we assume there is no interference between SBSs. The downlink speed $\mathcal{R}_{i,j}$ is given below:

$$\mathcal{R}_{i,j} = W^{SBS}_{i,j} \log_2 \left(1 + \frac{\mathcal{P}^{s_j}_T . {d_{i,j}}^{-\alpha_{SBS}}}{\sigma^2_{SBS}}\right), \forall i \in \{1, \cdots, M\}, \forall j \in \{1, \cdots, N\}, \tag{3.12}$$

where

- $W^{SBS}_{i,j}$ is the downlink transmission bandwidth from SBS s_j to UT u_i,
- $\mathcal{P}^{s_j}_T$ is the SBS transmission power,
- $d_{i,j}$ is the distance between u_i and s_j,
- σ^2_{SBS} is the average noise power in communication with SBS,
- α_{SBS} is the path loss factor.

Then, the components of power consumption of SBS s_j are given as follows:

- β_{SBS} is the inverse of power amplifier efficiency factor,
- $\mathcal{P}^{s_j}_C$ is the offset of site power.

When SBS s_j transmits file f_z of length f_{l_z} to UT u_i, the energy consumption can be computed as [109]:

$$E^{s_j}{}_z = \frac{f_{l_z}}{\mathcal{R}_{i,j}} . (\beta_{SBS} \times \mathcal{P}^{s_j}_T + \mathcal{P}^{s_j}_C). \tag{3.13}$$

Table 3.1 defines the various system parameters used in our system model.

3.3 Problem Statement

This dissertation aims to find solutions to achieve latency-efficient and energy-efficient caching strategies in mobile edge networks (MENs). The solution is designed based on the relationship between the MEN model and the objective of obtaining minimum latency that satisfies wireless network applications/services (discussed in section 2.2.1).

Table 3.1: System model parameters and their definitions.

System Symbols	**Definition**	**Fixed/ Variable**
N	Number of SBSs	F/V
M	Number of user terminals	F/V
s	Set of SBSs	F/V
u	Set of user terminals	F/V
c_s	Set of cache storages in SBSs	F/V
c_u	Set of cache storages in user terminals	F/V
F	Number of files in MBS storage library	F/V
f	Set of files in the library	F/V
p	Set of files popularity	V
γ	Skewness of the Zipf distribution popularity	F/V
$T_{i,j}^{SBS}$	Contact duration of mobile UT u_i and SBS s_j.	F/V
$\lambda_{i,j}$	Contact rate of mobile UT u_i and SBS s_j.	F/V
d^{SBS}	Communication range between SBS and UT	F
d^{D2D}	Communication range of two UT devices	F
$T_{i,k}^{UT}$	Contact duration of mobile UT u_i and UT u_k.	F
$h_{i,z}^{UT}$	Cache hit references of UT u_i and file f_z.	V
$h_{i,z}^{SBS}$	Cache hit references of SBS s_j and file f_z.	V
f_{l_z}	Length of file z	V
$T_{d_{j,z}}^{SBS}$	Download time required for UT u_i to download file f_z from SBS s_j.	V
$T_{d_{j,z}}^{UT}$	Download time required for UT u_i to download file f_z from UT u_k.	V
$T_{d_z}^{B}$	Backhaul download time.	F
$R_{i,z}^{s_j}$	Date rate of file f_z of the UT u_i to download from SBS.	V
$R_{k,z}^{u_i}$	Data rate of file f_z of the UT u_i to download from UT u_k.	V

β_{UT}	Inverse of power amplifier efficiency factor.	F
$\mathcal{P}_T^{u_k}$	Mobile device transmission power.	F
$\mathcal{P}_C^{u_k}$	Circuit power consumption of mobile UT u_k.	F
$\mathcal{P}_H^{u_k}$	Energy consumption of of caching hardware devices.	F
$\mathcal{P}^{u_k}$	Power consumption of UT u_k in D2D communication	F
E^{u_k}	Energy consumption in D2D communication of UT u_k.	V
$W_{i,k}^{UT}$	Channel bandwidth from UT u_k to UT u_i	F
$d_{i,k}$	Distance between UT u_k and UT u_i	V
$d_{i,j}$	Distance between UT u_i and SBS s_j	V
σ_{UT}^2	Average noise power for D2D communication.	V
α_{UT}	Path loss factor for D2D communication.	V
α_{SBS}	Path loss factor for SBS to UT communication.	V
$W_{i,j}^{SBS}$	Downlink transmission bandwidth from SBS s_j to UT u_i	F
$\mathcal{P}_T^{s_j}$	SBS transmission power	F
E^{s_j}	Energy consumption in SBS caching communication	V
σ_{SBS}^2	Average noise power	V
L^{UT}	Set of locations of all UTs in one macrocell.	V
L^{SBS}	Set of locations of all SBSs in one macrocell.	F

The proposed cache system design consists of three separate phases, obtained by adding a phase to the two phases of the cache system used in popular approaches in the literature (refer to section 2.3.1). Figure 3.2 shows the main elements of the proposed cache system objectives and their related system models and cache phases.

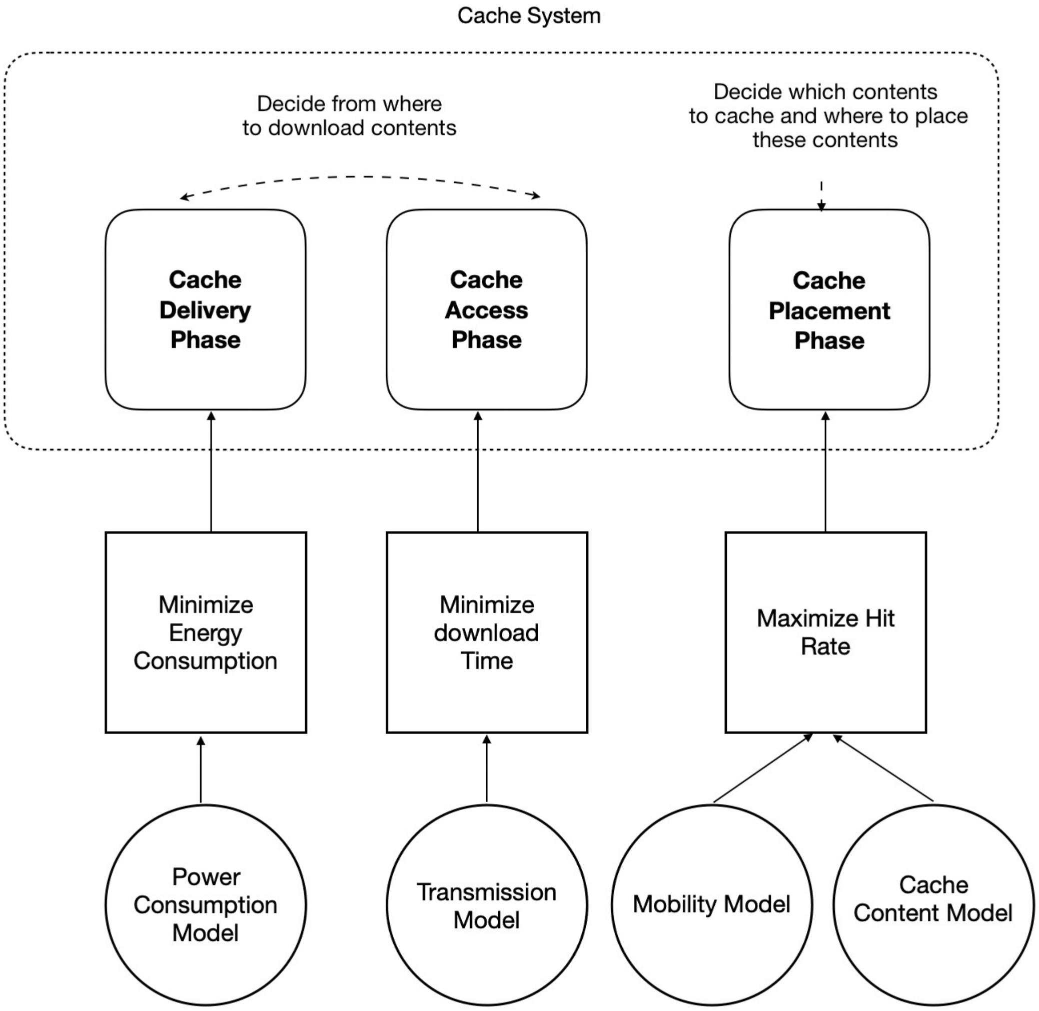


Figure 3.2: Main elements of the proposed CSOs for mobile edge networks.

The cache phases are given as follows:

1. **Cache Placement Phase**: In this phase, each cache is filled with the appropriate contents. We need to formulate an optimization problem that decides which contents to cache and where to place these contents to satisfy the objective function, which requires maximizing the hit rate. Maximizing the hit rate leads to maximizing QoS parameters that are, in our case, the latency of downloading the files and throughput of the network, respectively.

   To decide which contents to store in UT and SBS caches, both the mobility and cache content models are included in the design of cache placement strategy within the download latency constraints.

The mobility model helps decide cache content placement, which depends on users' mobility, communication radius, and contact duration. The cache content model helps to cache the most famous content based on Zipf's popularity distribution.

2. **Cache Access Phase**: Users send their requests to download content. The files are stored in UTs and SBSs caches at the edge of the network. Therefore, downloading the requested contents requires deciding from where to bring these contents: either from nearby UT or SBS caches.

   Given the transmission model, the decision can be formulated to achieve the objective function that requires minimizing the download time that results in an energy-efficient solution.

3. **Cache Delivery Phase**: In reality, the download time affects the energy efficiency of the cache policy and the power consumption during the downloading of the requested content. Therefore, the cache delivery phase aims to obtain the optimal transmission power within contact duration.

   Therefore, we need to build a multi-objective optimization function for cache access and delivery phases, deciding where to download the requested contents. Each objective has a weight factor depending on the overall efficiency and flexibility in operating the caching system.

3.3.1 Latency Efficient Caching

Several sources contribute to the total latency of a wireless network. Using MENs that enable caching is one of the promising solutions to establish latency-efficient wireless networks (as discussed in section 2.2.3). Caching at the edge of the network (UTs, SBSs, and MBS caches) minimizes the total latency by providing the requested files closer to the users. This requires minimizing the probability of downloading the files from the core network and maximizing the probability of downloading the files from edge caches.

Consider a mobile edge network (MEN) described in section 3.2 with N SBSs, M UTs, and F files in the main library. At time t_0, the set of locations of the UTs (L^{UT}) are variable values and given as follows:

$L^{UT} = \{l_1^{UT}, l_2^{UT}, \cdots, l_M^{UT}\}$

and the set of locations L^{SBS} of the SBSs are fixed values and given as follows:

$L^{SBS} = \{l_1^{SBS}, l_2^{SBS} \cdots, l_N^{SBS}\}$

The contact durations of SBSs and UTs are given in (3.2) and (3.3), respectively. We can compute the matrix $(T^{UT})_{(M\times M)}$ that represents the contact durations between UT u_i and UT u_k in device to device (D2D) communications for $i = 1, 2, \cdots, M$ and $k = 1, 2, \cdots, M$. Then,

$$T^{UT} = \begin{bmatrix} 0 & t_{1,2}^{UT} & \cdots & t_{1,M}^{UT} \\ t_{2,1}^{UT} & 0 & \cdots & t_{2,M}^{UT} \\ \vdots & \vdots & \cdots & \vdots \\ t_{M,1}^{UT} & t_{M,2}^{UT} & \cdots & 0 \end{bmatrix}. \tag{3.14}$$

Each individual element of the matrix represented as $t_{i,k}^{UT}$, is zero for $i = k$ and non-zero otherwise, $t_{i,k}^{UT} = 0$ means that the contact duration for local caches is zero ($t_{1,1}^{UT} = 0, t_{2,2}^{UT} = 0$, etc.).

The matrix $(T^{SBS})_{(M\times N)}$ represents the contact durations between UT u_i and SBS s_j in UT to SBS communications for $i = 1, 2, \cdots, M$ and $j = 1, 2, \cdots, N$. Then:

$$T^{SBS} = \begin{bmatrix} t_{1,1}^{SBS} & t_{1,2}^{SBS} & \cdots & t_{1,N}^{SBS} \\ t_{2,1}^{SBS} & t_{2,2}^{SBS} & \cdots & t_{2,N}^{SBS} \\ \vdots & \vdots & \cdots & \vdots \\ t_{M,1}^{SBS} & t_{M,2}^{SBS} & \cdots & t_{M,N}^{SBS} \end{bmatrix}, \tag{3.15}$$

where each individual element of the matrix is represented as $t_{i,j}^{SBS}$. Combining the two matrices (3.14) and (3.15) results in $(\mathcal{T}^{UT,SBS})_{M\times(M+N)}$ such that:

$$\mathcal{T}^{UT,SBS} = \begin{bmatrix} 0 & t^{UT}_{1,2} & \cdots & t^{UT}_{1,M} & t^{SBS}_{1,1} & t^{SBS}_{1,2} & \cdots & t^{SBS}_{1,N} \\ t^{UT}_{2,1} & 0 & \cdots & t^{UT}_{2,M} & t^{SBS}_{2,1} & t^{SBS}_{2,2} & \cdots & t^{SBS}_{2,N} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ t^{UT}_{M,1} & t^{UT}_{M,2} & \cdots & 0 & t^{SBS}_{M,1} & t^{SBS}_{M,2} & \cdots & t^{SBS}_{M,N} \end{bmatrix}. \tag{3.16}$$

The matrix given by (3.16) includes the contact durations between all UTs and other UTs and SBSs within one cell. Similarly, a matrix $\mathcal{D}^{\mathcal{UT},\mathcal{SBS}}$ can be defined for the communication ranges at time t, when both UT u_i and UT u_k are within the D2D communication range d^{D2D}, and UT u_i and SBS s_j are within the communication range d^{SBS}, respectively. Then, $(\mathcal{D}^{\mathcal{UT},\mathcal{SBS}})_{M\times(M+N)}$ can be written as follows:

$$\mathcal{D}^{UT,SBS} = \begin{bmatrix} 0 & d^{UT}_{1,2} & \cdots & d^{UT}_{1,M} & d^{SBS}_{1,1} & d^{SBS}_{1,2} & \cdots & d^{SBS}_{1,N} \\ d^{UT}_{2,1} & 0 & \cdots & d^{UT}_{2,M} & d^{SBS}_{2,1} & d^{SBS}_{2,2} & \cdots & d^{SBS}_{2,N} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ d^{UT}_{M,1} & d^{UT}_{M,2} & \cdots & 0 & d^{SBS}_{M,1} & d^{SBS}_{M,2} & \cdots & d^{SBS}_{M,N} \end{bmatrix}, \tag{3.17}$$

where $d^{UT}_{i,k}$ and $d^{SBS}_{i,j}$ are the distances between UT u_i and UT u_k and the distances between UT u_i and SBS s_j, respectively. A value of $d^{UT}_{i,k} = 0$ means that the communication range for local caches is zero ($d^{UT}_{1,1} = 0, d^{UT}_{2,2} = 0, etc.$). They are computed as follows:

$$d^{UT}_{i,k} = \begin{cases} ||l_i - l_k||, & \text{if} \quad ||l_i - l_k|| < d^{D2D} \\ 0, & \text{if} \quad i = k, \end{cases} \tag{3.18}$$

and

$$d^{SBS}_{i,j} = ||l_i - l_j||\text{if} \quad ||l_i - l_j|| < d^{SBS}. \tag{3.19}$$

The contents are requested based on their popularity by following the Zipf popularity distribution given in (3.1). Each user requests content f_z with the probability of p_z, where, p_1 represents the most popular file with popularity rank 1 and p_F represents the least popular file with popularity rank F, respectively. Then, the popularity vector is given as below:

$$p = [p_1, p_2, \cdots, p_F]. \tag{3.20}$$

According to the mobility model discussed in section 3.2.3, the number of contacts between UTs and other UTs and SBSs follows Poisson distribution. Then, $P_m^{UT,SBS}$ is a matrix of size $(M \times (M+N))$ that represents the probability of UT to contact other UTs or SBSs. Each individual element of the matrix is represented as $P_{m_{i,k}}^{UT}$ and $P_{m_{i,j}}^{SBS}$ for the probability of UT u_i to contact with UT u_k, and for the probability of UT u_i to contact with SBS s_j, respectively, such that:

$$P_m^{UT,SBS} = \begin{bmatrix} p_{m_{1,1}}^{UT} & p_{m_{1,2}}^{UT} & \cdots & p_{m_{1,M}}^{UT} & p_{m_{1,1}}^{SBS} & p_{m_{1,2}}^{SBS} & \cdots & p_{m_{1,N}}^{SBS} \\ p_{m_{2,1}}^{UT} & p_{m_{2,2}}^{UT} & \cdots & p_{m_{2,M}}^{UT} & p_{m_{2,1}}^{SBS} & p_{m_{2,2}}^{SBS} & \cdots & p_{m_{2,N}}^{SBS} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ p_{m_{M,1}}^{UT} & p_{m_{M,2}}^{UT} & \cdots & p_{m_{M,M}}^{UT} & p_{m_{M,1}}^{SBS} & p_{m_{M,2}}^{SBS} & \cdots & p_{m_{M,N}}^{SBS} \end{bmatrix}. \tag{3.21}$$

If we consider the mobility of users within the cell, then the most popular files need to be encoded and cached at different locations within the cell. The user may request these files while moving. If the user can download the file from nearby UTs or SBS without the need to use a backhaul link, then the hit rate (defined in section (2.3.2)) will be increased. To maximize the cache hit rate, which means maximizing the probability of downloading the files from edge caches, we need to decide where to place the files. The attributes that contribute to finding the possible solution for cache placement that increases cache hit rate are as follows:

1. File popularity probability, p,

2. Contact duration of UTs and SBSs, $\mathcal{T}^{UT,SBS}$.

3. Communication ranges, $\mathcal{D}^{UT,SBS}$.

4. User terminal UT contact probability $P_m^{UT,SBS}$.

Depending on the above attributes, we decide where to place the files in UT and SBS caches during the cache placement phase. Assume that the file size is fixed. Then, while achieving maximum cache hit rate, the total number of files cached in UT and SBS caches should be within cache storage capacities in UTs and SBSs, respectively.

Assume that matrix $(A^{UT})_{M\times F}$ represents the cache strategy of the files that are cached in UTs, where $a_{k,z} \in A^{UT}$ represents file f_z that can be served by UT u_k $(k = 1, \cdots, M)$. Let matrix $(A^{SBS})_{N\times F}$ represent the cache strategy of the files cached in SBSs, where $a_{j,z} \in A^{SBS}$ represents portion of encoded data of file f_z that can be served by SBS s_j $\forall j \in \{1, \cdots, N\}$. Then,

$$a_{k,z}^{UT} = \begin{cases} 1, & \text{if} \quad f_z \in u_k \\ 0, & \text{if} \quad f_z \notin u_k, \end{cases} \tag{3.22}$$

$$a_{j,z}^{SBS} = \begin{cases} 1, & \text{if} \quad f_z \in s_j \\ 0, & \text{if} \quad f_z \notin s_j, \end{cases} \tag{3.23}$$

Define matrix $A = [A^{UT} : A^{SBS}]_{(M+N)\times F}$ as the cache strategy matrix that needs to be solved using cache placement algorithm, such that:

$$A = \begin{bmatrix} a_{1,1}^{UT} & a_{1,2}^{UT} & \cdots & a_{1,F}^{UT} \\ a_{2,1}^{UT} & a_{2,2}^{UT} & \cdots & a_{2,F}^{UT} \\ \vdots & \vdots & & \vdots \\ a_{M,1}^{UT} & a_{M,2}^{UT} & \cdots & a_{M,F}^{UT} \\ a_{1,1}^{SBS} & a_{1,2}^{SBS} & \cdots & a_{1,F}^{SBS} \\ a_{2,1}^{SBS} & a_{2,2}^{SBS} & \cdots & a_{2,F}^{SBS} \\ \vdots & \vdots & & \vdots \\ a_{N,1}^{SBS} & a_{N,2}^{SBS} & \cdots & a_{N,F}^{SBS} \end{bmatrix}. \tag{3.24}$$

To find the contents of matrix A, either one or zero (cache or not cache), we need to formulate the individual objective functions that contribute to the solution.

Given the files popularity p, the algorithm should select the files with the higher probability of being requested by users. Then, the objective function is written as:

$$\underset{<A>}{\text{maximize}} \quad : \sum_{i=1}^{M+N} A(i,:) \times p_z^T. \tag{3.25a}$$

$$\text{subject to} \quad : \sum_{z=1}^{F} a_{i,z} \times f_{l_z} \leq C_{u_i}, \ \ \forall i \in \{1, \cdots, M\}, \tag{3.25b}$$

$$\sum_{z=1}^{F} a_{j,z} \times f_{l_z} \leq C_{s_j}, \ \ \forall j \in \{1, \cdots, N\}, \tag{3.25c}$$

For F files in the library, we have many possible solutions on which files to be placed in each UT and SBS cache. According to the objective function in (3.25a), the files with the maximum probability of popularity will be selected to be stored on UT and SBS caches. The constraints (3.25b) and (3.25c) mean that the total number of files stored in UT and SBS can not exceed the storage capacity C_{u_i} and C_{s_j} of UT u_i and SBS s_j, respectively.

At time t, the contact duration $\mathcal{T}^{UT,SBS}$ and communication range $\mathcal{D}^{\mathcal{UT},\mathcal{SBS}}$ can be computed as in (3.16) and (3.17), respectively. The users should obtain their requested contents within the deadline denoted by T_d, while they are within the communication range d^{D2D} and d^{SBS} for D2D communication and UT to SBS communication, respectively. It is defined as the time frame required to successfully transmit a certain amount of data from one peer to another peer while both peers are within the communication range. Then, the algorithm should select the place to store the contents. The probability of having successful communication from caches closer to users (minimum distance between users and caches) is maximized. The objective functions for contact duration and communication range can be formulated as follows:

$$\underset{<A>}{\text{maximize}} \quad : \sum_{i=1}^{M} \sum_{z=1}^{F} \mathcal{T}^{UT,SBS}(i,:) \times A(:,z) \tag{3.26a}$$

$$\text{subject to} \quad : \quad 0 < t_{i,k} < T_d, \ \ \forall i \in \{1, \cdots, M\}, \ \ \forall k \in \{1, \cdots, M\}, \tag{3.26b}$$

$$0 < t_{i,j} < T_d, \ \ \forall i \in \{1, \cdots, M\}, \ \ \forall j \in \{1, \cdots, N\}, \tag{3.26c}$$

and

$$\underset{< A >}{\text{minimize}} \quad : \sum_{i=1}^{M}\sum_{z=1}^{F} \mathcal{D}^{UT,SBS}(i,:) \times A(:,z) \tag{3.27a}$$

$$\text{subject to} \quad : \quad 0 < d_{i,k} < d^{UT}, \ \forall i \in \{1,\cdots,M\}, \ \forall k \in \{1,\cdots,M\}, \tag{3.27b}$$

$$0 < d_{i,j} < d^{SBS}, \ \forall i \in \{1,\cdots,M\}, \ \forall j \in \{1,\cdots,N\}. \tag{3.27c}$$

Finally, we need to maximize the probability that UT u_i contacts with another UT u_k and SBS s_j, then the objective function is written us:

$$\underset{< A >}{\text{maximize}} \quad : \sum_{i=1}^{M}\sum_{z=1}^{F} P_m^{UT,SBS}(i,:) \times A(:,z) \tag{3.28a}$$

For the latency efficient caching, we can use the weighted sum method to solve the multi objective functions given in (3.25a), (3.26a), (3.27a), and (3.28a) by selecting weights w_x^L $(x = 1,\cdots,4)$ for each objective function that aims to achieve latency efficient caching. Then, for M UTs and N SBSs we have the following:

$$f_1 = \sum_{i=1}^{M+N} A(i,:) \times p_z^T \ \forall z \in \{1,\cdots,F\}, \tag{3.29}$$

$$f_2 = \sum_{i=1}^{M}\sum_{z=1}^{F} \mathcal{T}^{UT,SBS}(i,:) \times A(:,z), \tag{3.30}$$

$$f_3 = \sum_{i=1}^{M}\sum_{z=1}^{F} \mathcal{D}^{UT,SBS}(i,:) \times A(:,z), \tag{3.31}$$

and

$$f_4 = \sum_{i=1}^{M}\sum_{z=1}^{F} P_m^{UT,SBS}(i,:).A(:,z), \tag{3.32}$$

Then, formulate one objective function for cache placement that maximizes the probability of UT u_i to download the requested files from nearby UTs and SBSs within contact durations, as follows:

$$\underset{< A, w_x^L >}{\text{maximize}} \quad : \sum_{x=1}^{4} g(A, w_x^L) \tag{3.33a}$$

$$\text{subject to} \quad \sum_{z=1}^{F} a_{i,z}.f_{l_z} \leq C_{u_i}, \ \ \forall i \in \{1, \cdots, M\}, \tag{3.33b}$$

$$\sum_{z=1}^{F} a_{j,z}.f_{l_z} \leq C_{s_j}, \ \ \forall j \in \{1, \cdots, N\}, \tag{3.33c}$$

$$a_{k,z}^{UT} \in [0,1], \ \ \forall k \in \{1, \cdots, M\}, \tag{3.33d}$$

$$a_{j,z}^{SBS} \in [0,1], \ \ \forall j \in \{1, \cdots, N\}, \tag{3.33e}$$

$$\sum_{x=1}^{4} w_x^L = 1, \tag{3.33f}$$

$$w_1^L, w_2^L, w_3^L, \text{ and } w_4^L > 0, \tag{3.33g}$$

where

$$g(A, w_x^L) = w_1^L.f_1 + w_2^L.f_2 - w_3^L.f_3 + w_4^L.f_4, \tag{3.34}$$

and C_{u_i} and C_{s_j} are the cache storage capacity of UT u_i and SBS s_j, respectively.

The objective function in (4.1a) represents the policy to cache placement in UTs and SBSs such that the cache hit rate is maximized. This is carried out by proactively caching the popular contents and ensuring minimum latency by placing the contents closer to UTs before their requests depending on their location within the macrocell. The constraints (4.1b) and (4.1c) mean that the total files stored in UT and SBS can not exceed the storage capacity of UT cache C_{u_i} and the storage capacity of the SBS cache C_{s_i}, respectively. The inequalities in (4.1d) and (4.1e) indicate the non-negativeness of the optimization variables, which means that we either do not cache or we cache one complete file in each cache. The last constraints (4.1f) and (4.1g) indicate that the summation of all the weights equals one and weights are positive, respectively. The solution to (4.1a) depends on the cache content model and mobility model presented in sections 3.2.2 and 3.2.3 respectively, number of UTs, and number of SBSs in one cell.

3.3.2 Latency and Energy Efficient Caching

After placing the cache contents based on files popularity and users' mobility, files are stored in UTs and SBSs caches. Then, users send their requests to download the file. At this point, the problem of deciding from where to download the requested file includes two essential objectives to achieve latency and energy-efficient operation. The first objective is to download the file with minimum download time, and the second objective is to deliver the file to the UT with minimum transmission power.

For cache access phase, (3.6) and (3.7) can be used to compute the download time depending on both file f_z length and data rate between UT u_i and another UT u_k and between UT u_i and SBS s_j, respectively. Then, $T_d^{UT,SBS}$ is a matrix of size $(M \times (M+N))$ that represents the time required for UT u_i to download the file f_z from another UT u_k and/or from SBS s_j respectively, such that:

$$
{T_d}^{UT,SBS} = \begin{bmatrix}
0 & t^{UT}_{d_{1,2}} & \cdots & t^{UT}_{d_{1,M}} & t^{SBS}_{d_{1,1}} & t^{SBS}_{d_{1,2}} & \cdots & t^{SBS}_{d_{1,N}} \\
t^{UT}_{d_{2,1}} & 0 & \cdots & t^{UT}_{d_{2,M}} & t^{SBS}_{d_{2,1}} & t^{SBS}_{d_{2,2}} & \cdots & t_{d^{SBS}_{2,N}} \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
t^{UT}_{d_{M,1}} & t^{UT}_{d_{M,2}} & \cdots & 0 & t^{SBS}_{d_{M,1}} & t^{SBS}_{d_{M,2}} & \cdots & t^{SBS}_{d_{M,N}}
\end{bmatrix}. \tag{3.35}
$$

For the cache delivery phase, if the file is cached in UT and/or SBS caches, then we can compute the energy consumption to transfer the file f_z from another UT and/or from SBS through D2D communication and SBS communication using (3.10) and (3.13), respectively. Then, $\mathcal{E}^{UT,SBS}$ is a matrix of size $(M \times (M+N))$ that represents the the transmission power required to transfer the file f_z from UT u_k and/or from SBS s_j to UT u_i respectively, such that:

$$
\mathcal{E}^{UT,SBS} = \begin{bmatrix}
0 & e^{UT}_{1,2} & \cdots & e^{UT}_{1,M} & e^{SBS}_{1,1} & e^{SBS}_{1,2} & \cdots & e^{SBS}_{1,N} \\
e^{UT}_{2,1} & 0 & \cdots & e^{UT}_{2,M} & e^{SBS}_{2,1} & e^{SBS}_{2,2} & \cdots & e^{SBS}_{2,N} \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
e^{UT}_{M,1} & e^{UT}_{M,2} & \cdots & 0 & e^{SBS}_{M,1} & e^{SBS}_{M,2} & \cdots & e^{SBS}_{M,N}
\end{bmatrix}. \tag{3.36}
$$

Then, achieve minimum time to file f_z from another UT u_k and/or from SBS s_j into UT u_i, is formulated as follows:

$$\underset{<x>}{\arg\min} : q_1^{i,z}(x)[a] \tag{3.37}$$

$$\text{subject to} : a_{k,z}^{UT} = 1, \forall k \in \{1, \cdots, M\}, [b] \tag{3.38}$$

$$a_{j,z}^{SBS} = 1, \forall j \in \{1, \cdots, N\}, [c] \tag{3.39}$$

where

$$q_1^{i,z} = T_d^{UT,SBS^*}(i,:) \odot A(:,z)^T \quad \forall i \in \{1, \cdots, M\}, \quad \forall z \in \{1, \cdots, F\}. \tag{3.40}$$

T_d^{UT,SBS^*} denotes the normalized download time. The objective function (3.37) helps to decide from which caches to download the file f_z resulting in minimum download time. Constraints (3.38) and (3.39) guarantee the file f_z is cached in UT u_k cache and/or SBS s_j cache.

The second objective function that aims to achieve minimum energy consumption for delivering the file f_z from UT u_k and/or SBS s_j to UT u_i can be formulated as follows:

$$\underset{<x>}{\arg\min} : \quad q_2^{i,z}(x)[a] \tag{3.41a}$$

$$\text{subject to} : \quad 0 < \mathcal{P}_T^{u_k} \leq \mathcal{P}^{u_k}{}_{max}, \forall k \in \{1, \cdots, M\}, [b] \tag{3.41b}$$

$$0 < \mathcal{P}_T^{s_j} \leq \mathcal{P}_{max}^{s_j}, \forall j \in \{1, \cdots, N\}.[c] \tag{3.41c}$$

where

$$q_2^{i,z} = \mathcal{E}^{UT,SBS^*}(i,:) \odot A(:,z)^T \quad \forall i \in \{1, \cdots, M\}, \quad \forall z \in \{1, \cdots, F\}, \tag{3.42}$$

$\mathcal{E}^{UT,SBS^*}$ is the normalized energy consumption in D2D communication and in SBS communication defined in (3.10) and (3.13), respectively. The objective function in (3.41a) aims to minimize the energy consumption for delivering the file f_z to UT u_i from UT u_k and/or from SBS s_j. The symbol $\odot$ indicates an element by element multiplication (Hadamard product). The constraint in (3.41b) and (3.41c) guarantee the transmit power $\mathcal{P}_T^{u_k}$ of UT u_k and the transmit power $\mathcal{P}_T^{s_j}$ of SBS s_j are less than or equal to $\mathcal{P}^{u_k}{}_{max}$ and $\mathcal{P}_{max}^{s_j}$, respectively.

For the latency and energy-efficient caching, we can use the weighted sum method to solve the multi-objective functions given in (3.37) and (3.41a) by selecting weights w_y^E $(y = 1, 2)$ for each objective function that aims to achieve latency and energy-efficient caching. Values in (3.35) and

(3.50) are measured on different scales. Thus, we need to normalize both matrices by scaling all values between 0 and 1. Then, we formulate one objective function for cache access and cache delivery that minimizes the latency and energy consumption for accessing and delivering the requested files from nearby UTs and SBSs, respectively, as follows:

$$
\begin{aligned}
&\underset{<x>}{\arg\min} : q^{i,z}(x)[a] && (3.43a)\\
\text{subject to :}\quad & a^{UT}_{k,z} = 1, k \in \{1, \cdots, M\}, [b] && (3.43b)\\
& a^{SBS}_{j,z} = 1, j \in \{1, \cdots, N\}, [c] && (3.43c)\\
& 0 < \mathcal{P}^{u_k}_{T} \le \mathcal{P}^{u_k}{}_{max}, k \in \{1, \cdots, M\}, [d] && (3.43d)\\
& 0 < \mathcal{P}^{s_j}_{T} \le \mathcal{P}^{s_j}_{max}, j \in \{1, \cdots, N\}, [e] && (3.43e)\\
& w^{E}_{1} + w^{E}_{2} = 1, [f] && (3.43f)\\
& w^{E}_{1} \text{ and } w^{E}_{2} > 0, [g] && (3.43g)
\end{aligned}
$$

where

$$
\begin{aligned}
q(x)^{i,z} &= w^{E}_{1} \times q1 + w^{E}_{2} \times q2\\
&= w^{E}_{1} \times (T^{UT,SBS^{*}}_{d}(i,:) \odot A(:,z)^{T}) + w^{E}_{2} \times (\mathcal{E}^{UT,SBS^{*}}(i,:) \odot A(:,z)^{T}) \qquad (3.44)\\
&\forall i \in \{1, \cdots, M\}, \forall z \in \{1, \cdots, F\}.
\end{aligned}
$$

The objective function in (3.43a) represents the policy to cache access and cache delivery from UTs and SBSs such that minimizing the latency to download a requested file f_z and minimizing the transmission power of the same file f_z. The constraints (3.43b) and (3.43c) guarantee the file f_z is cached in UT u_k and/or SBS s_j, respectively. The constraint in (3.43d) guarantees the transmit power $\mathcal{P}^{u_k}_{T}$ of UT u_k is less than or equal to $\mathcal{P}^{u_k}{}_{max}$. The constraint in (3.43e) guarantees the transmit power $\mathcal{P}^{s_j}_{T}$ of SBS s_j is less than or equal to $\mathcal{P}^{s_j}_{max}$. The last two constraints (3.43f) and (3.43g) indicate that the summation of the weights equals one and weights are positive, respectively. The solution to (3.43a) depends on the transmission model and power consumption model presented in sections (3.2.4) and (3.2.5), respectively, number of UTs, and number of SBSs in one cell.

3.4 Illustrative Example

To illustrate how to formulate the caching problem and find the solution in MEN, consider an example of cache placement, cache access, and cache delivery at the edge of MEN. Assume a small cell network having one MBS, five mobile UTs (M=5), three SBS ($N = 3$), and 50 files in the library ($F = 50$). Considering the contact duration between UTs and between UTs and SBSs, communication ranges of mobile users in D2D communication and UT to SBS communication, and content popularity that follows Zipf popularity distribution to decide which contents should be cached at the edge of the network. Figure (3.3) shows an illustration of this example.

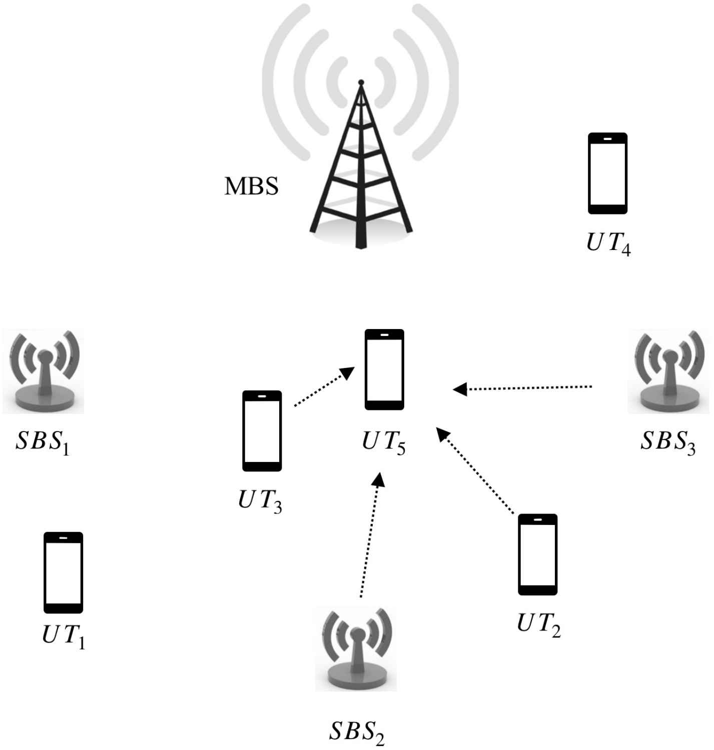


Figure 3.3: An illustration example of caching problem in mobile edge network (MEN).

The cache phases can be explained through the following steps:

3.4.1 Step 1: Cache Placement Phase

This step is before user requests. Assume, at time t_0, the set of locations L^{UT} of the UTs are variable values and are given, such that

$L^{UT} = \{l_1^{UT}, l_2^{UT}, l_3^{UT}, l_4^{UT}, l_5^{UT}\}$

and the set of locations L^{SBS} of the SBSs are fixed values and equals to:

$L^{SBS} = \{l_1^{SBS}, l_2^{SBS}, l_3^{SBS}\}$

At time t, we can compute the contact durations of UTs and SBSs given in (3.2) and (3.3), respectively. The contact duration for 5 UTs and 3 SBS results in a matrix $(\mathcal{T}^{UT,SBS})5\times8$ which includes the contact durations between all UTs and other UTs and SBSs within one cell, such that:

$$\mathcal{T}^{UT,SBS} = \begin{bmatrix} 0 & t_{1,2}^{UT} & t_{1,3}^{UT} & t_{1,4}^{UT} & t_{1,5}^{UT} & t_{1,1}^{SBS} & t_{1,2}^{SBS} & t_{1,3}^{SBS} \\ t_{2,1}^{UT} & 0 & t_{2,3}^{UT} & t_{2,4}^{UT} & t_{2,5}^{UT} & t_{2,1}^{SBS} & t_{2,2}^{SBS} & t_{2,3}^{SBS} \\ t_{3,1}^{UT} & t_{3,2}^{UT} & 0 & t_{3,4}^{UT} & t_{3,5}^{UT} & t_{3,1}^{SBS} & t_{3,2}^{SBS} & t_{3,3}^{SBS} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ t_{5,1}^{UT} & t_{5,2}^{UT} & t_{5,3}^{UT} & t_{5,4}^{UT} & 0 & t_{5,1}^{SBS} & t_{5,2}^{SBS} & t_{5,3}^{SBS} \end{bmatrix}. \tag{3.45}$$

The D2D communication range d^{D2D} between UT u_i and UT u_k and the communication range d^{SBS} between UT u_i and SBS S_j can be computed as given in (3.18) and (3.19), respectively, to form matrix $(\mathcal{D}^{\mathcal{UT},\mathcal{SBS}})_{5\times8}$ that can be written as follows:

$$\mathcal{D}^{UT,SBS} = \begin{bmatrix} 0 & d_{1,2}^{UT} & d_{1,3}^{UT} & d_{1,4}^{UT} & d_{1,5}^{UT} & d_{1,1}^{SBS} & d_{1,2}^{SBS} & d_{1,3}^{SBS} \\ d_{2,1}^{UT} & 0 & d_{2,3}^{UT} & d_{2,4}^{UT} & d_{2,5}^{U,T} & d_{2,1}^{SBS} & d_{2,2}^{SBS} & d_{2,3}^{SBS} \\ d_{3,1}^{UT} & d_{3,2}^{UT} & 0 & d_{3,4}^{UT} & d_{3,5}^{UT} & d_{3,1}^{SBS} & d_{3,2}^{SBS} & d_{3,3}^{SBS} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ d_{5,1}^{UT} & d_{5,2}^{UT} & d_{5,3}^{UT} & d_{5,4}^{UT} & 0 & d_{5,1}^{SBS} & d_{5,2}^{SBS} & d_{5,3}^{SBS} \end{bmatrix}. \tag{3.46}$$

The contents are requested based on their popularity by following Zipf popularity distribution given in (3.1). The total number of files in the library is given as 50 files ($F = 50$) such that $f =$

$\{f_1, f_2, \cdots, f_{50}\}$. Each user requests contents f_z randomly and independently with the probability of p is a vector of size $(1 \times F)$ such that:

$$p = p_1, p_2, \cdots, p_{50}, \tag{3.47}$$

where p_1 represents the most popular file with rank 1 and p_{50} represents the least popular file with rank 50, respectively. Then, most popular files are placed in cache locations selected by the cache placement algorithm. According to the mobility model discussed in section (3.2.3), the number of contacts between UTs and other UTs and SBSs follows the Poisson distribution. So then, $P_m^{UT,SBS}$ is a matrix of size (5×8) that represents the probability of UT to contact with other UTs or SBSs, and can be written as follows:

$$P_m^{UT,SBS} = \begin{bmatrix} p^{UT}_{m_{1,1}} & p^{UT}_{m_{1,2}} & p^{UT}_{m_{1,3}} & p^{UT}_{m_{1,4}} & p^{UT}_{m_{1,5}} & p^{SBS}_{m_{1,1}} & p^{SBS}_{m_{1,2}} & p^{SBS}_{m_{1,3}} \\ p^{UT}_{m_{2,1}} & p^{UT}_{m_{2,2}} & p^{UT}_{m_{2,3}} & p^{UT}_{m_{2,4}} & p^{UT}_{m_{2,5}} & p^{SBS}_{m_{2,1}} & p^{SBS}_{m_{2,2}} & p^{SBS}_{m_{2,3}} \\ p^{UT}_{m_{3,1}} & p^{UT}_{m_{3,2}} & p^{UT}_{m_{3,3}} & p^{UT}_{m_{4,4}} & p^{UT}_{m_{3,5}} & p^{SBS}_{m_{3,1}} & p^{SBS}_{m_{3,2}} & p^{SBS}_{m_{3,3}} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ p^{UT}_{m_{5,1}} & p^{UT}_{m_{5,2}} & p^{UT}_{m_{5,3}} & p^{UT}_{m_{5,4}} & p^{UT}_{m_{5,5}} & p^{SBS}_{m_{5,1}} & p^{SBS}_{m_{5,2}} & p^{SBS}_{m_{5,3}} \end{bmatrix}. \tag{3.48}$$

Given the size of caches in UT devices C_{u_i} such that $C_{u_i} = C_{u_1}, C_{u_2}, \cdots, C_{u_5}$ and SBSs C_{s_j} such that $C_{s_j} = C_{s_1}, C_{s_2}, C_{s_3}$, respectively. Files can be cached from main library into UT caches and SBS caches by designing a cache placement algorithm that implements the following tasks:

1. Input the datasets: $(p_z)_{(1X50)}, (\mathcal{D}^{UT,SBS})_{(5X8)}, (\mathcal{T}^{UT,SBS})_{(5X8)}, (P_m^{UT,SBS})_{(5X8)}, (C_{u_i})_{(1X5)}, and\ (C_{s_j})_{(1X5)}.$

2. Set the constraints specified in (4.1b)-(4.1g) as follows:

   - **Equation (4.1b):** The total size of files stored in each UT u_i cache must not exceed the corresponding cache size C_{u_i}

- **Equation (4.1c):** The total size of files stored in each SBS s_j cache must not exceed the corresponding cache size C_{s_j}
- **Equation (4.1d):** We have two options for the UT u_k cache, either store full file $a_{k,z}^{UT} = 1$ in the cache or do not store the file $a_{k,z}^{UT} = 0$ in the cache.
- **Equation (4.1e):** We have two options for the SBS s_j cache, either store full file $a_{j,z}^{SBS} = 1$ in the cache or do not store the file $a_{j,z}^{SBS} = 0$ in the cache.
- **Equation (4.1f):** The sum of weights of all objective functions must not exceed 1, $\sum_{x=1}^{4} w_x^L = 1$.
- **Equation (4.1g):** Weights of all objective functions must be positive numbers, w_1^L, w_2^L, w_3^L, and $w_4^L > 0$.

3. Solve the objective function in (3.25a): Select the files to be cached associated with the highest probability of popularity p_z. This can be done by finding all the possible solutions for each row in matrix A. For each UT and SBS, which represent one row in matrix A, we have ($2^F = 2^{50}$) possible solutions. We need to find a solution that maximizes the probability of storing the most popular files while satisfying the given problem constraints. The following example explains one possible solution for one cache (either UT cache or SBS ca che) associated with one row in matrix A.

Example 1 *Assume the files popularity is given as:*

$p_z = [0.97 \;\; 0.76 \;\; \cdots 0.58 \;\; \cdots \;\; 0.001]_{(1X50)}$

For the first UT u_1, one of the possible solutions can be:

$A(1,:) = [1 \;\; 1 \;\; \cdots 1 \;\; \cdots \;\; 0]_{(1X50)}$, *then*

$A(1,:) \times p_z^T = 0.97 + 0.76 + \cdots + 0.58 + \cdots + 0 = 2.31$

For the second UT u_2, one of the possible solutions can be:

$A(2,:) = [1 \;\; 0 \;\; \cdots \;\; 0]_{(1X50)}$, *then*

$A(2,:) \times p_z^T = 0.97 + 0 + \cdots + 0 = 0.97$

This process is carried out for all rows, and the resulting scores from all the rows $(2.31, 0.97, \cdots)$ *is added together. This process is repeated for different combinations of matrix A until we reach the maximum value of this accumulated score. The matrix A corresponding to this maximum score is our solution.*

4. Solve the objective function in (3.26a): For each column of matrix A, there are number of possible solutions to place 1 or 0 corresponding to cache or not cache the file f_z. Then for each column in matrix A, we have $(2^{(M+N)} = 2^{(5+3)} = 2^8)$ possible solutions. We need to find the solution by deciding where to store the file f_z to maximize the contact duration between each UT u_i and the corresponding UT u_k and/or SBS s_j in matrix $(\mathcal{T}^{UT,SBS})_{(5X8)}$. This means maximizing the probability of achieving successful communication. In the following example, one possible solution for one column of matrix A is explained:

Example 2 *Assume the contact duration between UT* u_3 *and other UTs and other SBSs is given in the third row of the matrix* $(\mathcal{T}^{UT,SBS})(5X8), as follows:$

$\mathcal{T}^{UT,SBS}(3,:) = [15.2\ \ 66.6\ \ 0\ \ 109\ \ 102\ \ 15.2\ \ 65.6\ \ 114.6]_{(1X8)}$

For the first file f_1*, one of the possible solutions to cache or not cache the file in all the available UT and SBS caches can be as follows:*

$A(:,1) = [0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1]^T_{(8X1)}$*, then*

$\mathcal{T}^{UT,SBS}(3,:) \times A(:,1) = 0 + 66.6 + 0 + 109 + 102 + 0 + 0 + 114.6 = 392.2$

For the second file f_2*, one of the possible solutions to cache or not cache the file in all the available UT and SBS caches can be as follows:*

$A(:,2) = [0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1]^T_{(8X1)}$*, then*

$\mathcal{T}^{UT,SBS}(3,:) \times A(:,2) = 0 + 66.6 + 0 + 109 + 102 + 0 + 65.6 + 114.6 = 457.8$

This process is carried out for all columns, and the resulting scores from all the columns $(392.2, 457.8, \cdots)$ *are added together. This process is repeated for different combinations of matrix A until we reach the maximum value of this accumulated score. The matrix A corresponding to this maximum score is our solution.*

5. Solve the objective function in (3.27a): For each column of matrix A, there are a number of possible solutions to place 1 or 0 corresponding to cache or not cache the file f_z. Then for each column in matrix A, we have $(2^{(M+N)} = 2^{(5+3)} = 2^8)$ possible solutions. We need to find the solution by deciding which place to store the file f_z to minimize the communication range between each UT u_i and the corresponding UT u_k and/or SBS s_j in matrix $(\mathcal{D}^{UT,SBS})_{(5X8)}$. This means maximizing the probability of achieving successful communication. In the following example, one possible solution for one column of matrix A is explained:

   Example 3 *Assume the communication range between UT u_4 and other UTs and other SBSs is given in the fourth row of the matrix $(\mathcal{D}^{UT,SBS})_{(5X8)}$, as follows:*

   $\mathcal{D}(4,:) = [22.56\ \ 11.58\ \ 129.14\ \ 0\ \ 17.4\ \ 22.6\ \ 11.5\ \ 129.14]_{(1X8)}$

   For the first file f_1, one of the possible solutions to cache or not cache this file in all the available UT and SBS caches can be as follows:

   $A(:,1) = [1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0]^T_{(8X1)}$*, then*

   $\mathcal{D}(4,:) \times A(:,1) = 22.56 + 0 + 129.14 + 0 + 17.4 + 0 + 0 + 0 = 169.1$

   For the second file f_2, one of the possible solutions to cache or not cache this file in all the available UT and SBS caches can be as follows:

   $A(:,2) = [0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0]^T_{(8X1)}$*, then*

   $\mathcal{D}(4,:) \times A(:,2) = 0 + 0 + 0 + 17.4 + 22.6 + 11.5 + 0 = 51.5$

   This process is carried out for all columns. The resulting scores from all the columns $(169.1, 51.5, \cdots)$ *are added together. This process is repeated for different combinations of matrix A until we reach the minimum value of this accumulated score. The matrix A corresponding to this maximum score is our solution.*

6. Solve the objective function in (3.28a): For each column of matrix A, there are a number of possible solutions to place 1 or 0 corresponding to cache or not cache the file f_z. Then for each column in matrix A, we have $(2^{(M+N)} = 2^{(5+3)} = 2^8)$ possible solutions. We need to find the solution by deciding which place to store the file f_z to maximize the probability that UT

u_i contacts with the corresponding UT u_k and/or SBS s_j in matrix $(P_m^{UT,SBS})_{(5X8)}$. In the following example, one possible solution for one column of matrix A is explained:

Example 4 *Assume the probability of contact between UT u_5 and other UTs and other SBSs is given in the fifth row of the matrix $(P_m^{UT,SBS})_{(5X8)}$, as follows:*

$P_m(5,:) = [0.51 \;\; 0.03 \;\; 0.16 \;\; 0.007 \;\; 0.103 \;\; 0.09 \;\; 0.1 \;\; 0]_{(1X8)}$

For the first file f_1, one of the possible solutions to cache or not cache the file in all the available UT and SBS caches can be as follows:

$A(:,1) = [0 \;\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 1 \;\; 0 \;\; 1]^T_{(8X1)}$*, then*

$P_m(5,:) \times A(:,1) = 0 + 0 + 0 + 0.007 + 0.103 + 0.09 + 0 + 0 = 0.2$

For the second file f_2, one of the possible solutions to cache or not cache this file in all the available UT and SBS caches can be as follows:

$A(:,2) = [1 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 0 \;\; 0]^T_{(1X8)}$*, then*

$P_m(5,:).A(:,2) = 0.51 + 0.03 + 0 + 0 + 0.103 + 0.09 + 0 + 0 = 0.733$

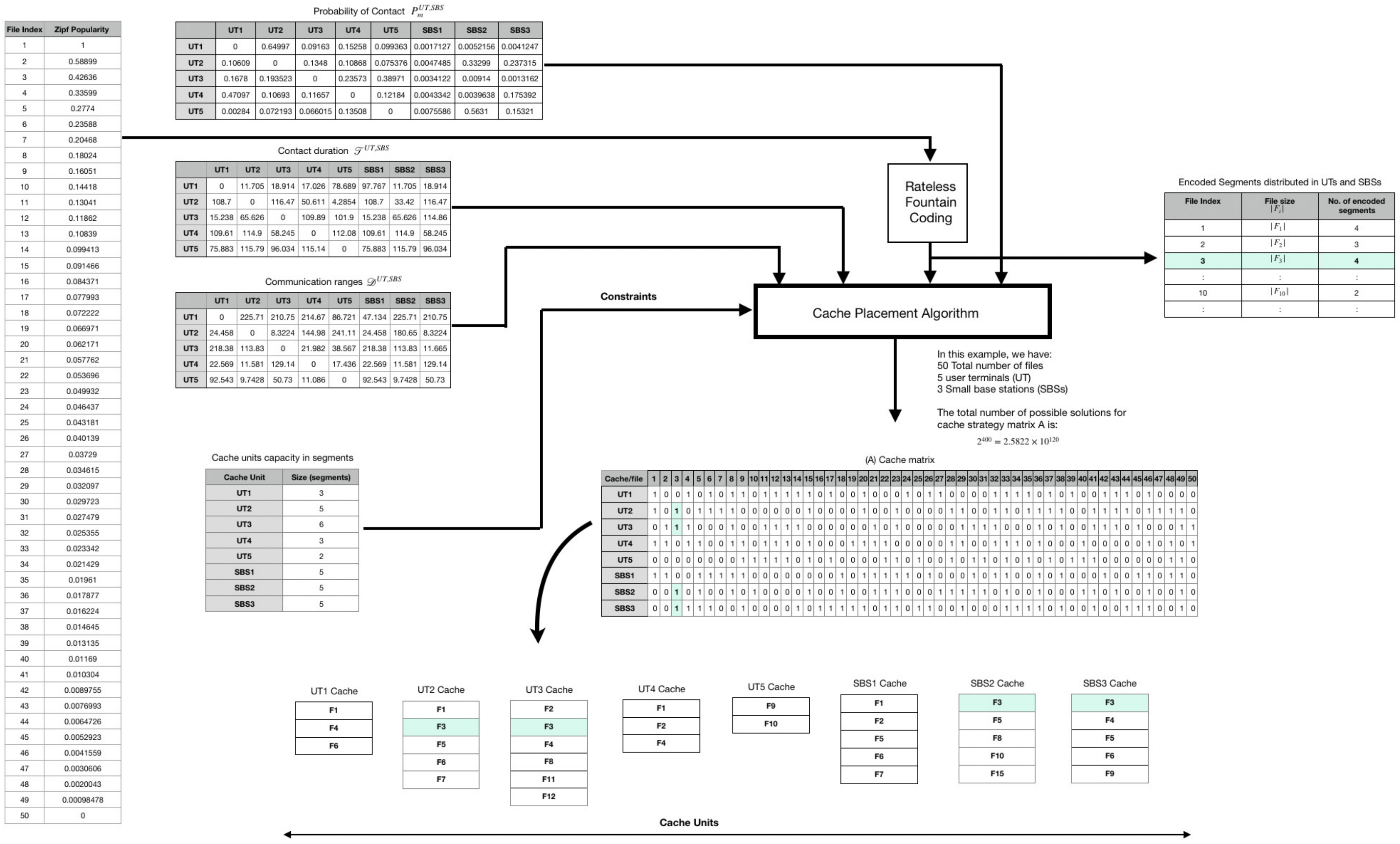
This process is carried out for all columns. The resulting scores from all the columns $(0.2, 0.733, \cdots)$ *are added together. This process is repeated for different combinations of matrix A until we reach the maximum value of this accumulated score. The matrix A corresponding to this maximum score is our solution.*

7. Repeat steps 3-6 to cover all the possible solutions (rows and columns of matrix A).

8. Solve the weighted sum of multi-objective function given in (4.1a): For matrix A, we have $2^{((M+N)XF)} = 2^{(8X50)} = 2^{400}$ possible solutions. Also, the weights of each objective function are a variable that can be optimized to select the settings that achieve the highest cache hit rate.

9. Use the selected solution of matrix A to store files in UT caches and SBS caches.

At the end of this phase, most popular files are placed in caches proactively prior to a user request. Figure 3.4 shows an example of the cache placement phase, illustrating all the input data sets, the output, and the contents of UT caches and SBS caches.



File Index	Zipf Popularity
1	1
2	0.58899
3	0.42636
4	0.33599
5	0.2774
6	0.23588
7	0.20468
8	0.18024
9	0.16051
10	0.14418
11	0.13041
12	0.11862
13	0.10839
14	0.099413
15	0.091466
16	0.084371
17	0.077993
18	0.072222
19	0.066971
20	0.062171
21	0.057762
22	0.053696
23	0.049932
24	0.046437
25	0.043181
26	0.040139
27	0.03729
28	0.034615
29	0.032097
30	0.029723
31	0.027479
32	0.025355
33	0.023342
34	0.021429
35	0.01961
36	0.017877
37	0.016224
38	0.014645
39	0.013135
40	0.01169
41	0.010304
42	0.0089755
43	0.0076993
44	0.0064726
45	0.0052923
46	0.0041559
47	0.0030606
48	0.0020043
49	0.00098478
50	0

Probability of Contact $P_m^{UT,SBS}$

	UT1	UT2	UT3	UT4	UT5	SBS1	SBS2	SBS3
UT1	0	0.64997	0.09163	0.15258	0.099363	0.0017127	0.0052156	0.0041247
UT2	0.10609	0	0.1348	0.10868	0.075376	0.0047485	0.33299	0.237315
UT3	0.1678	0.193523	0	0.23573	0.38971	0.0034122	0.00914	0.0013162
UT4	0.47097	0.10693	0.11657	0	0.12184	0.0043342	0.0039638	0.175392
UT5	0.00284	0.072193	0.066015	0.13508	0	0.0075586	0.5631	0.15321

Contact duration $\mathscr{T}^{UT,SBS}$

	UT1	UT2	UT3	UT4	UT5	SBS1	SBS2	SBS3
UT1	0	11.705	18.914	17.026	78.689	97.767	11.705	18.914
UT2	108.7	0	116.47	50.611	4.2854	108.7	33.42	116.47
UT3	15.238	65.626	0	109.89	101.9	15.238	65.626	114.86
UT4	109.61	114.9	58.245	0	112.08	109.61	114.9	58.245
UT5	75.883	115.79	96.034	115.14	0	75.883	115.79	96.034

Communication ranges $\mathscr{D}^{UT,SBS}$

	UT1	UT2	UT3	UT4	UT5	SBS1	SBS2	SBS3
UT1	0	225.71	210.75	214.67	86.721	47.134	225.71	210.75
UT2	24.458	0	8.3224	144.98	241.11	24.458	180.65	8.3224
UT3	218.38	113.83	0	21.982	38.567	218.38	113.83	11.665
UT4	22.569	11.581	129.14	0	17.436	22.569	11.581	129.14
UT5	92.543	9.7428	50.73	11.086	0	92.543	9.7428	50.73

Cache units capacity in segments

Cache Unit	Size (segments)
UT1	3
UT2	5
UT3	6
UT4	3
UT5	2
SBS1	5
SBS2	5
SBS3	5

Encoded Segments distributed in UTs and SBSs

File Index	File size $\lvert F_i \rvert$	No. of encoded segments
1	$\lvert F_1 \rvert$	4
2	$\lvert F_2 \rvert$	3
3	$\lvert F_3 \rvert$	4
:	:	:
10	$\lvert F_{10} \rvert$	2
:	:	:

In this example, we have:
50 Total number of files
5 user terminals (UT)
3 Small base stations (SBSs)

The total number of possible solutions for cache strategy matrix A is:

$$2^{400} = 2.5822 \times 10^{120}$$

(A) Cache matrix

Cache/file	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
UT1	1	0	0	1	0	1	0	1	1	0	1	1	1	1	1	0	1	0	0	1	0	0	0	1	0
UT2	1	0	1	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
UT3	0	1	1	1	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	1	0	1	0	0
UT4	1	1	0	1	1	0	0	1	1	0	1	1	1	0	1	0	0	0	1	1	1	1	0	0	0
UT5	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	0	0	0	0	1	1	0	1
SBS1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0
SBS2	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	1	1	1	0	0
SBS3	0	0	1	1	1	1	0	0	1	0	0	0	0	1	0	1	1	1	1	1	0	1	1	0	1

Cache/file	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
UT1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	1	0	1	0	0	0	0
UT2	0	0	1	1	0	0	1	1	0	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	0
UT3	0	0	0	1	1	1	1	0	0	0	1	0	1	0	0	1	1	1	0	1	0	0	0	1	1
UT4	0	0	1	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1
UT5	0	0	1	0	1	1	0	1	0	1	0	1	0	1	1	1	0	0	0	0	0	0	1	1	0
SBS1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	0	1	1	0	1	1	0
SBS2	0	1	1	1	1	1	0	1	0	0	1	0	0	0	1	1	0	1	0	0	1	0	0	1	0
SBS3	1	0	0	1	0	0	0	1	1	1	1	0	1	0	0	1	0	0	1	1	1	0	0	1	0

UT1 Cache	UT2 Cache	UT3 Cache	UT4 Cache	UT5 Cache	SBS1 Cache	SBS2 Cache	SBS3 Cache
F1	F1	F2	F1	F9	F1	F3	F3
F4	F3	F3	F2	F10	F2	F5	F4
F6	F5	F4	F4		F5	F8	F5
	F6	F8			F6	F10	F6
	F7	F11			F7	F15	F9
		F12					

Figure 3.4: An Illustration example of cache placement phase.

3.4.2 Step 2: User Request Contents

This step starts when user requests contents. User can obtain contents through local cache, D2D caching, SBS caching, or MBS caching. Deciding from where to download the file is made in the next phase. Figure 3.5 illustrates an example, where UT_5 requests file f_3. The file f_3 are stored in UT_2, UT_3, SBS_2, and SBS_3.

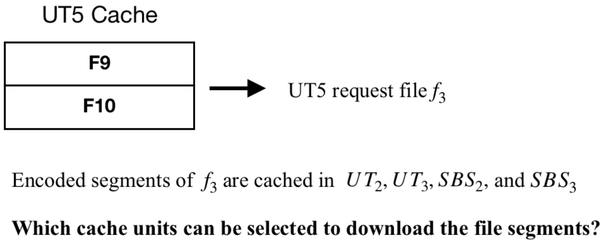

UT5 Cache
F9
F10
UT5 request file f_3
Encoded segments of f_3 are cached in UT_2, UT_3, SBS_2, and SBS_3
Which cache units can be selected to download the file segments?

Figure 3.5: An Illustration example of user request phase.

3.4.3 Step 3: Cache Access and Cache Delivery Phase

The cache system receives the request in this step and decides where to download the files to achieve energy-efficient cache access and delivery. Where to download the files depends on the minimum download time (cache access decision) and minimum transmission power (cache delivery decision).

From matrix A resulting from cache placement phase and the given size of files f_{l_z}, we compute the locations table of files in UTs and SBSs devices. Figure 3.6 shows an example of the locations table of the cached files.

Compute the download time of the stored file to be downloaded by UT u_i from another UT u_k and/or from SBS s_j using (3.6) and (3.7), respectively. Such that:

$$\mathcal{T}^{UT,SBS}{}_{d} = \begin{bmatrix} t^{UT}_{d_{1,1}} & t^{UT}_{d_{1,2}} & t^{UT}_{d_{1,3}} & t^{UT}_{d_{1,4}} & t^{UT}_{d_{1,5}} & t^{SBS}_{d_{1,1}} & t^{SBS}_{d_{1,2}} & t^{SBS}_{d_{1,3}} \\ \\ t^{UT}_{d_{2,1}} & t^{UT}_{d_{2,2}} & t^{UT}_{d_{2,3}} & t^{UT}_{d_{2,4}} & t^{UT}_{d_{2,5}} & t^{SBS}_{d_{2,1}} & t^{SBS}_{d_{2,2}} & t^{SBS}_{d_{2,3}} \\ \\ t^{UT}_{d_{3,1}} & t^{UT}_{d_{3,2}} & t^{UT}_{d_{3,3}} & t^{UT}_{d_{4,4}} & t^{UT}_{d_{3,5}} & t^{SBS}_{d_{3,1}} & t^{SBS}_{d_{3,2}} & t^{SBS}_{d_{3,3}} \\ \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \\ t^{UT}_{d_{5,1}} & t^{UT}_{d_{52}} & t^{UT}_{d_{5,3}} & t^{UT}_{d_{5,4}} & t^{UT}_{d_{5,5}} & t^{SBS}_{d_{5,1}} & t^{SBS}_{d_{5,2}} & t^{SBS}_{d_{5,3}} \end{bmatrix}. \tag{3.49}$$

Compute the transmission power between UT u_i that requests the file and other UT u_k and SBS s_j using (3.10) and (3.13), such that:

$$\mathcal{E}^{UT,SBS} = \begin{bmatrix} e^{UT}_{1,1} & e^{UT}_{1,2} & e^{UT}_{1,3} & e^{UT}_{1,4} & e^{UT}_{1,5} & e^{SBS}_{1,1} & e^{SBS}_{1,2} & e^{SBS}_{1,3} \\ \\ e^{UT}_{2,1} & e^{UT}_{2,2} & e^{UT}_{2,3} & e^{UT}_{2,4} & e^{UT}_{2,5} & e^{SBS}_{2,1} & e^{SBS}_{2,2} & e^{SBS}_{2,3} \\ \\ e^{UT}_{3,1} & e^{UT}_{3,2} & e^{UT}_{3,3} & e^{UT}_{3,4} & e^{UT}_{3,5} & e^{SBS}_{3,1} & e^{SBS}_{3,2} & e^{SBS}_{3,3} \\ \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \\ e^{UT}_{5,1} & e^{UT}_{5,2} & e^{UT}_{5,3} & e^{UT}_{5,4} & e^{UT}_{5,5} & e^{SBS}_{5,1} & e^{SBS}_{52} & e^{SBS}_{5,3} \\ \\ \end{bmatrix}. \tag{3.50}$$

Then the design of cache access and cache delivery algorithm requires implementing the following steps:

1. Input the datasets: $A_{(8X50)}, (\mathcal{T}^{UT,SBS}{}_{d})_{(5X8)}, (\mathcal{E}^{UT,SBS})_{(5X8)}$ and the files location table illustrated in (Figure (3.6)).

2. Set the constraints specified in (3.43b)-(3.43g) as follows:

   - **Equation (3.43b):** The file f_z must be cached in UT u_k in order to download them.

- **Equation (3.43c):** The file f_z must be cached in SBS s_j in order to download them.
- **Equation (3.43d):** The transmit power $\mathcal{P}_T^{u_k}$ of UT u_k must be less than or equal to $\mathcal{P}^{u_k}{}_{max}$ in order to select u_k to download the file.
- **Equation (3.43e):** The transmit power $\mathcal{P}_T^{s_j}$ of SBS s_j must be less than or equal to $\mathcal{P}_{max}^{s_j}$ in order to select SBS s_j to download the file.
- **Equation (3.43f):** The sum of weights of the two objective functions must not exceed 1, $w_1^E + w_2^E = 1$.
- **Equation (3.43g):** Weights of the two objective functions must be positive numbers, $w_1^E \; and \; w_2^E > 0$.

3. Solve the objective function in (3.37): For each UT $u_i, \forall i \in 1, 2, \cdots, M$ and each file $f_z \;\; \forall z \in 1, 2, \cdots, F$, there are $(M + N)$ possible locations to download from M UT and/or N SBS caches, if the files are cached. We need to select the locations that result in minimum download time. The aim is to achieve latency-efficient caching (this is the cache access objective). In the following example, the computation is performed for one user and one file to explain the procedure of selecting from where to download the requested file:

Example 5 *For the same example given in section 3.4.2, we assumed UT u_5 requests file f_3 and this file are stored in $u_2, u_3, s_2,$ and s_3.*

For UT u_5, the download time of the file from all other UTs and SBSs are given in the fifth row of the matrix $(\mathcal{T}^{UT,SBS}{}_d)_{(5X8)}$, as follows:

$\mathcal{T}_d^{UT,SBS^*}(5,:) = [63 \;\; 97 \;\; 80 \;\; 92 \;\; 63 \;\; 17 \;\; 9 \;\; 3]_{(1X8)}$.

Then, the normalized download time $\mathcal{T}_d^{UT,SBS^*}(5,:)$ is given as:

$\mathcal{T}_d^{UT,SBS^*}(5,:) = [0.6383 \;\; 1 \;\; 0.8191 \;\; 0.9468 \;\; 0.6383 \;\; 0.1489 \;\; 0.0638 \;\; 0]_{(1X8)}$

File f_3 status in matrix A is given in the third column of matrix A, such as:

$A(:,3)^T = [0 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 1]_{(1X8)}^T$, *then*

$\mathcal{T}_d^{UT,SBS^*}(5,:) \odot A(:,3)^T = [0 \;\; 1 \;\; 0.8191 \;\; 0 \;\; 0 \;\; 0 \;\; 0.0638 \;\; 0]_{(1X8)}$

Thus, we have multiple locations for the files, and we need to select the locations that result in minimum download time. In this example, the priority for downloading the files will be given to the cache location in the order s_3, s_2, u_3, and u_2, respectively.

4. Solve the objective function in (3.41a): For each UT $u_i, \forall i \in 1, 2, \cdots, M$ and each file f_z $\forall z \in 1, 2, \cdots, F$, there are $(M + N)$ possible locations to download from M UT and/or N SBS caches, if the files are cached. We need to select the locations that result in minimum energy consumption for delivering the file f_z. The aim is to achieve energy-efficient caching (this is the cache delivery objective). In the following example, the computation is performed for one user and one file to explain the procedure of selecting from where to download the requested file:

Example 6 *For the same example given in section 3.4.2, we assumed UT u_5 requests file f_3 and the file is stored in $u_2, u_3, s_2,$ and s_3.*

The energy consumption for delivering the files from all UTs in D2D communication and all from SBSs in SBS communication to UT u_5 are given in the fifth row of the matrix $(\mathcal{E}^{UT,SBS})_{(5X8)}$, as follows:

$\mathcal{E}^{UT,SBS}(5,:) = [214 \;\; 85 \;\; 72 \;\; 102 \;\; 53 \;\; 15 \;\; 6 \;\; 10]_{(1X8)}$.

Then, the normalized energy consumption $\mathcal{E}^{UT,SBS^*}(5,:)$ is given as:

$\mathcal{E}^{UT,SBS^*}(5,:) = [1 \;\; 0.3798 \;\; 0.3173 \;\; 0.4615 \;\; 0.2260 \;\; 0.0433 \;\; 0 \;\; 0.0192]_{(1X8)}$

File f_3 status in matrix A is given in the third column of matrix A, such as:

$A(:,3)^T = [0 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 1]_{(1X8)}$, *then*

$\mathcal{E}^{UT,SBS^*}(5,:) \odot A(:,3)^T = [0 \;\; 0.3798 \;\; 0.3173 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0.0192]_{(1X8)}$

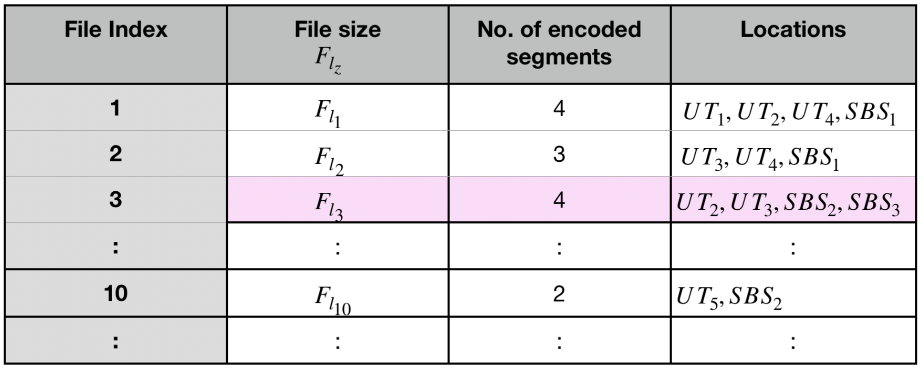
Thus, we have multiple locations for the files, and we need to select the locations that result in minimum energy consumption. In this example, the priority for downloading the files will be given to the cache location in the order s_2, s_3, u_3, and u_2, respectively.

5. Solve the weighted sum of multi-objective function given in (3.43a): Select from where to download the requested file. Also, the weight of each objective function (latency and energy) is a variable that can be optimized to select the settings that achieve higher performance.

6. Download the files from the selected locations.

7. If file f_z is not found in UT and SBS caches, then download the requested file from MBS.

8. Stop cache access and cache delivery algorithm.



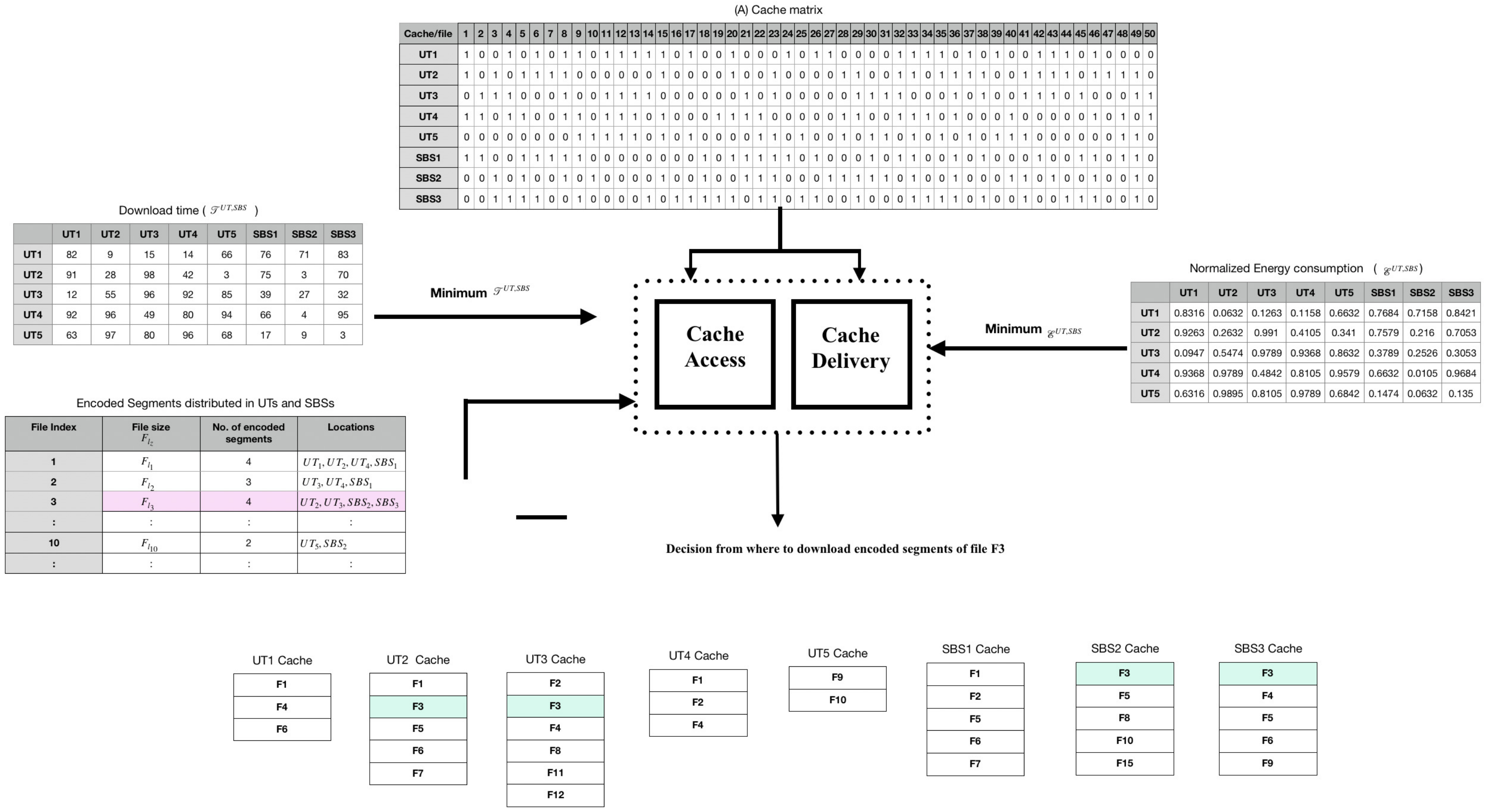
File Index	File size F_{l_z}	No. of encoded segments	Locations
1	F_{l_1}	4	UT_1, UT_2, UT_4, SBS_1
2	F_{l_2}	3	UT_3, UT_4, SBS_1
3	F_{l_3}	4	UT_2, UT_3, SBS_2, SBS_3
:	:	:	:
10	$F_{l_{10}}$	2	UT_5, SBS_2
:	:	:	:

Figure 3.6: An Illustration example of files distributed in UTs and SBSs caches.

Figure 3.7 shows an example of the cache access and cache delivery phase, illustrating all the input data sets, the output, and the contents of UT caches and SBS caches.

3.5 Summary

This chapter presented, mobile edge network (MEN) as a network model that enables the use of caching capabilities at the edge of the network in the macro base station (MBS), small base stations (SBSs), and user terminals (UTs). First, we defined the main system models, components, and relationship of system models. Then, problem formulation for latency efficient and energy-efficient caching optimization was proposed. Illustrative examples followed this for possible user requests, access, and delivery. The following chapter will use the weighted-sum fusion decision to solve the cache placement optimization problem formulated in this chapter.



(A) Cache matrix

Cache/file	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
UT1	1	0	0	1	0	1	0	1	1	0	1	1	1	1	1	0	1	0	0	1	0	0	0	1	0
UT2	1	0	1	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
UT3	0	1	1	1	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	1	0	1	0	0
UT4	1	1	0	1	1	0	0	1	1	0	1	1	1	0	1	0	0	0	1	1	1	1	0	0	0
UT5	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	0	0	0	0	1	1	0	1
SBS1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0
SBS2	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	1	1	1	0	0
SBS3	0	0	1	1	1	1	0	0	1	0	0	0	0	1	0	1	1	1	1	1	0	1	1	0	1

Cache/file	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
UT1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	1	0	1	0	0	0	0
UT2	0	0	1	1	0	0	1	1	0	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	0
UT3	0	0	0	1	1	1	1	0	0	0	1	0	1	0	0	1	1	1	0	1	0	0	0	1	1
UT4	0	0	1	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1
UT5	0	0	1	0	1	1	0	1	0	1	0	1	0	1	1	1	0	0	0	0	0	0	1	1	0
SBS1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	0	1	1	0	1	1	0
SBS2	0	1	1	1	1	1	0	1	0	0	1	0	0	0	1	1	0	1	0	0	1	0	0	1	0
SBS3	1	0	0	1	0	0	0	1	1	1	1	0	1	0	0	1	0	0	1	1	1	0	0	1	0

Download time ($\mathscr{T}^{UT,SBS}$)

	UT1	UT2	UT3	UT4	UT5	SBS1	SBS2	SBS3
UT1	82	9	15	14	66	76	71	83
UT2	91	28	98	42	3	75	3	70
UT3	12	55	96	92	85	39	27	32
UT4	92	96	49	80	94	66	4	95
UT5	63	97	80	96	68	17	9	3

Minimum $\mathscr{T}^{UT,SBS}$
Cache Access
Cache Delivery
Minimum $\mathscr{E}^{UT,SBS}$

Normalized Energy consumption ($\mathscr{E}^{UT,SBS}$)

	UT1	UT2	UT3	UT4	UT5	SBS1	SBS2	SBS3
UT1	0.8316	0.0632	0.1263	0.1158	0.6632	0.7684	0.7158	0.8421
UT2	0.9263	0.2632	0.991	0.4105	0.341	0.7579	0.216	0.7053
UT3	0.0947	0.5474	0.9789	0.9368	0.8632	0.3789	0.2526	0.3053
UT4	0.9368	0.9789	0.4842	0.8105	0.9579	0.6632	0.0105	0.9684
UT5	0.6316	0.9895	0.8105	0.9789	0.6842	0.1474	0.0632	0.135

Encoded Segments distributed in UTs and SBSs

File Index	File size F_{l_z}	No. of encoded segments	Locations
1	F_{l_1}	4	UT_1, UT_2, UT_4, SBS_1
2	F_{l_2}	3	UT_3, UT_4, SBS_1
3	F_{l_3}	4	UT_2, UT_3, SBS_2, SBS_3
:	:	:	:
10	$F_{l_{10}}$	2	UT_5, SBS_2
:	:	:	:

Decision from where to download encoded segments of file F3

UT1 Cache	UT2 Cache	UT3 Cache	UT4 Cache	UT5 Cache	SBS1 Cache	SBS2 Cache	SBS3 Cache
F1	F1	F2	F1	F9	F1	F3	F3
F4	F3	F3	F2	F10	F2	F5	F4
F6	F5	F4	F4		F5	F8	F5
	F6	F8			F6	F10	F6
	F7	F11			F7	F15	F9
		F12					

Figure 3.7: An Illustration example of cache access and delivery phase.

Chapter 4

Cache Placement Using Weighted-Sum Fusion Decision

4.1 Introduction

Cache placement at the edge of the network faces several challenges due to continuously changing content popularity, user mobility, and the number of users within each network. In addition, more challenges appear in caching at MENs due to high computation requirements of future applications that need to satisfy power and delivery time constraints. Existing studies are based on user mobility to carry out cache placement at the edge of the wireless network. In [194], the authors study the influence of using the statistical traffic and users' context information on the resources needed and network locations for pre-caching the contents. The considered information includes file popularity, user location, and mobility pattern. The authors propose two procedures for cache placement: the first one studies the correlation between users and files and uses a support vector machine (SVM) to predict which contents to place on SBS caches depending on users' preferences. The second procedure studies the social context of cache placement by placing contents on a set of user devices considered influential users within the network. Then D2D communications can be established between those users and other users in the network. Their results highlighte the importance of investigating context awareness and social ties between users to reduce peak data traffic demands via proactive caching. The authors in [123] consider context-aware proactive caching. The authors formulate a multi-armed bandit optimization problem that considers user mobility by computing connected users' preferences and users' service priorities within a specific time slot. The content popularity is varied over time by considering the movement of users and the connection of a new set of users. The algorithm proposed by the authors uses online learning of context-specific content popularity of connected users, which can include personal preferences, equipment, and user service priority provided by an operator. In [75], the authors use the transfer learning approach to learn the content popularity. First, they classify the training dataset using a K-means clustering algorithm based on access feature correlation during the specific time interval. Then, the algorithm classifies the content on the same cluster with a more considerable access feature correlation. They formulate a proactive content caching optimization model to minimize the average transmission cost while increasing the cache hit rate. The authors used a greedy algorithm to solve the proactive cache problem.

In [137], the authors consider realistic cache placement features by describing user mobility via a set of frequently visited locations within a cell, where one or more SBSs cover each location. The influence of mobility-awareness in cache placement algorithm is studied in [109]. The authors formulate the problem of coded segments caching at base stations (BSs) and user terminals (UTs), considering users' mobility and the amount of data content per transmission. User mobility is presented as a peer-to-peer connectivity model based on the parameters of contact duration and frequency. The problem is formulated as an integer programming problem, which is solved using sub-modular optimization. In [195], authors propose proactively loading content based on the balance between user trajectories and traffic requirements. The contents are placed in cells other than the congested cells. User mobility patterns are modeled using a semi-Markov renewal process.

In most of the work above, the impact of user mobility on caching strategy has been considered from one perspective: users' content preferences, frequently visited places by users, users' social context, content segmentation and placement of the segments in multiple cache locations, or users' contact probabilities. However, cache placement performance improvement by simultaneously considering different user mobility attributes and their relationship can directly affect cache placement decisions is a challenging problem that calls for detailed investigations. In this chapter, we propose a mobility-aware latency-efficient cache placement algorithm based on weighted fusion decisions.

This chapter investigates a new formulation of the cache placement problem, considering four objectives to place contents in SBSs and UT caches. The multi-objective function sets the advantages of user mobility patterns to decide on each SBS and UT cache content. The problem is formulated as a weighted fusion decision with four objectives related to user mobility computed from recorded data sets and one related to content popularity.

The aim of the formulated multi-objective function is cache hit rate maximization. This aim is achieved by maximizing file content popularity, maximizing the contact duration between a mobile user and the cache where the desired contents are stored, minimizing the communication range between the mobile user and cache, and maximizing the UT contact probability. Identification of the effect of user mobility on the cache placement performance is an essential feature of this new formulation. User mobility is considered by analyzing the impact of user speed, user mobility patterns through different paths within one cell to identify highly used paths in the wireless network, user contact probability with other users and SBSs, and communication ranges between previous users and SBSs locations along previous used paths.

The simulation results are computed for up to $10,000$ user requests from mobile users from $10,000$ random locations over three different paths in a single wireless network cell. The file requests are made according to the Zipf popularity distribution. The evaluation of our work is implemented by comparing the results of our proposed cache placement algorithm with three different cache placement techniques. These techniques are popularity, random, and mobility cache placement.

Our results describe the impact of user mobility attributes on increasing the cache hit rate, which decreases the latency of downloading the requested data content. Furthermore, the results show the impact of user mobility attributes on reducing the consumed energy for transmitting the contents to the UT.

4.2 Cache Placement Based On Decision Theory

The weighted sum method is used to solve the multi-objective functions given in (3.25a), (3.26a), (3.27a), and (3.28a) and achieve latency efficient cache placement. Different weights w_x ($x = 1, \cdots, 4$) can be chosen for each objective function in order to achieve latency efficient caching. Subsequently, for M UTs and N SBSs, a single objective function for cache placement is formulated that maximizes the probability that a UT u_i downloads the requested files from UTs and SBSs are within the contact durations. An objective function for cache placement is formulated, maximizing the probability that a UT u_i downloads the requested files from UTs and SBSs within the contact durations. The objective function can be written as follows:

$$\underset{< A, w_x >}{\text{maximize}} \quad \sum_{x=1}^{4} g^*(A, w_x) \tag{4.1a}$$

subject to

$$\sum_{z=1}^{F} a_{i,z}.f_{l_z} \le C_{u_i}, \forall i \in \{1, \cdots, M\} \tag{4.1b}$$

$$\sum_{z=1}^{F} a_{j,z}.f_{l_z} \le C_{s_j}, \forall j \in \{1, \cdots, N\} \tag{4.1c}$$

$$a_{k,z}^{UT} \in [0, 1], \forall k \in \{1, \cdots, M\} \tag{4.1d}$$

$$a_{j,z}^{SBS} \in [0, 1], \forall j \in \{1, \cdots, N\} \tag{4.1e}$$

$$\sum_{x=1}^{4} w_x = 1 \tag{4.1f}$$

$$w_1, w_2, w_3, \text{ and } w_4 > 0, \tag{4.1g}$$

The weighted sum of the normalized objective values is as follows:

$$\begin{aligned} g^*(A, w_x) = w_1. &\sum_{i=1}^{M+N} A(i, :) \times p_z^T \;\; \forall z \in \{1, \cdots, F\}, \\ + w_2. &\sum_{i=1}^{M} \sum_{z=1}^{F} \mathcal{T}^{*UT,SBS}(i, :) \times A(:, z) \\ - w_3. &\sum_{i=1}^{M} \sum_{z=1}^{F} \mathcal{D}^{*UT,SBS}(i, :) \times A(:, z) \\ + w_4. &\sum_{i=1}^{M} \sum_{z=1}^{F} P_m^{UT,SBS}(i, :) \times A(:, z), \end{aligned} \tag{4.2}$$

where, p_z^T, $\mathcal{T}^{*UT,SBS}$, $\mathcal{D}^{*UT,SBS}$, and $P_m^{UT,SBS}$ represent the file popularity probability, normalized contact duration, normalized communication range, and contact probability, respectively.

The objective function in (4.1a) represents the cache placement strategy for UTs and SBSs such that the cache hit rate is maximized. This is achieved by proactively caching popular contents and ensuring minimum latency by placing contents in the vicinity of the UTs before their requests, depending on their locations within a macrocell. The constraints in (4.1d) and (4.1e) indicate that the entries of the solution matrix are binary variables, corresponding to either caching or not caching a complete file unit in each cache. Finally, the last constraints (4.1f) and (4.1g) make sure that the sum of all the weights equals one and that individual weights are positive, respectively.

The problem of deciding optimal cache placement at SBSs to maximize the probability of finding the desired files through its local SBSs without using backhaul links or downloading from MBS (which means to minimize the total overall delay of all UTs) has been proven in [196] as an NP-hard problem.

In [137], it was shown that the caching problem of files in SBSs while compromising users moving in and out of the SBS coverage area to minimize the probability that the request reaches MBS (which means to maximize cache hit rate) is an NP-hard problem. While in [197], it has been proved that the cache placement is an NP-hard problem for caching in UTs through D2D communication to maximize the total amount of data offloaded to caches. Their proof considers input parameters: the mobility of UTs, files popularity, and cache storage capacity.

The problem targeted in our work aims to find which files to cache at SBSs and UTs to maximize the cache hit rate considering users' mobility, content popularity, and cache storage capacity. Therefore, the cache placement algorithm defined in (4.1a) is an NP-hard problem. Therefore, we need to find an efficient solution to build a model that can learn the hidden features in the input data sets, features of system attributes and their relationships, the relationship between cache placement in previous decisions, and their input attributes to predict next decisions that may improve overall system performance.

The problem of cache placement involves a decision-making problem, where we have several possibilities on which contents to cache and where to cache these contents. We need to select one action among these possibilities based on the available information.

The main task of the decision theory is to design decision rules that optimize the objective functions subject to certain constraints. In our case, the cache placement model gives two possible decisions for each file f_z in the main library, either 1 or 0, indicating to cache or not to cache the file, respectively. In this approach, contact duration probability, communication range probability, and user terminal UT contact probability will be checked and multiplied by weight w_1, w_2, w_3, and w_4, respectively.

Then, the summation of the weighted and normalized variables is compared with a threshold value. The file will be placed only if the weighted sum is above a threshold. The weighted decision fusion algorithm steps are illustrated in Algorithm 1. The complexity of the cache placement problem increases with the increase in the number of users, the number of SBSs, and the number of files in MBS. We proposed a multi-objective optimization problem to reduce the complexity computation required to solve the cache placement algorithm to find matrix A of size $(M+N)\times F$. As a result, the complexity of computing each objective (steps 1 to 4) is reduced to $O(N^2)$. In the weighted-sum approach, each condition controls the effect of that condition on the final decision. The management and implementation of Algorithm 1 is centralized in the MBS.

Algorithm 1 : Cache Placement Based Weighted-Sum Approach

```
    Input: $(\mathcal{T}^{*UT})_{(MXM)}, (\mathcal{T}^{*SBS})_{(MXN)}, (\mathcal{D}^{*UT})_{(MXM)}, (\mathcal{D}^{*SBS})_{(MXN)}, (P_m^{UT})_{(MXM)},$
    $(P_m^{SBS})_{(MXN)}, (C_{u_i})_{(1XM)},\ (C_{s_j})_{(1XN)}$, and $p$.
1:  Find maximum of $(\mathcal{T}^{UT})_{(MXM)}$ and $(\mathcal{T}^{SBS})_{(MXN)}$
2:  Find minimum of $(\mathcal{D}^{UT})_{(MXM)}$ and $(\mathcal{D}^{SBS})_{(MXN)}$
3:  Find maximum of $(P_m^{UT})_{(MXM)}$ and $(P_m^{SBS})_{(MXN)}$
4:  Find maximum of $p$
5:  Compute the weighted-sum of the results obtained from steps 1, 2, and 3 to form matrix A
6:  Place selected files on SBS and UT caches following matrix A
    Output: Cache strategy matrix A
```

4.3 Simulation Results

Simulation results are provided to evaluate the impact of different variables on the cache placement performance in MEN with four different caching strategies:

- Popularity caching [109]: It assumes that we place the most popular contents in each SBS and UT cache.

- Random caching [109]: It assumes that we place contents in SBS and UT caches randomly.

- Mobility aware caching [137]: Where user mobility is modeled by a set of highly visited locations in the macrocell. A set of SBSs may cover each location. Users may connect with multiple SBSs at different times while on the move. Contents are delivered by considering user mobility and contact duration limits. Under this strategy, we assume the files are stored and delivered as one file.

- Weighted-sum approach caching: It assumes that in cache placement we place contents in SBSs and UTs depending on: file popularity probability, p, contact duration of UTs and SBSs, $\mathcal{T}^{UT,SBS}$, communication ranges, $\mathcal{D}^{UT,SBS}$, and user terminal UT contact probability $P_m^{UT,SBS}$.

The four strategies will be evaluated by computing the hit rate. The simulation is implemented on a certain subarea with area dimension, and a number of SBSs [197]. We assume the macrocell includes one MBS, 15 SBSs, and there are three different paths that users pass through as shown in Figure 4.1. The simulation results are computed for up to $10,000$ user requests sent from mobile users in $10,000$ random locations in different paths. The simulation parameters are illustrated in Table 4.1. One of the parameters is varied. The remaining are fixed during each simulation to compare the impact of system parameters on the performance of the three caching algorithms. The results are computed by taking the mean value of the measured values from three different paths. Initially, we assume all the parameters have equal weights, which means $w_1 = w_2 = w_3 = w_4 = 0.25$ and the threshold is 0.5 to take the 50% of the effect of each input attributes.

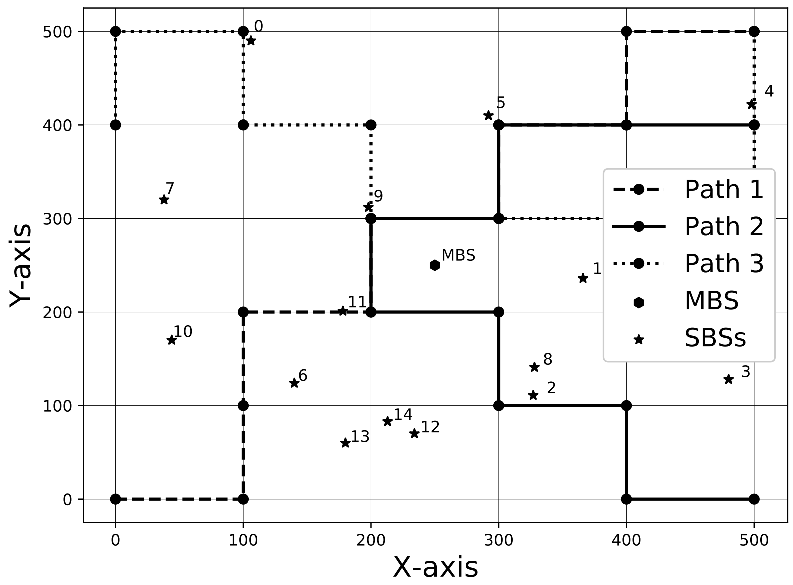


Figure 4.1: A 500m × 500m macro cell with one MBS, 15 SBSs, and three paths.

Table 4.1: Simulation parameters

Parameter	**Value**
Number of files in MBS	100
Number of SBSs (N)	5, 10, 15
SBSs cache size	(10-100)% of MBS library
UTs cache size	(50)% of SBSs cache size
File size	2 MB
SBS data rate	(2, 4, 8, 16)Mbps
SBS communication range	(50, 75, 100, 200)m
User speed	(10, 20, 30) m/sec
Number of user requests	10,000

We assume user mobility parameters follow the work in [109]. It is defined as an independent Poisson process, the pairwise contact duration between mobile UT and SBS follows the exponential distribution with parameter $\lambda_{i,j}^{SBS}$, such that $\Gamma(10, 1/100)$ represents the contact rate between mobile UT and SBS. The algorithms are implemented using Python 3 in Anaconda environment.

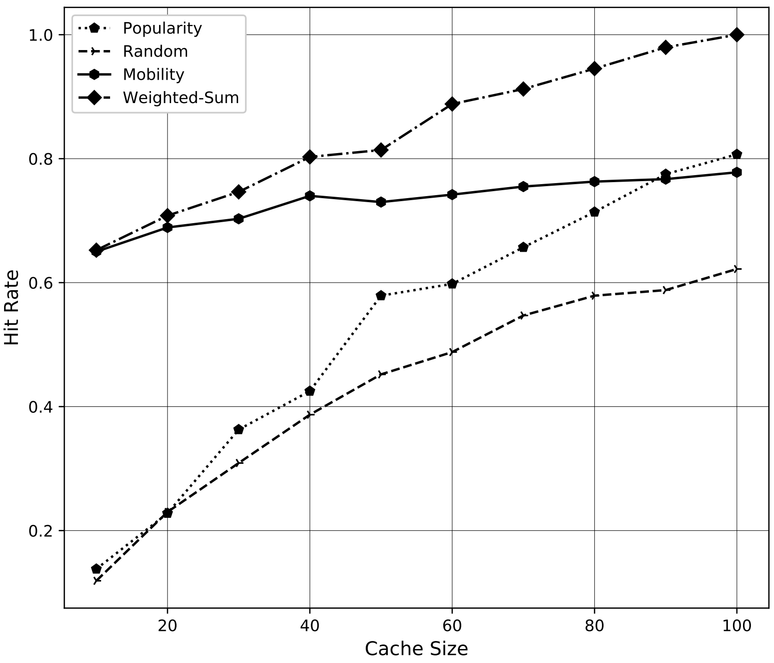


Figure 4.2: Impact of cache size (% of the total library size) on the hit rate.

4.3.1 Impact of Cache Size:

Figure 4.2 shows the impact of cache size, which can range from 10% to 100% of the total library size. As expected, increasing the cache size will increase the cache hit rates. This is because more files can be cached when the cache size is larger.

In all cache placement algorithms, the cache hit rate increases near linearly with the cache size increase. Mobility caching and weighted-sum approach perform better than popularity and random caching algorithm. The performance of weighted-sum caching is very close to mobility caching and outperforms it in larger cache size (cache size above 60% of the main library).

4.3.2 Impact of Number of SBSs:

Figure 4.3 shows the impact of cache size on the hit rate when number of SBSs $= 5, 10$, and 15. We observe the cache hit rate increases linearly with the increase of cache size and the increase of the number of SBSs. This is because when the number of SBS increases with the increase of cache size, more files are stored in the caches, increasing the possibilities of finding the requested content.

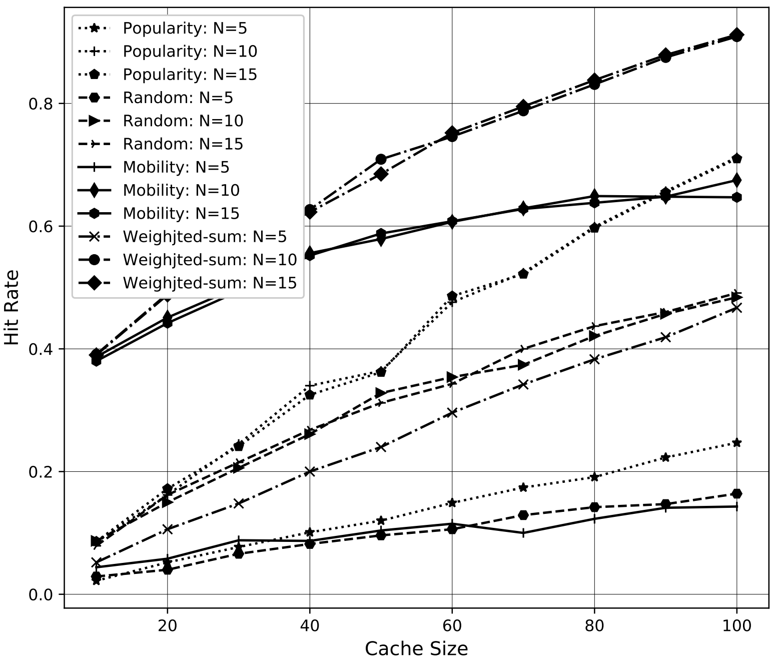


Figure 4.3: Impact of cache size (% of the total library size) on the hit rate with different number of SBSs.

The Weighted-sum approach outperforms other caching techniques for all number of SBSs.

4.3.3 Impact of SBS Data Rate:

We explore the impact of changing the SBS data rate on cache placement algorithm performance as shown in Figure 4.4. SBSs data rates vary from 4 to 16 Mbps. Increasing the SBS data rate increases the hit rate in cache placement algorithms since SBSs can transfer more data during contact with the UT. The hit rate in the weighted-sum approach outperforms other techniques for all data rates. When the data rate is 4 Mbps, the hit rate in popularity caching exceeds random and mobility caching. Hence, for higher data rates (8 and 16 Mbps), mobility caching outperforms the popularity and random caching. This is because increasing the SBS data rate with fixed file size allows the user to complete downloading the contents within available time while moving from one location to another location.

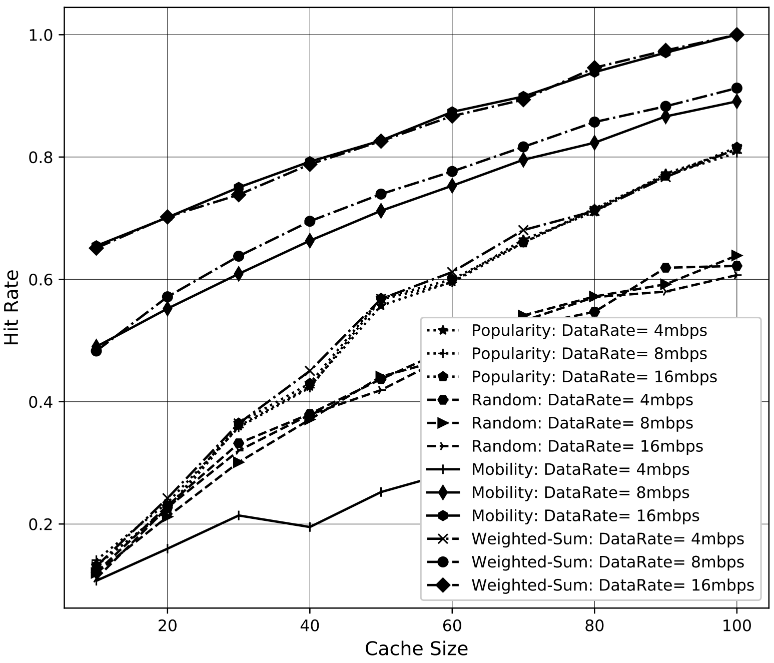


Figure 4.4: Impact of cache size (% of the total library size) on the hit rate with different values of SBSs data rate.

4.3.4 Impact of SBSs Communication Range:

Figure 4.5 presents the influence of SBS communication range varying from 75 to 200 m on the cache hit rate. As the results show, the cache hit rate increases in all algorithms when the communication range increases. As the SBS communication range increases, the number of SBSs within the communication range of the UT increases, so the possibility of finding the requested files within the caches of all the covered SBSs is higher. When the SBSs communication range is equal to 75m and 100m, the weighted-sum approach gives better results than mobility, popularity, and random caching algorithms. Also, the results show when the communication range is 200m, popularity caching provides a higher hit rate than random and mobility algorithms. In popularity caching during the cache placement phase, the popular files are placed in all SBSs and UTs within the wireless network cell. Thus, increasing the communication range covers more SBSs, which increases the possibility of finding the requested contents on one of the SBS or UT caches. Although popularity caching, in this case, results in higher performance, there is a penalty of losing energy while storing

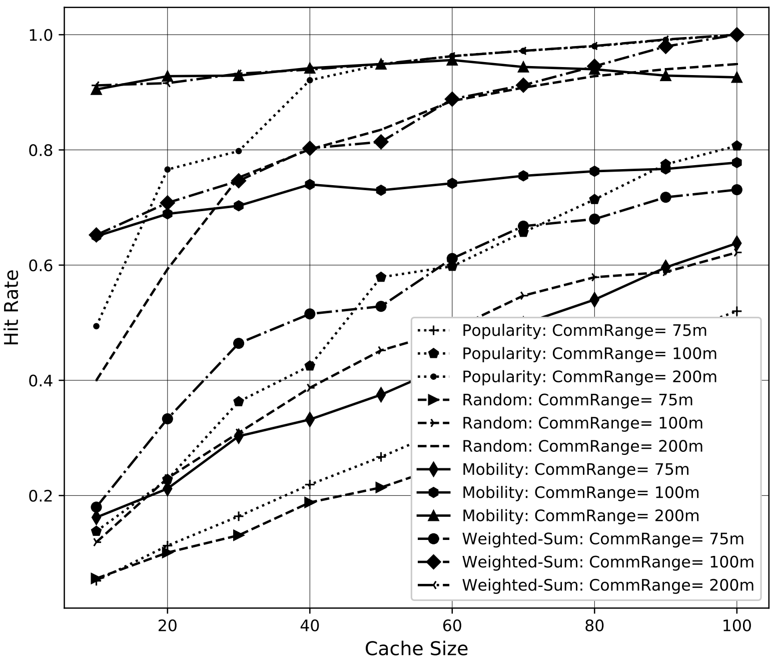


Figure 4.5: Impact of cache size (% of the total library size) on the hit rate with different values of SBSs communication range.

files in some SBSs that UTs will never reach (have low contact probability or outside communication range between SBS and UT). More studies are required to decide the trade-off between the energy consumption used to store files in SBSs and UTs and the obtained SBS hit rate.

4.3.5 Impact of User Terminal Speed:

We investigated the impact of user speed on caching strategy performance. We assume user speed can be 10, 20, or 30 m/s. Figure 4.6 illustrates the impact of changing user speed on cache hit rate for different values of cache size. It is observed that the weighted-sum approach outperforms popularity, random, and mobility caching for all user speeds. Also, mobility caching outperforms popularity and random caching at user speed 10 and 20 m/s. While at higher user speed (30 m/s), popularity caching gives a higher hit rate than random and mobility caching. This is because user mobility is considered while caching contents in SBSs along the paths in weighted-sum and mobility approaches. UT can download the contents from SBSs along the path while moving from

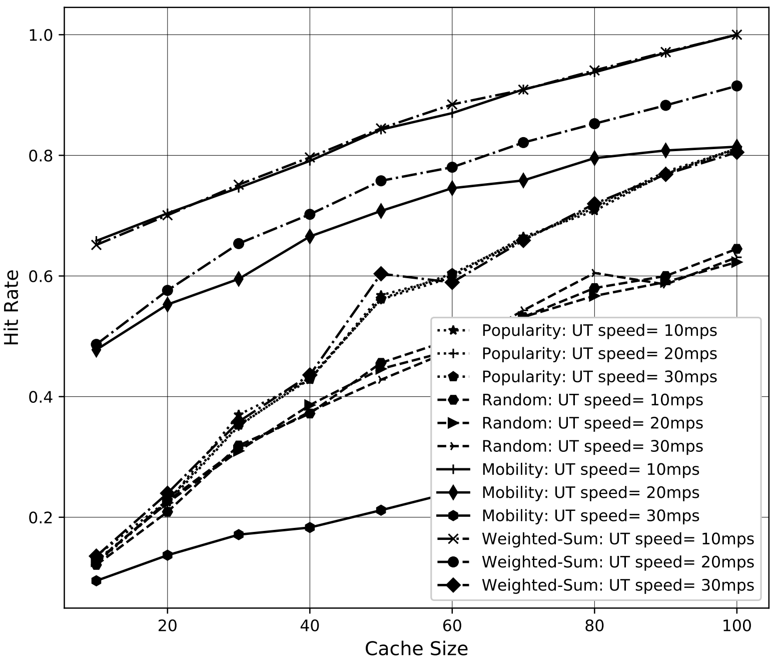


Figure 4.6: Impact of cache size (% of the total library size) on the hit rate with different values of user speed.

one location to another. When user speed increases, popularity caching outperforms mobility and random caching at all values of cache size. When user speed is higher (30 m/s), there is not enough time to download the content within the file deadline time. At higher user speed, the hit rate decreases and reaches zero for all strategies (not shown in the figure). This is because, at higher user speed, the available time to download the contents is less than the file download time. Users will not be able to download the contents from SBSs while moving along the paths. Contents will be downloaded from MBS.

4.3.6 Impact of SBS Normalized Energy Consumption:

For cache size equal 20% and 80% from the MBS librabry size, we investigate the impact of changing number of SBSs, SBSs data rate, SBSs communication range, and user terminal speed on the SBS normalized energy consumption for transmitting the contents from SBSs to UTs in the Figures 4.7, 4.8, 4.9, and 4.10, respectively.

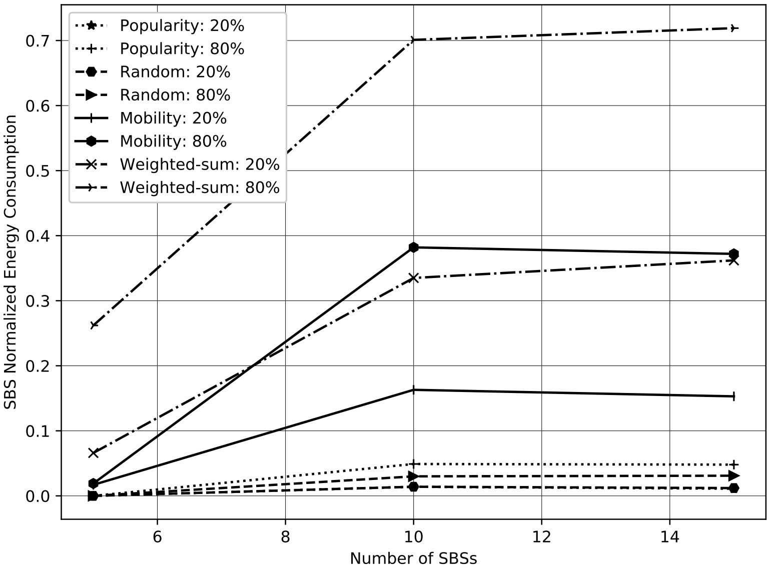


Figure 4.7: The impact of number of SBSs on the SBS normalized energy consumption.

As the figures show, the energy consumption in SBSs increases with the increase in the cache size. As the cache size increases, the cache hit rate increases which means contents are mostly downloaded from SBSs and UTs. Increasing energy consumption for transmitting the contents from SBSs to UTs means decreasing the energy consumption for downloading the exact contents from MBS. In popularity and random caching techniques, the increase of cache size does not result in a significant increase in the hit rate, which means contents are neither found in SBSs caches nor UT caches and are delivered to user terminals from MBS. While in mobility and weighted-sum techniques, the increase in cache size increases SBSs energy consumption. The increase depends on the impact factors (number of SBSs, SBSs data rate, SBSs communication range, and user terminal speed).

4.3.7 Impact of Weighted-Sum Threshold:

Figure 4.11 shows the impact of changing the threshold value from 0.25 to 1 on the cache hit rate. As expected, the hit rate increases when the threshold decreases. Setting a small threshold value means allowing any of the input attributes to contribute to content caching. This may result in caching contents even if only one of the input attributes has a high probability. For example, caching

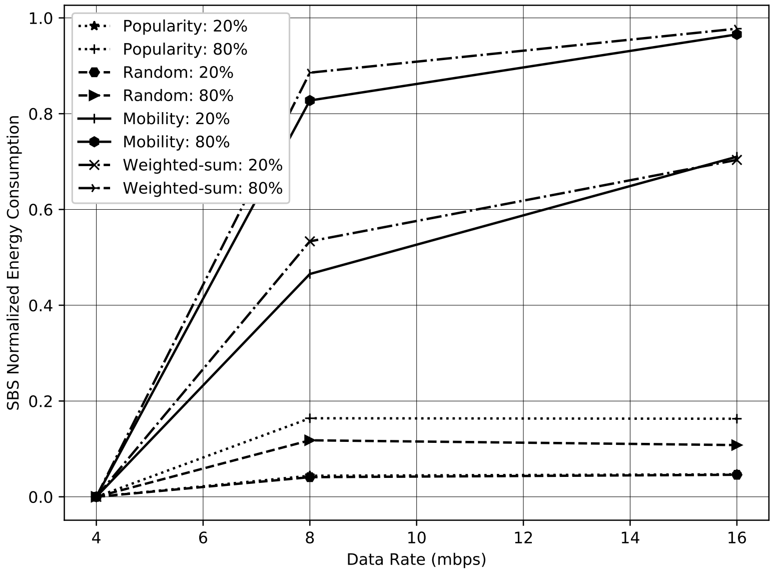


Figure 4.8: The impact of number of SBSs on the SBS normalized energy consumption.

popular content in SBS or UT cache may be out of the communication range of the UTs.

4.4 Summary

We proposed a new mobility-aware and latency-efficient cache placement algorithm for MENs. The algorithm simultaneously considers varying storage capacities, content popularity, contact probability, user mobility, contents-downloading latency, and energy consumption to transmit the contents to UTs. Its goal is to maximize the cache hit rate, which eventually minimizes the required content latency. We used an approach based on weighted-sum to decide to place contents at the edge of the network. Simulation results are used to analyze the impact of cache size, number of SBSs, SBSs data rate, SBS communication range, user terminal speed, SBS normalized energy consumption, and weighted-sum threshold on cache placement performance. Finally, we compare our proposed algorithm with respect to existing cache placement techniques. In the following chapter, we will propose a new solution to the cache placement problem by formulating it as a classification problem and solving it using the supervised learning technique.

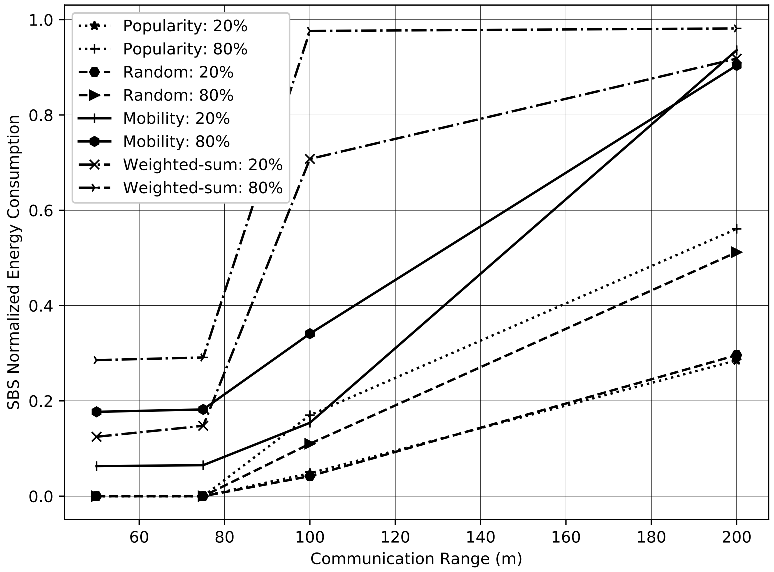


Figure 4.9: The impact of SBS communication range on the SBS normalized energy consumption.

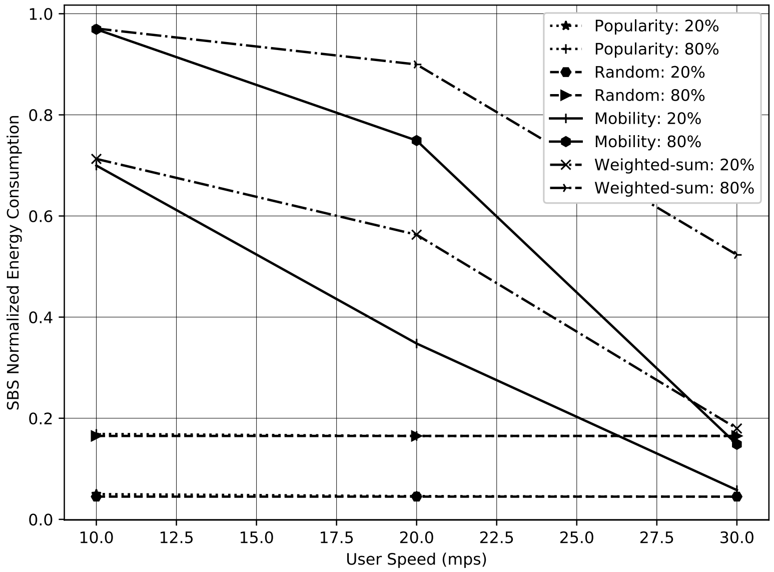


Figure 4.10: The impact of user terminal speed on the SBS normalized energy consumption.

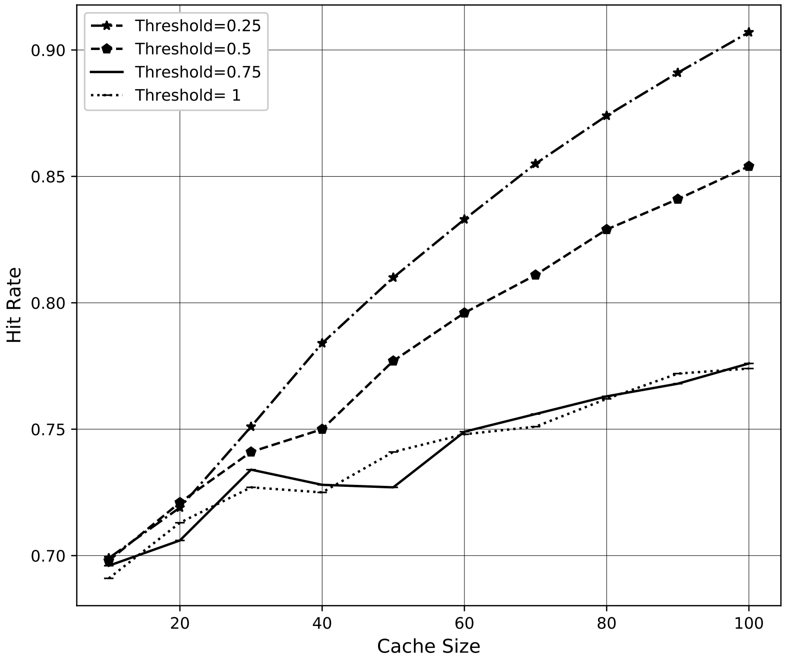


Figure 4.11: Impact of cache size (% of the total library size) on the hit rate with different values of threshold.

Chapter 5

Cache Placement Using Supervised Learning Techniques

In this chapter, a new formulation of the cache placement problem is investigated. This is based on our formulation in [211] presented in the last chapter of a latency efficient multi-objective function that aims to maximize the cache hit rate by maximizing the file content popularity, maximizing the contact duration between the mobile user and the cache that stores the required contents, minimizing the communication range between the mobile user and the cache, and maximizing the UT contact probability. In the formulation mentioned above and solution, we assumed equal weights for all input parameters. We focus on the values of weighted-sum (WS) cache placement input attributes to model the relationship between these values and the hit rate when the users are moving. Weights are assigned to each input to assess the contribution of input attributes on cache placement decisions.

5.1 Introduction

Various cache placement schemes have been designed for edge caching in wireless networks. However, most existing literature on cache content placement algorithms maximizes the cache hit rate by placing content based on content popularity changes. This is because a few popular files cause a large number of content requests. Using popular files, cache placement may result in replication of the same file in many caches in one cell. In [198], the authors presented a cache placement algorithm where the decision of cache or not cache (1 or 0) is converted into a convex optimization problem where the decision is relaxed to any value in [0,1]. Using an optimization technique, a non-integer solution is found and converted into an integer solution. In this algorithm, the authors took content popularity distribution as an input to the optimization problem and specified the problem's constraints based on the network topology.

A deterministic caching approach was used for cache placement based on computational greedy algorithms. In [199], an optimal greedy algorithm was proposed for cache content placement. The authors exploited the search for optimal contents by balancing between the content diversity in caches within the wireless network cell and the channel diversity gain. The authors in [200] designed the cache placement with the objective to minimize the delay of delivering the file to the end-users. They proved that the objective function was a monotone non-increasing super-modular function that a greedy algorithm could solve to obtain a locally optimal solution. Deterministic caching approaches can improve network performance only with the assumption that UTs stay at the exact locations for several hours [200].

To enlarge the set of cached files and exploit caching diversity, cooperative caching and device-to-device (D2D) caching approach was presented in [201], [202], and [203]. The D2D approach utilizes UT caches to improve network performance. The D2D cache placement algorithms considered content popularity, distances between users in one cell, and the social association between users [204]. An asymptotically optimized online learning content placement and content delivery algorithm was proposed in [204]. The algorithm used a distributed online learning approach based on stochastic gradient descent (SGD) at individual edge servers. Studies that use multi-winner auction theory for cache placement-based D2D communication showed that the network throughput increased linearly with the number of UTs in one cell [205].

A learning-based proactive caching was investigated to predict the content demand of users. In [206], the authors used singular value decomposition (SVD) to predict content preferences. In [207], the users-to-content association was used to predict content popularity using non-negative matrix factorization (NMF) machine learning. This is a linear model that learns through one layer relationship between the user and the contents. Multi-layer learning was proposed in [208] that used a deep neural network, which provides a nonlinear model for the association between users and the content demands. Distributed deep learning (DDL) was proposed in [209] in which deep learning was used to predict users' demand at the edge of the network. Then, the trained models were collected by the central server from the MENs, and the global model was updated accordingly.

The cache placement problem was studied from the perspective of user mobility. In [3], the authors presented the effects of exploiting the statistical traffic pattern and users' context information, such as file popularity, user location, and mobility pattern on the number of resources needed and at which network location the content should be pre-cached. Context-aware proactive caching was also considered in work proposed in [123]. The authors formulated a multi-armed bandit optimization problem. They considered user mobility by computing connected users' preferences and users' service priorities in a given time slot. The content popularity was considered to change over time by considering users' movements and connections between new users. In [137], the authors addressed realistic features of cache placement by describing user mobility via the set of highly visited locations within the cell where one or more SBSs covered each location. The impact of mobility-awareness in cache placement algorithm was discussed in [109]. The authors formulated the problem of caching of coded segments at base stations (BSs) and UTs taking into account user mobility and the content amount per transmission. User mobility was presented as a peer-to-peer connectivity model with contact duration and contact frequency perspectives. The problem was formulated as an integer programming problem and solved by sub-modular optimization.

In most previous works, such as that in [3], [123], [137], and [109], the impact of user mobility on caching strategy was considered from one of the following perspectives: either from users' content preferences, highly visited places by the users, user social context, content segmentation and multicasting of segments to multiple cache locations, or users contact probability. However, improving the quality of service by increasing the cache hit rate that minimizes the latency of downloading the contents and increases the cache hit rate that reduces the total energy consumption requires learning techniques to model cache placement. Learning techniques can exploit different user mobility attributes and the relationship between these attributes that can affect cache placement decisions. This is a challenging problem and motivates further investigations. The future wireless network should be more adaptive to user mobility, data rate, delay, number of SBSs and user terminals, and many other network information and parameters. To improve the adaptiveness to the dynamic changes in wireless networks, network intelligence should be used to model network status and update future predictions. This is an open research challenge requiring the design of edge caching strategies that can automatically adapt the wireless network changes [210].

The multi-objective cache placement optimization is formulated as a classification problem. The formulation of cache placement as a binary classification was investigated in [212]. The authors considered the input propriety of video contents and user context requesting the video contents are the input attributes that defined the space of cache decision. The input attributes included the number of views, likes, shares, comments, language, user subscription, and user past views. Unlike previous work, we explore mobility input attributes such as user locations, contact duration, communication ranges, contact probability between UTs and SBSs, content popularity, and the correlation between these input attributes that can separate the decision space into two regions cache and not to cache.

Unlike existing techniques, we propose a new cache placement algorithm formulated as a binary classification problem (to cache and not to cache) based on user locations, contact probability, communication range, contact duration, and content popularity. Artificial neural networks (ANN), support vector machine (SVM), and logistic regression (LR) are used to model cache placement decisions. We investigate the characteristics of the input features (attributes) and the properties of these characteristics that affect supervised machine learning approaches.

We investigate the performance of new cache placement models using supervised learning techniques by placing the proposed models in work with different system parameters. These parameters are varied to study the sensitivity of classification decisions with the change of system parameters.

5.1.1 Input Attributes Influence Weights

The cache placement decision is a binary decision (place or not to place the contents) that results in one of two cases: either the content is found ($Hit = 1$), or the content is not found ($Hit = 0$) in UT and SBS caches when the user requests the content. We will use the binary classification conditional probabilities presented in [213]. As mentioned before, in cache placement, we have four input attributes: file popularity probability (Λ_z), normalized contact duration ($\mathcal{T}^{*UT,SBS}$), normalized communication range $\mathcal{D}^{*UT,SBS}$, and contact probability $P_m^{UT,SBS}$. Each input attribute has its influence on the cache placement decision, which is represented by the weights. The combination of these influence weights gives an influence score for cache placement decisions.

To formulate the influence of weights, we assume the influence weight for the cache placement decision for each input attribute to be between -1 and $+1$. That means the certainty of the decision (place) produces a weight of $+1$, and the certainty of the decision (not place) produces a weight of -1. Thus, the influence weight of the input attribute value represents the influence weight for an input attribute on cache placement decision relative to its opposite. For any input attribute, the sum of the conditional probabilities for each possible result is 1 as the events are mutually exclusive, shown as follows:

$$Pr(Hit \mid I_i) + Pr(Miss \mid I_i) = 1, \tag{5.1}$$

where I_i is the value of the input attribute $\forall i \in \{1, \cdots, 4\}$. Consider that the cache hit is positive, and the probabilities can be mapped between the range of 0 to +1. Then, the cache miss is negative, and the probabilities can be mapped between the range -1 to 0. The range of mapped probabilities for any input attribute is between -1 to +1. Since each weight represents its influence on both probabilities, we can compute the influence weight for an input attribute for both cache states as follows:

$$w_i = Pr(Hit \mid I_i) + Pr(Miss \mid I_i). \tag{5.2}$$

Assume for a given period; there are $\mathcal{V}$ number of user requests. Each request results in a hit or a miss, then the hit rate (h) can be computed as follows:

$$h = \frac{\text{Total Number of Hits}}{\text{Total Number of Requests } (\mathcal{V})}. \tag{5.3}$$

To demonstrate the calculations for influence weights, assume $Pr(Hit \mid \Lambda_z) = 0.6$, $Pr(Hit \mid \mathcal{T}^{*UT,SBS}) = 0.8$,$Pr(Hit \mid \mathcal{D}^{*UT,SBS}) = 0.9$, and $Pr(Hit \mid P_m^{UT,SBS}) = 0.4$. Then, $Pr(Miss \mid \Lambda_z) = -0.4$, $Pr(Miss \mid \mathcal{T}^{*UT,SBS}) = -0.2$, $Pr(Miss \mid \mathcal{D}^{*UT,SBS}) = -0.1$, and $Pr(Miss \mid P_m^{UT,SBS}) = -0.6$ which are mapped between 0 and -1. According to (5.2), the file popularity probability influence weight (w_1), normalized contact duration influence weight (w_2), normalized communication range influence weight (w_3), and contact probability influence weight (w_4) are equal to $0.2, 0.6, 0.8,$ and -0.2, respectively. Figure 5.1 shows an example of input attribute influence weights on the two classes of the cache placement problem.

5.1.2 Classification Based Influence Scores

We can compute a single score for each user request for each influence weight of the input attributes. Suppose we calculate the summation and averaging of the input attribute influence weight w_i for all user requests for input attribute I_i. In that case, we can compute a score that can be used to indicate the influence weight of the input attribute. For our cache placement input attributes, the influence weight input attribute score is

$$\mathcal{S}_a = \frac{1}{\mathcal{V}} \sum_{j=1}^{\mathcal{V}} w_{a,j}, \tag{5.4}$$

where $a = \{1, 2, 3, 4\}$ and $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$ represent the influence weight input attribute scores for (Λ_z), $(\mathcal{T}^{*UT,SBS})$, $\mathcal{D}^{*UT,SBS}$, and $P_m^{UT,SBS}$, respectively. $w_{a,j}$ is the influence weight for the ath input attributes and jth user request.

According to (5.4), the influence score can be used as evidence that the corresponding example results in one class and not the other class. The samples that represent previous cache placement decisions for $\mathcal{V}$ user requests are considered as a training data set. From the training data sets, we can search for cases that result in incorrect classification and misclassification. The classification is the task of predicting the class to which the set of input attributes belongs. The classifier $k(I, w)$ aims to define a decision surface $k(I, w) = 0$ such that for a given set of input attributes I, the classifier can predict to which class the input attributes belong. Then, given the value of I, its class label y is predicted according to the following:

$$y\prime = k(I, w), \tag{5.5}$$

where $k(I, w)$ is used to decide on which side of the decision surface $k(I, w) = 0$ lies the input attributes I [214]. For a given training set with $\mathcal{V}$ samples that result in either $y = -1$ in a miss or

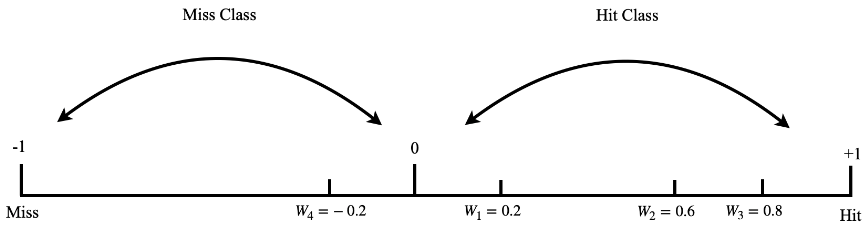


Figure 5.1: Example of input attribute influence weights of two classes cache placement.

$y = 1$ in a hit, the goal is to design a hyperplane that separates these two classes.

$$\begin{aligned} k(I, w) &= f(I, w) \\ &= f(I, w_0, w_1, w_2, w_3, w_4) = 0, \end{aligned} \tag{5.6}$$

where w_0 is the bias and f is the classifier function, which can be either linear or nonlinear. We need to build a classifier that can place points with negative score values on one side of the hyperplane and place points with positive score values on the other side of the hyperplane. Figure 5.2 shows an example scenario where the hyperplane is separating the 4-dimensional space into two regions. For each row in the training set, we have input attributes and the value of the label y. Label y is the class that this row belongs to, so it is either $y = -1$ or $y = +1$. The influence weights w_i for $i = 1, \cdots, 4$ of input attributes and the value of the bias w_0 can be computed by minimizing the mean square error (MSE) cost function $J(w_j)$ as follows [215]

$$\text{minimize} \quad J(w) = \frac{1}{\mathcal{V}} \sum_{j=1}^{\mathcal{V}} (y_j - y'_j)^2, \tag{5.7}$$

where y' is the predicted class for the given values of influence weights and input attributes, and y is the actual class value for the jth sample. The MSE should be minimized such that the predicted responses would be as close as possible to the actual values of the class. The weights and bias are estimated to formulate the optimal model for the classifier. A learning technique is presented in the following section that can estimate the values of influence weights and build the cache placement model.

5.2 Hyperplane Parameters Estimation based on Learning Techniques

The hyperplane parameter estimation can be implemented using supervised machine learning classifiers. A supervised classifier requires a dataset consisting of input/output pairs. Each row in the data set represents the observations (input), or what is referred to as the vector of features and the corresponding class y. The classification function aims to build a model by training the given dataset, and then the model will be able to estimate the output $y\prime$ for new input features. The classifier learns from the previous datasets to predict the output when given new input features. The classifier's output is either placing the content (cache) or not placing the contents (not cache). The objective function of the learning algorithm is to minimize the cost function given in (5.7). The algorithm used for optimizing the objective function is the stochastic gradient descent (SGD) algorithm, which learns statistics iteratively from the training dataset.

As presented in the following subsections, three supervised machine learning (ML) techniques are used as binary classifiers for the cache content placement algorithm. We have adopted the SGD algorithm in the three learning techniques because, in the literature, it has been shown that SGD can minimize the loss function evaluated over a given training set and find the right set for the estimated hyperparameters. Furthermore, it can avoid the overfitting problem by stopping the iterations of the optimization routine early before it converges [216].

Consider the training data set, also called the observations, that consists of n input features $(I_1, I_2, \cdots, I_n)$ and the corresponding class y for $\mathcal{V}$ number of observations. The classifier output y can be 1 that means the observation belongs to the class cache or 0, which means the observation belongs to the class, not cache.

5.2.1 Artificial Neural Network (ANN) Cache Placement

Artificial Neural Network (ANN) is first adapted for binary classification of cache placement problems. The adaptive momentum (Adam) gradient descent-based optimizer is used for estimating the cache content. The Adam algorithm is a combination of RMSprop and SGD with momentum. It uses the square gradient to scale the learning and uses the moving average of the gradient instead of the gradient with momentum [217]. The Adam algorithm is utilized in machine learning problems with high-dimensional input parameter space and massive datasets. The algorithm calculates the adaptive learning rate individually for various parameters involved in the training of gradients [217].

We consider an ANN with three layers: one input layer, one hidden layer, and one output layer. The input layer is the first layer of the ANN, and it does not have any weights associated with it. The input layer consists of a set of n neurons representing the number of input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation followed by a sigmoid nonlinear activation function. The output layer receives the output from the hidden layer and computes the output value y. Thus, the ANN estimates a nonlinear classifier that separates the two classes. This classifier can be written as $k(I, w) = f_{\Omega}(I, w)$, where $f_{\Omega}(I, w)$ is a nonlinear function consisting of the weights of the neurons in the hidden and output layer, as well as the activation function.

$$\Delta\Omega_t = -\alpha \frac{m'_t}{\sqrt{v'_t + \epsilon}} \tag{5.8}$$

$$\Omega_{t+1} = \Omega_t + \Delta\Omega_t \tag{5.9}$$

- m the exponential average of gradient along Ω.
- v the exponential average of squares of gradient along w.
- m'_t the bias-corrected first moment estimate.
- v'_t the bias-corrected second raw moment estimate.
- Ω_t the model weights at time t.
- Ω_{t+1} the updated model weights at time $t + 1$.
- α the initial learning rate.
- ϵ very small number to prevent any division by zero in the implementation, assume $\epsilon = 10^{-8}$.

The algorithm of artificial neural network for cache placement is shown in Algorithm 2.

5.2.2 Support Vector Machine (SVM) for Cache Placement

Support vector machine (SVM) is a learning technique that defines a hyperplane to split the attributes space (input features) into two classes. Given a training dataset $(\{(I_i, y_i)\}_{i=1}^{\mathcal{V}}$ with vectors $I_i \in \mathbb{R}$ and labels $y_i \in \{+1, -1\}$, the aim is to design a linear classier $k(I, w)$ that satisfies the following [218]:

Algorithm 2 : Artificial Neural Network Classifier for Cache Placement

```
    Input: Dataset with input features $I_j$ and one output label $y$.
    Output: Classifier model $k(I, w)$ given in (5.5).
 1: Initialize bias and weights $\Omega$=random value $\in [0, 1]$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, moving averages $m_{t-1} = v_{t-1} = 0$, $\epsilon = 10^{-8}$ and learning rate $\alpha = 0.001$.
 2: Split dataset into 70% for training and 30% for testing.
 3: Construct artificial neural network.
    Training the model:
 4:     For $i = 1$ : Number of epochs
 5:         $\eth_t = \nabla_\Omega J(\Omega_{t-1})$
 6:         $m_t = \beta_1 \times m_{t-1} + 1 + (1 - \beta_1) \times \eth_t$
 7:         $v_t = \beta_2 \times v_{t-1} + 1 + (1 - \beta_2) \times \eth_t^2$
 8:         $m'_t = \frac{m_t}{(1-\beta_1)}$
 9:         $v'_t = \frac{v_t}{(1-\beta_2)}$
10:         $\Omega_{t+1} = \Omega_t - \alpha \frac{m'_t}{\sqrt{v'_t + \epsilon}}$
11:     Endfor
12: Return model structure with updated bias and weights $\Omega_j$.
    Testing the model:
13:     For each instance in the testing dataset
14:         Read input attributes $I_j$
15:         Compute $y'$ using the resulting model from step 12.
16:     Endfor
    Output: Cache strategy matrix A
```

$$\begin{gathered} w^T I_i + w_0 \geq +1, \text{ if } y_i = +1, \\ \text{and} \\ w^T I_i + w_0 \leq -1 \text{ if } y_i = -1. \end{gathered} \tag{5.10}$$

This is equivalent to

$$y_i[w^T \phi(I_i) + w_0] \geq 1, \text{for } i = 1, \cdots, \mathcal{V}. \tag{5.11}$$

The nonlinear function $\phi(.)$ maps the input space to a high dimensional feature space. This means the SVM classifier objective function can be defined as [218]

$$\min_w J(w_j) = \frac{1}{2} w^T w + C \sum_{i=1}^{\mathcal{V} \max(0,1-y_i w^T I)}, (5.12)$$

where $\min \frac{1}{2} w^T w$ is the regularization term that maximize the margin and imposes a preference over the hypothesis space to achieve better generalization. The term $\sum_{i=1}^{\mathcal{V} \max(0,1-y_i w^T I)}$ is called the hinge loss (empirical risk loss) and is used to penalize weight vectors that make mistakes. The penalty parameter $C (C > 0)$ is used to control the trade-off between a large margin and a small hinge-loss.

As shown in (5.12), if the sign of the product between the true label y and the predicted value $y' = w^T I$ is positive and larger than 1, the loss is zero. If not, the loss will increase linearly. The stochastic gradient descent algorithm for SVM performs gradient descent for the objective function given in (5.12). The updating rule for the weights is then [219]

$$w_{t+1} = w_t - \alpha \nabla_{w_t} \left[\frac{1}{2} w^T w + C \sum_{i=1}^{\mathcal{V} \max(0,1-y_i w^T I)} \right], (5.13)$$

where α is the learning rate and ∇_{w_t} stands for a subgradient with respect to w_t. The algorithm of SVM for cache placement is shown in Algorithm 3.

5.2.3 Logistic Regression (LR) for Cache Placement

The binary logistic regression (LR) algorithm trains a classifier to make a binary decision about a class of new input observations. The LR model estimates the probability $P_r(y = 1 \mid I)$ that this observations belongs to the class cache or the probability $P_r(y = 0 \mid I)$ that it belongs to the class

Algorithm 3 : Support Vector Machine Classifier for Cache Placement

```
    Input: Dataset with input features $I_j$ and one output label $y$.
    Output: Classifier model $k(I, w)$ given in (5.5).
 1: Initialize bias $w_0$=0, weights $w_j = 0$, and learning rate $\alpha = 0.001$.
 2: Split dataset into 70% for training and 30% for testing.
    Training the model:
 3: For $i = 1$: Number of epochs
 4:         Pick a random sample $(I_i, y_i)$ from the training dataset.
 5:         Compute $\nabla_{w_t}$ of $J(w)$ (5.12)
                    $w_{t+1} = w_t - \alpha \nabla_{w_t} J(w_t)$
 6: Endfor
 7: Return model structure with updated bias $w_0$ and weights $w_j$.
    Testing the model:
 8: For each instance in the testing dataset
 9:         Read input attributes $I_j$
10:         Compute $y'$ using the resulting model from step 10.
11: Endfor
    Output: Cache strategy matrix A
```

not cache. The LR learns from the training set by updating the weights $w_1, w_2, \cdots, w_4$ and the bias w_0. As discussed in section 5.1.1, the weights indicate the importance of each input feature to the classification problem. The LR learns by updating the values of the weights and bias from a training set. Each weight w_i is associated with one input attribute I_i. The weight represents the importance of the input attribute to the classification decision. The bias terms are added to the weighted inputs.

According to Bayesian statistics, the posterior probability of a random event or uncertain proposition is the conditional probability assigned after the relevant evidence or the background is taken into account. Let us consider logistic regression model for a two-class (y_1, y_2) classification task, the posterior probabilities are modelled as [220]

$$P_r(y_1 \mid I) = \frac{1}{1 + \exp(-Iw)} \tag{5.14}$$

and

$$P_r(y_2 \mid I) = 1 - P(y_1 \mid I) \tag{5.15}$$

After complete training of the weights and bias, the classifier multiplies each input I_j by its updated weight w_j, sums up the weighted features, and adds the updated bias term w_0 as shown in Algorithm 5.

The classifier models described by Algorithms 2, 3, and 5 define the contents of SBS caches. This means these classifiers define the contents of the solution matrix A presented in the multi-objective function given in (4.1).

5.3 Experimental Results

In this section, we evaluate the performance of the classifier-based supervised machine learning algorithms for cache placement and access via simulations. In addition, we compared the performance of the proposed cache placement approaches with popularity, random, weighted sum cache placement algorithms.

5.3.1 Simulation Setup

Our simulation is implemented on a certain sub-area with area dimension and number of SBSs as described in [197]. We assume that the macrocell includes one MBS, 15 SBSs, and there are three different paths that users pass through with 11 locations in each path. The results are computed for $10,000$ randomly generated user locations and $10,000$ different user requests within one wireless network cell. Each request consists of the requested file ID. The requests of files follow Zipf's popularity distribution. A number of experiments are implemented to compare the impact of system parameters on the performance of different cache placement algorithms. During each experiment, one of the parameters is varied, and the remaining are fixed. The parameters are illustrated in Table 4.1. For the weighted-sum approach, we assumed all the parameters have equal weights, and the threshold is 0.5 to take the 50% of the effect of the input attributes.

We assume that the user mobility parameters follow the work in [109]. It is defined as an independent Poisson process, the pairwise contact duration between mobile UT and SBS follows the exponential distribution with parameter $\lambda_{i,j}^{SBS}$, such that $\Gamma(10, 1/100)$ represents the contact rate between mobile UT and SBS. Figure 5.3 shows the first few lines of the training dataset, which will be used to build the cache placement model. Each row represents one sample data and the corresponding cache placement decisions. The first five columns represent the input attributes ”features” of the cache placement model. The last 15 columns represent the corresponding cache placement decisions. Feature descriptions are as follows:

1. File popularity: File popularity probability as given in 3.1

2. Contact probability: Contact probability between mobile UT and SBS as discussed in section 3.2.3.

3. Location: User location can be in one of 11 possible locations.

4. Path number: User can use one of three possible paths.

5. Contact duration: The contact duration between mobile UT and SBS as given in 3.2.

The Last columns (SBS0-SBS14) represent the cache placement decisions for the SBSs 1-15, respectively. The algorithms are implemented using Python 3 in the Anaconda environment.

The dataset is divided into 70% data used for training, and 30% data is used for testing. The simulation was implemented using Python on a Jupyter notebook installed on Anaconda environment [221].

5.3.2 Cache Placement Algorithms

To evaluate the performance of the proposed classifier-based cache placement algorithm and investigate the relationship among various input attributes, we compare the impact of different variables on six caching techniques. The cache placement techniques are:

5.3.2.1 Popularity Caching

It assumes that the most popular contents are placed in each SBS cache [109].

5.3.2.2 Random Caching

It assumes that contents are placed randomly in each SBS cache.

5.3.2.3 Weighted-Sum Approach Caching

In our earlier work, we assumed that the contents are placed in SBS caches depending on input attributes such as file popularity, contact duration between UT and SBSs, communication ranges between UT and SBSs, and contact probability between UTs and SBSs [211].

5.3.2.4 Artificial Neural Network with SGD Learning

An ANN is trained on datasets of previous user input attributes, user requests, and the resulted in hit or miss, to extract information from input attributes and predict cache contents.

5.3.2.5 Logistic Regression (LR) with SGD Learning

It uses an optimization algorithm to train the cache placement model by minimizing the loss function in (5.7). The trained model can be used to predict cache contents.

5.3.2.6 Support Vector Machine (SVM) with SGD Learning

It uses SVM to find the hyperplane separating the decision space into two classes place or not place the contents that result in a cache hit or cache miss. The iterative learning process is implemented using the SGD algorithm that aims to minimize the cost function in (5.7).

5.3.3 Learning Technique Classification Models

In this section, the results are evaluated for various learning techniques used in this chapter: ANN, SVM, and LR. The performance of the classification models is explored for the same dataset using a confusion matrix that describes the classification model performance. It examines the testing dataset to find if the classification outcome is correct or not. Finally, the model accuracy scores are calculated to identify model errors [222].

Figures 5.4-5.6 show the confusion matrices for ANN, SVM, and LR classification models, respectively. These figures summarize the percentage of correct and incorrect predictions for each class and the types of errors made by the classifiers. The correct estimation is organized along the diagonal direction from top-left to bottom-right of the matrices. There are 23.13%, 21.27%, and 23.09% of the sample cases that belong to the not cached category, and they are predicted not cached by the ANN, SVM, and LR models, respectively. These samples are true negatives (TN), which means they are predicted not to cache, and the correct decision is not to cache. The other correct estimation is the true positive (TP) case, there are 75.35%, 74.64%, and 74.42% of the samples, that are cases in which the actual decision was to cache and they are predicted to be cached by the ANN, SVM, and LR models, respectively. This refers to the cases where users requested the files and the files were stored in the caches.

By looking at the classification errors, false positive (FP), gives 0.0%, 0.04%, and 1.86% for the ANN, SVM, and LR models, respectively. This means some percentage of samples in SVM and LR models are identified as cache and should not be cached. From the cache placement perspective, this can cause energy loss and more latency to store contents in some SBS caches that will not be requested from these SBSs and/or will not result in successful transmission to the requested UTs. From this point, ANN can be considered the best model for estimating cache contents compared to other cache placements algorithms used in this work.

Table 5.1 describes in more depth the performance of our estimators by applying the proposed ANN, SVM, and LR algorithms on the datasets and comparing the accuracy, precision, Recall, and F1 score of the results. The accuracy gives the ratio of correctly predicted observations to the total observations. Accuracy is defined as (TP+TN)/(TP+FP+FN+TN). Accuracy alone will not indicate the performance of the estimator. Analyzing other performance metrics will provide us with more understanding of our model before using it in the actual environment with different cases. The precision indicates how precise our model is out of those predicted positive (cache), how many of these samples are positive. Hence, Precision= TP/(TP+FP) and the sensitivity or recall performance index measures the ratio of correctly predicted positive observation (cache) to all observations in the actual values. That is, Recall = TP/(TP+FN)

Then, we compute the weighted average of precision and recall to compute the F1 score performance index. F1 score helps to understand the model accuracy better than the accuracy performance index when we have uneven class distribution, where F1 score is defined as

$$\text{F1 Score} = \frac{2 \times (\text{Recall} \times \text{Precision})}{(\text{Recall} + \text{Precision})} \tag{5.16}$$

Based on the results provided by Table 5.1, among the three classification models, ANN has performed best and has achieved the highest score as compared to the other algorithms. All the remaining algorithms appear to demonstrate a good performance compared to our previous work weighted-sum approach.

Table 5.1: Performance evaluation for classification models.

Classification	**Classification Models**			
Performance	**ANN**	**SVM**	**LR**	**Best Score**
Accuracy	0.98	0.79	0.81	ANN
Precision	0.99	0.98	0.96	ANN
Recall	0.98	0.98	0.96	ANN, SVM
F1 Score	0.99	0.98	0.96	ANN

5.3.4 Performance Evaluation

In this chapter, we show only the results of caching in SBS caches. The six strategies are evaluated and compared by computing the cache hit rate for different experiments. User mobility is considered by considering that users may request files while moving from one location to another location within the wireless network. Users can download files from nearby SBSs without the need to use a backhaul link. This will cause a cache hit. The cache hit rate indicates the number of cache hits divided by the total number of user requests.

5.3.4.1 Impact of Cache Size

To study the impact of SBS cache size on the cache hit rate, we consider SBS cache size ranges from 10% to 100% of the total library size. Figure 5.7 shows the results of changing SBS cache sizes on the SBS hit rate. In all the cache placement algorithms, increasing the cache size increases the cache hit rate. This is because increasing cache size means more files can be cached in SBS caches, which increases the possibility of finding the requested contents. In popularity and random caching, the cache hit rate increases almost linearly with the increase of SBS cache size. On the other hand, the WS performs better than popularity and random caching with a near-linear increase in the cache hit rate with the increase of SBS cache size.

The performance of the learning techniques ANN, SVM, and LR are very similar and better than the other techniques. Their results show that although increasing cache size increases the cache hit rate, the hit rate is 0.92 in ANN, 0.921 in SVM, and 0.91 in LR even for 10% SBS cache size. This means that the trained models in the three learning techniques could extract information from the dataset and correct the cache content placement regardless of the cache size.

5.3.4.2 Impact of User Speed

Figure 5.8 shows the impact of SBS cache size on hit rate for different user speeds. The figure shows the results for 10 m/sec and 25 m/sec for four cache placement techniques: WS, ANN, SVM, and LR. We observed that at low user speed (10 m/sec), the SVM caching technique outperforms ANN, LR, and WS. However, when increasing user speed up to 25 m/sec, the ANN gives a higher hit rate than the SVM, LR, and WS.

5.3.4.3 Impact of SBSs Communication Range

We explore the impact of changing the SBS communication range from 75m to 200m on cache placement performances shown in Fig. 5.9 as the results show, when the SBSs communication range increase, the SBSs cache hit rates increase in all the cache placement algorithms. As the SBS communication range increases, the number of SBSs within the communication range of the moving UT increases. Thus, the possibility of finding the requested contents within the SBSs caches is higher. At a lower SBS communication range (75m), the LR approach gives a higher hit rate than ANN, SVM, and WS caching techniques. After increasing the SBS communication range up to 200m, the ANN outperforms SVM, LR, and WS.

5.3.4.4 Impact of SBS Data Rate

Figure 5.10 presents the influence of changing SBS data rate on the cache hit rate. As the results show, increasing the SBS data rate increases the hit rate for all cache placement techniques since SBSs can transfer more data during contact with the UT. Thus, increasing the SBS data rate allows the user to complete downloading the contents within available time while moving from one location to another location. When the SBS data rate is 4Mbps, the hit rate in the SVM technique outperforms the ANN, LR, and WS caching algorithms. While for higher data rates (16 Mbps), the results of LR show a higher hit rate than the ANN, SVM, and WS.

5.3.4.5 Impact of File Size

Figure 5.11 shows the impact of changing the file size from 1MB to 8MB on the SBS cache hit rate. As expected, the hit rate decreases when the file size increases. This is because the users are downloading the files while moving from one location to the next along the path. When the file size increases, the users will not complete downloading the whole file during the required time, which results in a miss. When the file size is small (1MB), the ANN gives a higher hit rate than the SVM, LR, and WS caching techniques. After increasing the file size up to 8MB, the ANN again outperforms SVM, LR, and WS.

Table 5.2: Comparison of cache hit rate for different cache placement techniques.

System Parameters	**Cache Hit Rate**					
		ANN	**SVM**	**LR**	**WS**	**Best Score**
SBS Cache Size	Low	0.92	0.921	0.91	0.70	SVM
	High	0.994	0.988	0.983	0.91	ANN
User Speed	Low	0.964	0.962	0.963	0.837	ANN
	High	0.79	0.887	0.93	0.565	LR
Communication Range	Low	0.786	0.923	0.965	0.695	LR
	High	0.964	0.963	0.962	0.872	ANN
SBS Data Rate	Low	0.795	0.923	0.88	0.41	SVM
	High	0.964	0.962	0.966	0.84	LR
File Size	Low	0.964	0.962	0.963	0.836	ANN
	High	0.795	0.696	0.704	0.465	ANN

Table 5.2 compares cache hit rate for four cache placement algorithms while changing some system parameters between low and high values. The results show that changing the SBS cache size from low to a high slightly increases the cache hit rates for ANN, SVM, and LR models, and the ANN results in the highest score. The WS cache placement algorithm is improved when we increase the cache size. We trained the model to place the contents based on the input features by defining the two classes' hyperplane parameters. On the other hand, in WS, all the input attributes are assigned equal weights in the cache placement model.

Changing user speed from low to high impacts all algorithms. Knowing that in our simulation, the speed is only increased to the point that the UT can still download the contents from SBS while the user is on the move. The LR is the least sensitive to the change of user speed and results in a higher hit rate. For the SBS data rate, changing the rate from low to high values affects the cache hit rates for all algorithms since the amount of contents transmitted to the UT increases with a higher data rate. The SVM shows the least sensitivity to the changes in data rate, and the difference between low and high data rates is marginal. In the last experiment, we changed the stored file size and computed the effect of this parameter on the cache placement algorithms. In this case, the ANN has the highest hit rate compared to the remaining techniques, as shown in Fig. 5.11.

5.3.4.6 Impact on SBS and MBS Energy Consumption

Figure 5.12 shows the impact of changing cache size and cache placement technique on the normalized total energy consumption in SBSs and MBS required to download the files. As the figure shows, the normalized total energy consumption decreases with the increase in the cache size. As the SBSs cache size increases, the cache hit rate increases, which means the contents are mostly downloaded from the SBSs. The increase in energy consumption for transmitting the contents from SBSs to UTs means decreasing the energy consumption for downloading the exact contents from the MBS. When comparing the normalized total energy consumption required to transfer the contents from SBSs and MBS to UTs between different cache placement algorithms, we can see that the SVM and LR result in lower energy consumption. On the other hand, the WS technique results in higher total energy consumption than the learning techniques.

5.4 Discussion

In this study, we proposed a new cache placement using supervised learning techniques in MENs. To the best of our knowledge, this is the first work where cache placement is formulated as a binary classification based on a multi-objective optimization problem to improve the cache hit rates, which minimizes the total latency of downloading the contents to the end-users. Also, the proposed cache placement-based learning techniques minimize the total energy consumption required to download the contents. We analyzed three proposed learning techniques for predicting cache contents, ANN, SVM, and LR. The first part of the experiments measures the accuracy and prediction precision among these three algorithms. The three algorithms show competitive performance in their computed precision, Recall, and F1 score for the same datasets. The accuracy of prediction is higher in ANN compared to SVM and LR. There is no unique classifier that performs best for all situations of cache placement systems in MENs. To select one of the three algorithms to model the cache placement automatically for a given dataset, a number of input features, and computation resources, we should consider the following:

- If the collected dataset is not large, SVM or LR should be selected. Otherwise, ANN is selected.

- If the input features increase, consequently increasing the hyperplane dimension, ANN or SVM should be selected. Otherwise, LR can be selected.

- If the computation resources are limited, LR is selected, which can be implemented faster and easier. Otherwise, ANN or SVM is selected.

Further selection criteria can be considered after analyzing the results from the second part of the experiments that measure the cache hit rate in MENs while changing the system parameters. Table 5.2 can assist in the selection of an appropriate learning algorithm based on the changes in SBS cache size, user speed, SBS communication ranges, and file size, as follows:

- If the SBS cache size is small, SVM should be selected. Otherwise, ANN should be selected.

- If it is known that most users are moving at a low speed, ANN should be used. Otherwise, LR should be used at a higher user speed.

- If the SBS communication ranges are low, LR should be used. Otherwise, ANN should be used for high SBS communication ranges.

- If the SBS data rates are low, SVM should be used. Otherwise, LR should be used.

- If the file size is large, ANN should be used.

Continuous dataset accumulation is suggested to more understand the effect of classification decisions and the change in system parameters on the latency and the total consumption of the energy required to download the contents in mobile edge networks. Increasing the orders of magnitude of training datasets requires adopting new technology such as big data and developing the proposed algorithms to deep learning-based prediction algorithms. However, there is an implication from using supervised learning techniques in some practical applications, collecting datasets from moving users, and creating such a large dataset. At the same time, continuous changes in system parameters require a considerable amount of resources, time, and effort. These resources may not be available in many practical cases, limiting the use of deep learning methods. Also, increasing the training samples may cause the supervised learning technique to fit itself to the statistical noise in the training sample and result in an overfitting problem. To overcome these problems, using semi-supervised learning leverages both labeled and unlabelled data for learning.

This work focused on formulating the cache placement as a classification problem that can be solved using machine learning techniques. Different learning techniques were investigated to understand the problem and the input attributes correlated to the classification decisions. We analyzed the characteristics of the input features (attributes) related to the decision of cache content placement objectives and their properties. These input attributes were important to form the hyperplane that separated the multi-dimensional space into two decision regions (cache contents and not cache contents). The performance comparison of different machine learning models was carried out with the same datasets. Finally, the cache placement models using artificial neural networks, support vector machine, logistic regression, and weighted-sum algorithms were tested with actual data sets of user requests taken from published literature. The test was made by changing one of the system parameters, fixing the other parameters, and computing the hit rate to investigate the sensitivity of the classification by the changes in the environmental parameters.

It has been noticed that with the changes in system parameters, there were a few limitations and weaknesses in each classifier model used in this research. Therefore, generating an automatic algorithm selection that decides which algorithm is executed in each wireless network cell depending on the size of the dataset, the number of input features, the available resources, and values of system parameters is suggested for future work. Furthermore, the automatic selection of the algorithm can be a rule-based technique aiming to combine different learning algorithms.

This work yielded promising results for the formulation of cache placement as a classification problem. Still, continuous dataset accumulation and then adoption of semi-supervised deep learning techniques are also suggested to understand the effect of classification and system parameter changes on the latency and energy consumption caused by downloading the contents by requested users.

5.5 Summary

We proposed a latency-efficient multi-objective cache content strategy that maximizes the cache hit rate of SBSs in mobile edge networks (MENs). The multi-objective cache placement optimization was formulated as a classification problem. Unlike previous work, we used mobility input attributes such as user locations, contact duration, communication ranges, contact probability between UTs and SBSs, etc., and content popularity and the correlation between these input attributes separating the decision space into two regions cache and not cache. We used a stochastic gradient descent algorithm for the training of three supervised machine learning techniques: artificial neural network (ANN), support vector machine (SVM), and logistic regression (LR) to define the hyperplane that separates the cache content decision space. We tested our proposed cache placement models with actual data sets of user requests taken from published literature. The test was made by changing one of the system parameters, fixing the other parameters, and computing the hit rate to investigate the sensitivity of the classification by the changes in the environmental parameters. Finally, we compared our proposed algorithms with respect to existing cache placement techniques. In the following chapter, based on the results obtained from our proposed cache placement algorithm using the supervised learning technique, we will present a new approach using a semi-supervised self-training algorithm to solve the cache placement problem.

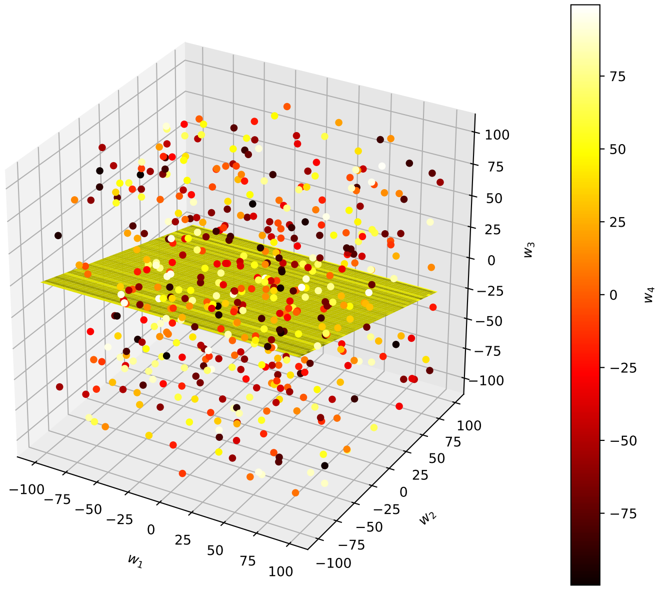
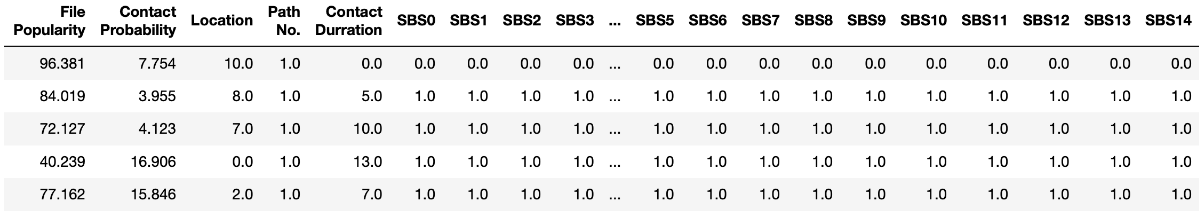


Figure 5.2: Hyperplane separating space into two classes.



File Popularity	Contact Probability	Location	Path No.	Contact Durration	SBS0	SBS1	SBS2	SBS3	...	SBS5	SBS6	SBS7	SBS8	SBS9	SBS10	SBS11	SBS12	SBS13	SBS14
96.381	7.754	10.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
84.019	3.955	8.0	1.0	5.0	1.0	1.0	1.0	1.0	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
72.127	4.123	7.0	1.0	10.0	1.0	1.0	1.0	1.0	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
40.239	16.906	0.0	1.0	13.0	1.0	1.0	1.0	1.0	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
77.162	15.846	2.0	1.0	7.0	1.0	1.0	1.0	1.0	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Figure 5.3: The input features and corresponding cache placement derisions.

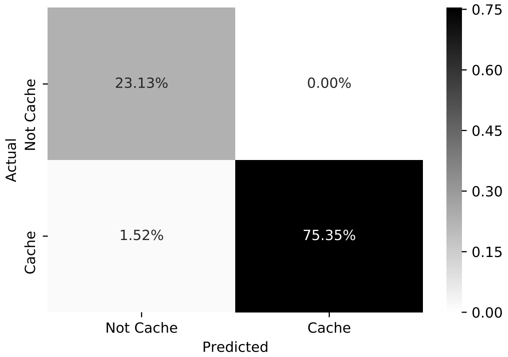


Figure 5.4: Confusion matrix of artificial neural network model.

Algorithm 4 : Logistic Regression Classifier for Cache Placement

```
Input: Dataset with input features $I_j$ and one output label $y$.
Output: Classifier model $k(I, w)$ given in (5.5).
Initialize bias $w_0$=0, weights $w_j = 0$, and learning rate $\alpha = 0.3$.
Split dataset into 70% for training and 30% for testing.
Training the model:
For $i = 1$: Number of epochs
    For each training instance in training dataset
        Compute $y' = P_r(y \mid I) = \frac{1}{1+\exp(-I_j w_j)}$
        Update bias and weights
            $w_0^{t+1} = w_0^t + \alpha(y - y')y'(1 - y')$
            $w_j^{t+1} = w_j^t + \alpha(y - y')y'(1 - y')I_j$
    Endfor
Endfor
Return model structure with updated bias and weights.
Testing the model:
For each instance in the testing dataset
        Read input attributes $I_j$
        Compute $y' = \frac{1}{1+\exp(-I_j w_j)}$
        If    $y' \geq 0.5$
            The decision is cache
        Elseif   $y' < 0.5$
            The decision is not cache
Endfor
Output: Cache strategy matrix A
```

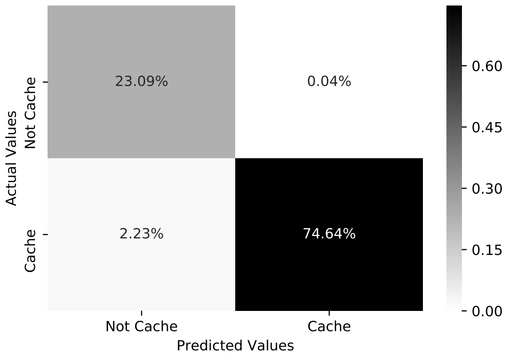


Figure 5.5: Confusion matrix of support vector machine model.

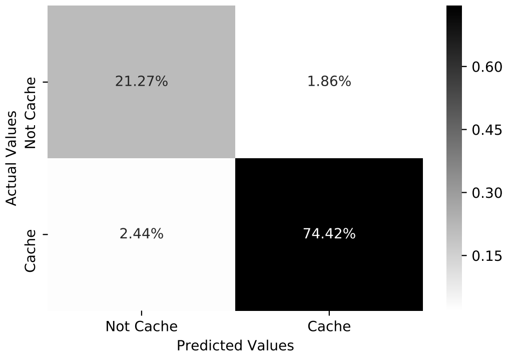


Figure 5.6: Confusion matrix of logistic regression model.

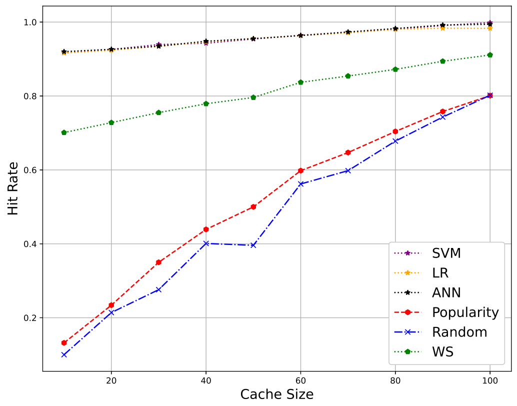


Figure 5.7: Impact of cache size (% of the total library size) on the cache hit rate.

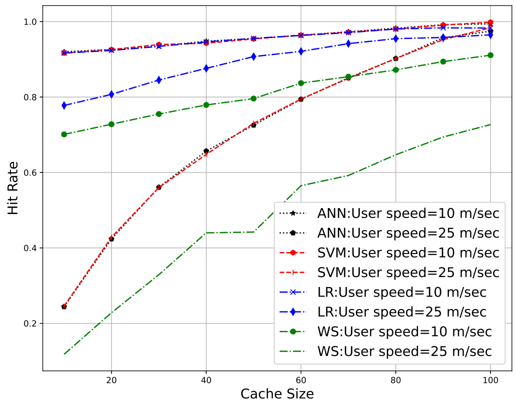


Figure 5.8: Impact of cache size (% of the total library size) on the hit rate with different values of user speed.

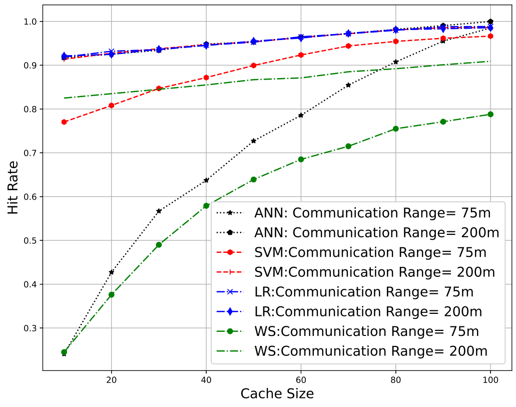


Figure 5.9: Impact of cache size (% of the total library size) on the hit rate with different values of SBSs communication range.

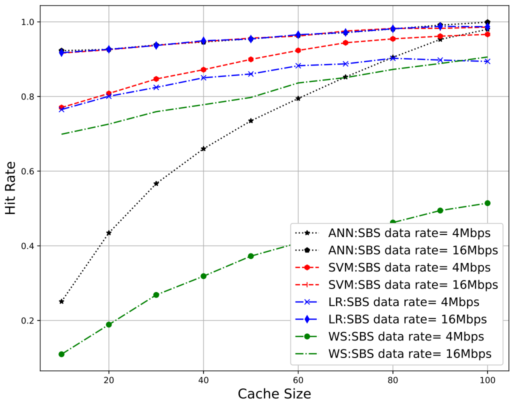


Figure 5.10: Impact of cache size (% of the total library size) on the hit rate with different values of SBSs data rate.

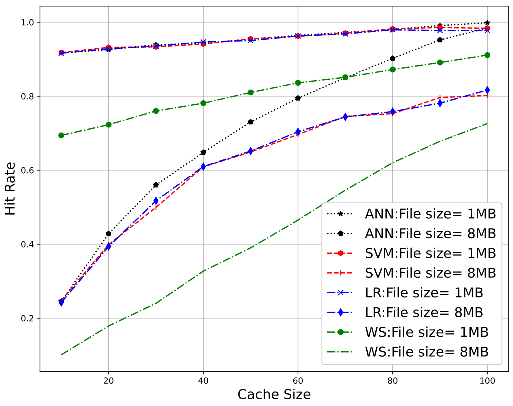


Figure 5.11: Impact of cache size (% of the total library size) on the hit rate with different values of file size.

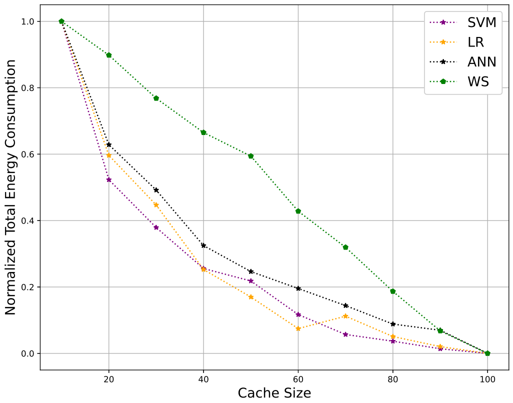


Figure 5.12: Impact of cache size (% of the total library size) on the normalized total energy consumption with different cache placement algorithms.

Chapter 6

Cache Placement Using Semi-Supervised Learning

In the previous chapter, supervised classification models and labeled data (input attributes and the corresponding target output) are used to train the cache placement model. However, in practical applications, collecting labeled samples is difficult, expensive, and time-consuming. When a user is sending requests for contents while moving in the cell, the unlabeled data are abundant and easy to obtain with every user request. Furthermore, supervised learning may face the overfitting of the training dataset. This occurs when the supervised classifier tends to fit itself to the statistical noise in the training sample. With the different and many changing input attributes in the wireless networks, errors are usually unavoidable and unpredictable. The resulting classifier will not model the true relationship between the input attributes and the corresponding output. In this chapter, we propose a multi-objective cache content strategy that aims to maximize the cache hit rate of small base stations (SBSs) in mobile edge networks (MENs) based on a semi-supervised self-trained learning technique (SSST). The strategy is formulated as a classification problem. A logistic regression learning technique is used to obtain a classifier based on a small amount of labeled data. Subsequently, unlabeled samples are classified by the classifier using a self-training algorithm. Finally, the cases with the highest prediction probability are added to the training set to increase the number of samples in the dataset. The proposed strategy is implemented and tested on samples with different input features, such as file popularity, user location, user to SBS contact probability, communication range, and contact duration.

6.1 Introduction

In supervised classification problems, labeled data (input attributes and the corresponding target output) are used for training the model. However, in practical applications, collecting labeled samples is difficult, expensive, and time-consuming. When a user is sending requests for contents while moving in the cell, the unlabeled data are abundant and easy to obtain with every user request. Furthermore, supervised learning may face the overfitting of the training dataset. This occurs when the supervised classifier tends to fit itself to the statistical noise in the training sample. With the different and many changing input attributes in the wireless networks, errors are usually unavoidable and unpredictable. Therefore, the resulting classifier will not model the true relationship between the input attributes and the corresponding output. In [223], the authors showed the ability to use semi-supervised learning to improve the classifier performance and avoid the overfitting problem. The authors in [224] suggested the use of a small amount of labeled data sets with a large amount of unlabeled datasets to build a semi-supervised self-training classifier model for disease classification. Their experimental results showed the effectiveness of using the classifier in solving the disease classification problem.

In [225], we proposed a semi-supervised self-trained learning technique (SSST) for the design of latency efficient cache placement binary classifier in mobile edge networks (MENs). This chapter first trains a supervised classifier based on the logistic regression (LR) learning technique using a small amount of labeled samples. Then, we use the label propagation algorithm presented in [226] to self-train the classifier and generate pseudo labels by using the trained classifier to predict class labels for all the unlabeled data. Then, the pseudo labels with the highest probability of being correct are added as a pseudo labels to the labeled training set. Finally, we retrain the classifier to predict cache contents for the labeled test dataset and use the results to evaluate the prediction accuracy. Our results show that the semi-supervised self-trained learning method for cache placement can outperform other cache placement techniques. Furthermore, the proposed technique can self-train and improve the classification accuracy when sufficient data for training the model are not available. Fault tolerance is analyzed to explore the relationship between system performance (cache hit rate and total energy consumption) and the robustness of the proposed cache placement algorithm when random failures occur in SBSs.

6.2 Semi-Supervised Self-Training (SSST) Binary Classifier

Semi-supervised self-training consists of two main steps. In the first step, a traditional supervised classifier is trained on a small set of labeled data to build the binary classifier. This chapter trains an LR classifier using a small set of labeled training data and predicts on the test data set. Then, the second step is to use a self-training technique, an iterative method for semi-supervised learning. Self-training uses the prediction probabilities to label unlabeled data. The labels with high probability prediction values are added to the dataset as new labels and will be used in the next iteration.

Consider the training data set, also called the observations, that consists of n input features $(I_1, I_2, \cdots, I_n)$ and the corresponding class y for $\mathcal{V}$ number of observations. The classifier output y can be 1 that means the observation belongs to the class cache or 0 which means the observation belongs to the class not cache. Suppose the dataset has m samples, which includes m_1 labeled samples and m_2 unlabeled samples, $m = m_1 + m_2$.

6.2.1 Logistic Regression for Supervised Classification

A supervised classifier requires a dataset consisting of input/output pairs. Each row in the data set represents the observations (input attributes), the vector of features and the corresponding class y. The classification function aims to build a model by training the given dataset, and then the model will be able to estimate the output $y\prime$ for new values of input features. The classifier learns from the previous datasets to predict the output when given new input features. The classifier's output is the decision of either placing the content (cache) or not placing the contents (not cache). The detailed discussion of logistic regression for supervised classification is presented in section 5.2.3.

6.2.2 Self-Training Technique

The trained classifier is used to predict the class label for all the unlabeled data samples. To find the data points I close to similar labels, a fully connected graph is created with the nodes representing all the data points for labeled and unlabeled data points. Any two nodes in the graph are connected

Algorithm 5 : Semi-Supervised Self-Training Classifier for Cache Placement

```
Input: Labeled dataset $m_1$ with input features $I_j$ and one output label $y$ and $m_2$ unlabeled dataset with input features $I_j$.
Initialize bias $w_0$=0, weights $w_j = 0$, and learning rate $\alpha = 0.3$.
Set training dataset $\Omega = m_1$.
While not converge OR iterations $\leq$ Maximum limit
    Train the LR classifier using labeled dataset $\Omega$.
    Generate pseudo labels with classifier predictions.
    Add the pseudo labels with 99% probability of prediction to the $\Omega$ dataset.
End While
Output: Classifier model $k(I, w)$ given in (5.5).
```

with weight d_{ij} so that the closer the nodes are in the Euclidean distance, the larger the weight is. The weights t_{ij} are controlled by the parameter σ, such that:

$$t_{ij} = \exp\bigg(- \frac{d_{ij}^2}{\sigma^2} \bigg) = \exp\bigg(- \frac{\sum_{d=1}^{D} (I_i^d - I_j^d)}{\sigma^2} \bigg). \tag{6.1}$$

The problem is to estimate the output y' from labeled and unlabeled data points. The model predicts labels for all unlabeled data and probabilities for those predictions, then adopts the high probability labels (in our work, we consider 99% as a high probability). The adopted labels are called ’pseudo labels and are added to the dataset for the next iterations. Finally, the classifier model is retrained again with the updated dataset. The iterations continue until there are no more predictions with probabilities higher than 99%, or no unlabeled data remains [226]. Algorithm 5 illustrates the steps for a semi-supervised self-training classifier for the cache placement technique. The workflow of our proposed semi-supervised self-training classifier for cache placement is shown in Fig. 6.1.

6.3 Simulation Results

In this section, we evaluate the performance of the classifier-based SSST algorithm for cache placement via simulations. We compare the performance of the proposed cache placement approach

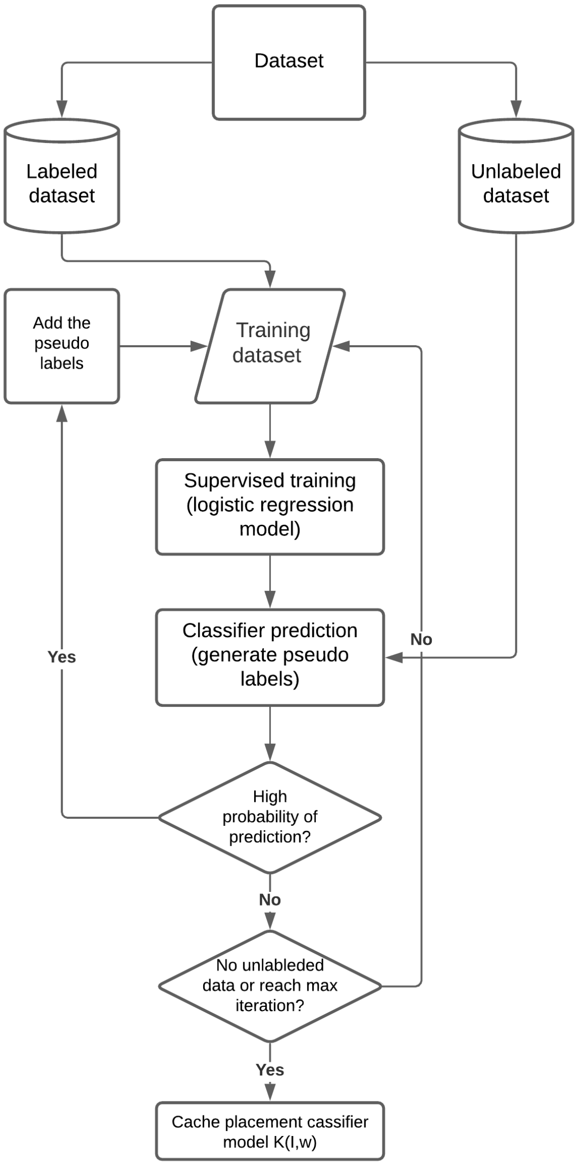

Dataset
Labeled dataset
Unlabeled dataset
Add the pseudo labels
Training dataset
Supervised training (logistic regression model)
Classifier prediction (generate pseudo labels)
Yes
No
High probability of prediction?
No
No unlableded data or reach max iteration?
Yes
Cache placement cassifier model K(I,w)

Figure 6.1: The work flow of our proposed semi-supervised self-training classifier for cache placement.

Table 6.1: Simulation parameters

Parameter	**Value**
Number of files in MBS	100
SBSs cache size	(10-100)% of MBS library
File size	(1-8) MB
SBS data rate	(2-16) Mbps
SBS communication range	(50-200) m
User speed	(10-30) m/sec
Number of user requests	10,000

with three supervised learning techniques: Artificial neural network (ANN), support vector machine (SVM), and logistic regression (LR) cache placement algorithms.

6.3.1 Simulation Setup

Our simulation is implemented on an area with a number of SBSs as described in [197]. We assume that the macrocell includes one MBS, 15 SBSs, and there are three different paths that users pass through with 11 locations in each path. The results are computed for $10,000$ randomly generated user locations and $10,000$ different user requests within one wireless network cell. Each request consists of the requested file ID. The requests follow the Zipf popularity distribution. A number of experiments are implemented to compare the impact of system parameters on the performance of different cache placement algorithms. The parameters are illustrated in Table 6.1. We assume that the user mobility parameters follow the work in [109]. It is defined as an independent Poisson process, the pairwise contact duration between mobile UT and SBS follows the exponential distribution with parameter $\lambda_{i,j}^{SBS}$, such that $\Gamma(10, 1/100)$ represents the contact rate between mobile UT and SBS. Figure 5.3 shows the first few lines of the training dataset, which will be used to build the cache placement model. The dataset is divided into 70% data used for training, and 30% data is used for testing. The simulation was implemented using Python on a Jupyter notebook installed in the Anaconda environment [221].

6.3.2 Learning Technique Classification Models

To evaluate the performance of the proposed SSST classifier-based cache placement algorithm and investigate the relationship among various input attributes, we compare the impact of different variables on four cache placement techniques. The cache placement techniques are: Semi-supervised self-training technique (SSST). Support vector machine (SVM) with SGD learning. Logistic regression (LR) with SGD learning. Artificial neural network with SGD learning (ANN).

The performance of the classification models is explored for the same dataset using a confusion matrix, which describes the classification model performance. It examines the testing dataset to find if the classification outcome is correct or not. The model accuracy scores are calculated to identify model errors [222].

Figure 6.2 shows the confusion matrix for SSST classification model. This figure summarizes the percentage of correct and incorrect predictions for each class and the types of errors made by the classifier. The correct estimation is organized along the diagonal direction from top-left to bottom-right of the matrix. There are 96% of the sample cases that belong to the not-cached category, and they are predicted not-cached by the SSST model. These samples are true negatives (TN), which means they are predicted not to cache, and the correct decision is not to cache. The other correct estimation is the true positive (TP) case, there are 98% of the samples, which are cases in which the actual decision was to cache, and they are predicted to be cached by the SSST model. This refers to the cases where users requested the files, and the files were stored in the caches. By looking at the classification errors, false positive (FP) gives 4.5%. This means a small percentage of samples in the SSST model are identified as cache, and they should not actually be cached. From the cache placement perspective, this can cause energy loss and more latency to store contents in some SBS caches that will not be requested from these SBSs and/or will not result in successful transmission to the requested UTs.

Table 6.2 describes in more depth the performance of the SSST estimator with three other learning techniques ANN, SVM, and LR., by applying ANN, SVM, and LR algorithms on the datasets and comparing the accuracy, precision, recall, and F1 score of the results. The accuracy gives the ratio of correctly predicted observations to the total observations. Accuracy is defined as (TP+TN)/(TP+FP+FN+TN). Accuracy alone will not indicate the performance of the estimator.

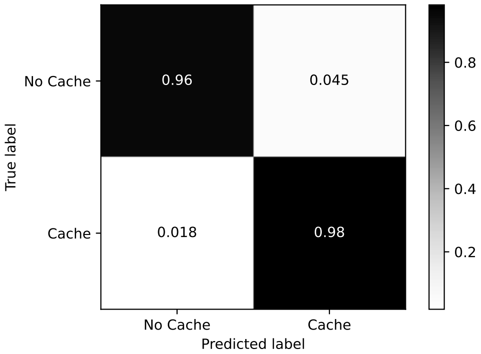


Figure 6.2: Confusion matrix of semi-supervised self-training classifier.

Analyzing other performance metrics will give us more understanding of our model before using it in the actual environment with different cases. The precision indicates how precise our model is, i.e., how many of these samples are actually positive out of those predicted positive (cache). Hence, Precision= TP/(TP+FP). The sensitivity or recall performance index measures the ratio of correctly predicted positive observation (cache) to all observations in the actual values, i.e., Recall = TP/(TP+FN)

Subsequently, we compute the weighted average of precision and recall to compute the F1 score performance index. F1 score helps to understand the model accuracy better than the accuracy performance index when we have uneven class distribution, where the F1 score is defined as

$$\text{F1 Score} = \frac{2 \times (\text{Recall} \times \text{Precision})}{(\text{Recall} + \text{Precision})} \tag{6.2}$$

Based on the results provided by Table 6.2, among the four classification models, SSST has the best performance and achieves the highest score compared to the other algorithms.

6.3.3 Performance Evaluation

We show only the results of caching in SBS caches. The four strategies are evaluated and compared by computing the cache hit rate for different experiments in this work. User mobility is considered by considering that users may request files while moving from one location to another location within the wireless network. Users can download files from nearby SBSs without the need to use a backhaul link. This will cause a cache hit. The cache hit rate indicates the number of cache hits

Table 6.2: Performance evaluation of classification models

Classification	**Classification Models**				
Performance	**SSST**	**ANN**	**SVM**	**LR**	**Best Score**
Accuracy	1.0	0.98	0.79	0.81	SSST
Precision	1.0	0.99	0.98	0.96	SSST
Recall	1.0	0.98	0.98	0.96	SSST
F1 Score	1.0	0.99	0.98	0.96	SSST

divided by the total number of user requests as given in (5.3).

6.3.3.1 Impact of Cache Size

To study the impact of SBS cache size on the cache hit rate, we consider SBS cache size ranges from 10% to 100% of the total library size. Figure 6.3 shows the effect of changing SBS cache sizes on the SBS hit rate. In all the cache placement algorithms, increasing the cache size increases the cache hit rate. This is because increasing cache size means more files can be cached in SBS caches, which increases the possibility of finding the requested contents.

We consider certain values for the variables user speed, SBS communication range, SBS data rate and file size to be equal to 10m/sec, 100m, 8Mbps, 1MB, respectively. The trained models in the four learning techniques can extract information from the dataset and give correct predictions for the cache content placement regardless of the cache size.

The performance of the learning techniques SSST, ANN, SVM, and LR are very similar at different cache sizes ranging from $10\% - 80\%$. At larger cache sizes of 90% and above, ANN and SSST give similar performance and outperform SVM and LR algorithms.

6.3.3.2 Impact of User Speed

Changing user speed from low to high impacts all algorithms. In our simulation, the speed is only increased to the point that the UT can still download the contents from SBS while the user is on the move. Figure 6.4 shows the impact of SBS cache size on hit rate for different user speeds. The figure shows the results for 10 m/sec and 25 m/sec for four cache placement techniques: SSST,

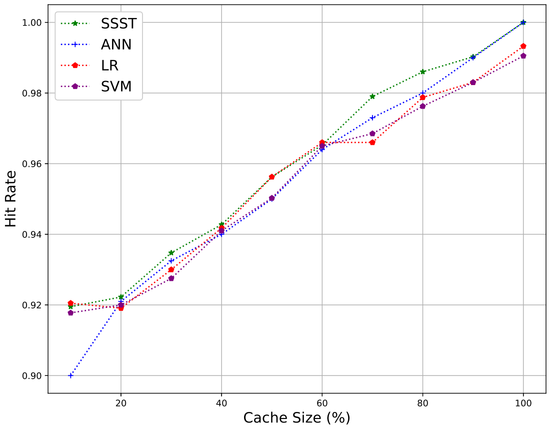


Figure 6.3: Impact of cache size (% of the total library size) on the hit rate.

ANN, SVM, and LR. We observed that the four techniques give similar results at low user speed (10 m/sec). However, when increasing user speed up to 25 m/sec, the SSST gives a higher hit rate than the LR, SVM, and ANN.

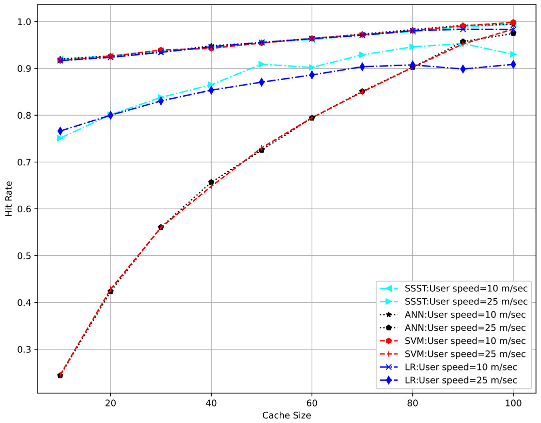


Figure 6.4: Impact of cache size (% of the total library size) on the hit rate with different values of user speed.

6.3.3.3 Impact of SBSs Communication Range

We explore the impact of changing the SBS communication range from 75m to 200m on cache placement performances shown in Fig. 6.5 as the results show, when the SBSs communication range increase, the SBSs cache hit rates increase in all the cache placement algorithms. This is because as the SBS communication range increases, the number of SBSs within the communication range of the moving UT increases. Thus, the possibility of finding the requested contents within the SBSs caches is higher. The SSST and LR approaches give a higher hit rate than ANN and SVM caching techniques at a lower SBS communication range. After increasing the SBS communication range up to 200m, the hit rates from the four learning techniques are very close to each other. In SSST and LR learning techniques, the resulting hit rates while changing the communication ranges from 75m up to 200m were relatively high for different values of cache sizes.

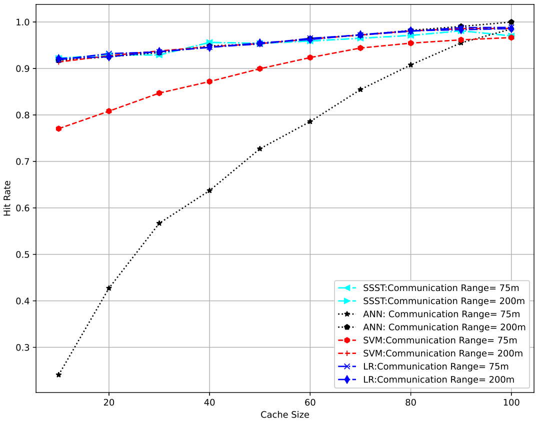


Figure 6.5: Impact of cache size (% of the total library size) on the hit rate with different values of SBSs communication range.

6.3.3.4 Impact of SBS Data Rate

Figure 6.6 presents the influence of changing SBS data rate on the cache hit rate. As the results show, increasing the SBS data rate increases the hit rate for all cache placement techniques since SBSs can transfer more data during contact with the UT. Increasing the SBS data rate allows the user to complete downloading the contents within available time while moving from one location to another location. When the SBS data rate is 4Mbps, the hit rate in the SSST technique outperforms the SVM, LR, and ANN caching algorithms. While for higher data rates (16 Mbps), the performances of the learning techniques SSST, ANN, SVM, and LR are very similar. As the figure illustrates, the SSST technique does not change the resulting hit rate values while changing the SBSs data rates.

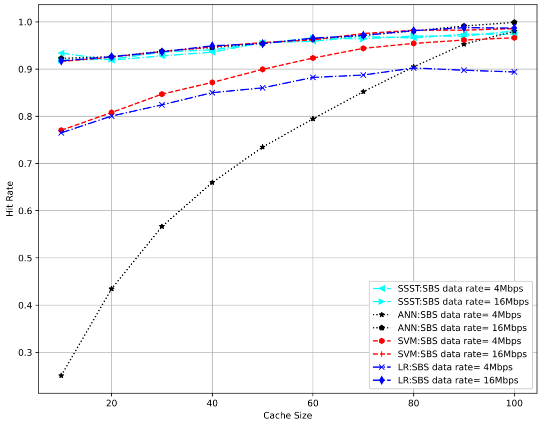


Figure 6.6: Impact of cache size (% of the total library size) on the hit rate with different values of SBSs data rate.

6.3.3.5 Impact of File Size

Figure 6.7 shows the impact of changing the file size from 1MB to 8MB on the SBS cache hit rate. As expected, the hit rate decreases when the file size increases. This is because the users are downloading the files while moving from one location to the next along the path. When the file size increases, the users will not complete downloading the whole file during the required time, which results in a miss. The SSST and ANN give a higher hit rate than the SVM and LR caching techniques when the file size is small. After increasing the file size up to 8MB, the SSST outperforms ANN, SVM, and LR.

6.3.3.6 Impact on SBS and MBS Energy Consumption

Figure 6.8 shows the impact of changing cache size and cache placement technique on the total energy consumption in SBSs and MBS required to download the files. As the figure shows, the normalized total energy consumption decreases with the increase in the cache size. As the SBSs cache size increases, the cache hit rate increases, which means the contents are mostly downloaded from the SBSs. Therefore, the increase in energy consumption for transmitting the contents from SBSs to UTs means decreasing the energy consumption for downloading the same contents from the MBS. When comparing the total energy consumption required to transfer the contents from SBSs and MBS to UTs between different cache placement algorithms, we see that the SSST-LR,

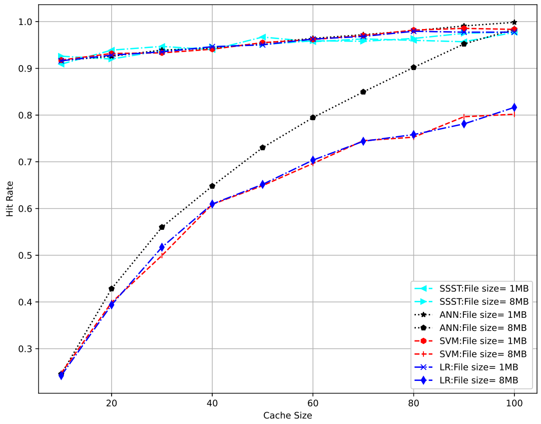


Figure 6.7: Impact of cache size (% of the total library size) on the hit rate with different values of file size.

SSST-SVM, and LR result in lower energy consumption at smaller cache sizes. On the other hand, the SSST-LR and SVM techniques result in lower total energy consumption than other techniques at larger cache sizes.

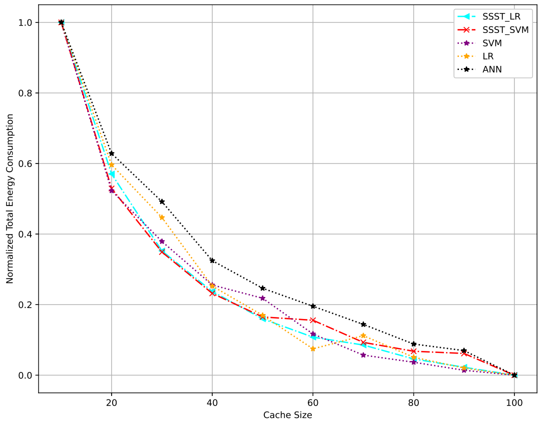


Figure 6.8: Impact of cache size (% of the total library size) on the normalized total energy consumption with different cache placement algorithms.

6.3.4 Fault-Tolerance

In MENs, SBS hardware or software failures, power failure, and network failure can occur, resulting in data being unavailable [227]. The SBS failure causes data access failure that results in SBS cache miss. In our proposed SSST cache placement algorithm and the three supervised learning techniques, data replicate placed on different SBSs to ensure the availability of the requested contents. To evaluate the impact of fault tolerance on the performance of the caching system in the MEN, we compare the SBS cache hit and the total energy consumption for delivering the requested contents to users for the proposed SSST and the three supervised learning techniques while changing two system parameters, user speed, and SBS communication range. Figures 6.9 and 6.10 illustrate the impact of SBS failure on the cache hit rate for cache size of 40% from the main library in one wireless network cell for user speeds equal to 10 m/sec and 25 m/sec, respectively. As the figures show, increasing user speed while having random failures in SBSs reduces the cache hit rate. The SSST algorithm does not behave as sensitive to changes in user speed. Figure 6.11 shows the impact of minimizing the SBS communication range to 75m while having randomly generated failures in SBSs on the cache hit rate. As the results show, SSST results in a higher hit rate compared to other cache placement techniques. Figure 6.12 illustrates the impact of minimizing the SBS communication range to 75m while having randomly generated failures in SBSs on the total energy consumption required to deliver the contents to the user terminals. In all learning algorithms, increasing the failure in SBSs while reducing the communication range to 75m reduce the number of working SBSs within the range of the UT. This leads to a decrease in the SBS cache hit rates, i.e., an increase in the SBS cache miss rates. At each cache miss, the user downloads the requested content from MBS instead of SBSs, which increases the total energy consumption required to deliver the contents to UTs. Since the SSST algorithm outperformed other techniques and resulted in higher cache hit rates, so it is expected that the total energy consumption will be less than other learning techniques as shown in Fig.6.11 and 6.12, respectively.

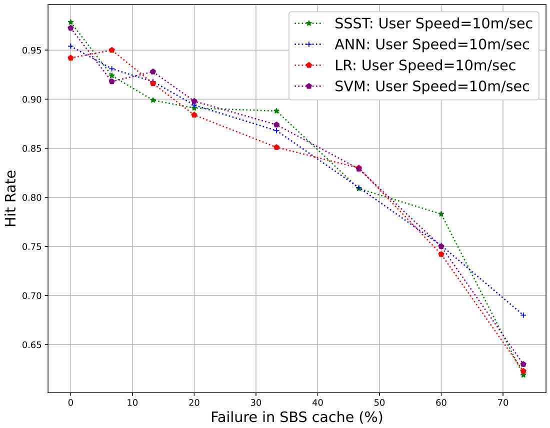


Figure 6.9: Impact of failure in SBSs on the cache hit rate with 10 m/sec user speed for different cache placement algorithms.

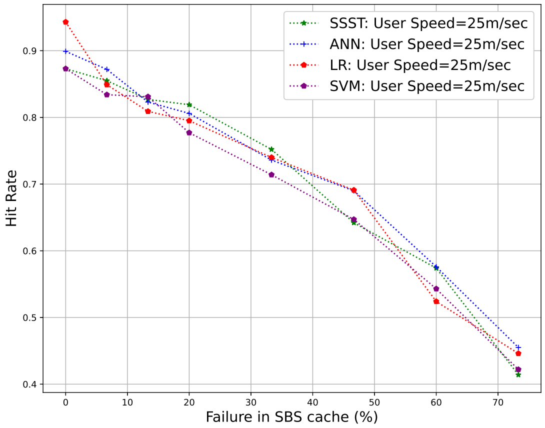


Figure 6.10: Impact of failure in SBSs on the cache hit rate with 25 m/sec user speed for different cache placement algorithms.

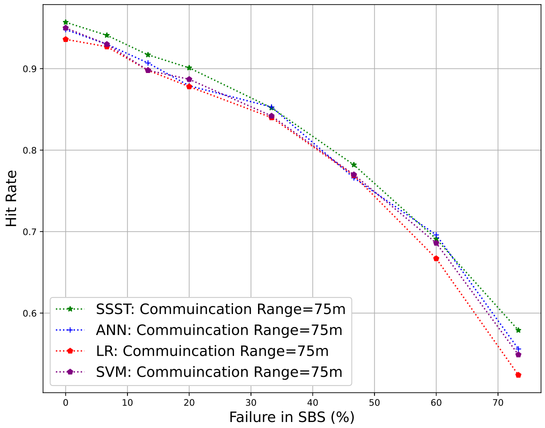


Figure 6.11: Impact of failure in SBSs on the cache hit rate with 75 m communication range for different cache placement algorithms.

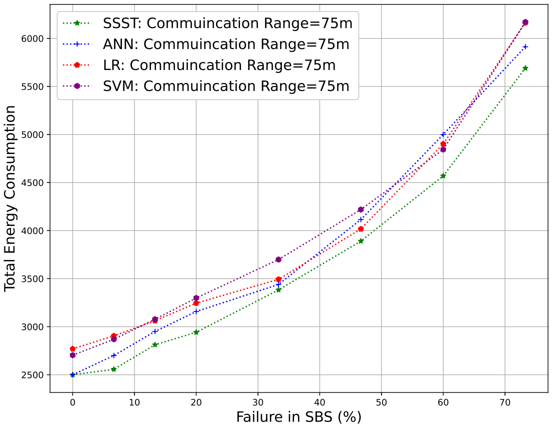


Figure 6.12: Impact of failure in SBSs on the cache hit rate with 75 m communication range for different cache placement algorithms.

6.4 Discussion

In this chapter, we developed a semi-supervised self-training classification model for the cache placement problem. We assessed the proposed SSST algorithm through experiments with datasets on different learning techniques. The performance comparison of different machine learning models was carried out with the same datasets. The test was made by changing one of the system parameters, fixing the other parameters, and computing the hit rate to investigate the sensitivity of the classification by the changes in the environmental parameters. We observed that using the SSST cache placement algorithm while changing system parameters, the hit rate of the SBS cache increases with the increase in SBS cache size decreases with the increase of user speed and remains unchanged when the SBS data rate and SBS communication ranges change. This work yielded promising results for the formulation of cache placement as a classification problem. Further experiments were implemented to understand the relationship between system performance and the robustness in our framework. As the results showed, the SSST algorithm was more robust and improved the performance of the cache placement problem. Continuous dataset accumulation is suggested to understand the effect of classification and the change in system parameters on the adaptivity and robustness of SSST for cache placement in mobile edge networks.

6.5 Summary

In the previous chapter, we proposed a latency-efficient multi-objective cache content strategy as a classification problem. This chapter proposed a new strategy to model the classification problem using the semi-supervised-self training technique. We used a small amount of labeled data and a logistic regression learning technique to obtain a classifier. Then, we classified unlabeled samples by the classifier using a self-training algorithm. The cases with the highest prediction probability were added to the training set to increase the number of samples in the dataset. We implemented and tested our proposed strategy on samples with different input features, such as file popularity, user location, user to SBS contact probability, communication range, and contact duration. Using experimental results on actual user request sequences data, we showed that the proposed algorithm significantly improves the prediction accuracy. It also saves time for labeling the data, making it an efficient solution to the cache placement problem. Finally, we evaluated the robustness of our proposed algorithm by studying its fault tolerance and testing different system parameters in a wireless network. We showed that our proposed algorithm is sensitive to some system parameters and has better fault tolerance for other system parameters.

Chapter 7

Conclusions and Future Work

Mobile edge network (MEN) has been considered as a network model that enables the use of caching capabilities at the edge of the network in the macro base station (MBS), small base stations (SBSs), and user terminals (UTs). We studied the effect of file popularity, SBS communication range, contact duration, and SBS to UT contact probability on cache placement in a mobile edge network. In this dissertation research, a mobility-aware latency efficient cache placement was formulated that aims to maximize the cache hit rate, which minimizes the latency of downloading the contents.

7.1 Conclusion

Upon reviewing recent developments in the design of caching in MEN, we noted several challenges in modeling and implementing caching placement, access, and delivery at the edge of the network due to continuous changes in content popularity and user mobility, and a number of users within the network. More challenges appear in caching at MENs due to high computation requirements of future applications that need to satisfy power and delivery time constraints with the quality of service requirements, improved network throughput, and reduced end-to-end and backhaul delay costs. Therefore, research was required to investigate the development of algorithms for cache placement, cache access, and cache delivery by utilizing mobile edge networks' data storage and computing capabilities. The main focus is on using machine learning and intelligent decision-making techniques to implement the algorithms.

In chapter 2, energy and latency efficient caching in mobile edge networks (MENs) were reviewed.

MEN enables caching capabilities at the edge of the network in the macro base station, small base stations, and user terminals. First, different caching techniques were presented and compared. Then the challenges that face the design of the caching system in MEN were discussed. We focused on the use of decision and learning theoretical approaches to solve these problems. MENs also enable complex computation, which allows deep learning techniques to be adapted in these networks to solve problems related to energy and latency constraints.

In chapter 3, we presented our cache system model for the mobile edge networks (MENs) to provide cache content placement, cache content access, and cache content delivery, taking into account different storage capacities, users mobility, content popularity distribution, latency, and power consumption. The system model includes the following models: network model, cache content model, transmission model, mobility model, and power consumption model. Then, we presented and discussed the problem formulations related to finding the optimal solution for cache content placement with minimum latency and cache content delivery with minimum power consumption.

In chapter 4, a weighted-sum approach is proposed to solve this problem. Simulation results show the impact of changing some system variables on cache hit rate. Furthermore, the results showed that the weighted-sum strategy offers better performance under the formulation of multi-objective function. Also, as the hit rate is maximized, the total energy consumption for delivering the contents to UTs is minimized because contents are delivered from SBS and other UT caches instead of delivering contents from MBS.

In chapter 5, the work focused on formulating the cache placement as a binary classification problem that can be solved using machine learning techniques. Different learning techniques were investigated to understand the problem and the input attributes correlated to the classification decisions. We analyzed the characteristics of the input features (attributes) related to the decision of cache content placement objectives and their properties. These input attributes were important to form the hyperplane that separated the multi-dimensional space into two decision regions (cache contents and not cache contents). The performance comparison of different machine learning models was carried out with the same datasets. Finally, the cache placement models using artificial neural networks, support vector machine, logistic regression, and weighted-sum algorithms were tested with actual data sets of user requests taken from published literature [197]. The test was conducted by changing one of the system parameters, fixing the other parameters, and computing the hit rate to investigate the sensitivity of the classification to the changes in the environmental parameters.

In chapter 6, we developed a semi-supervised self-training (SSST) classification model for the cache placement problem. We assessed the proposed SSST algorithm through experiments with datasets on different learning techniques. The performance comparison of different machine learning models was carried out with the same datasets. We observed that using the SSST cache placement algorithm while changing system parameters, the hit rate of the SBS cache increases with the increase in SBS cache size decreases with the increase of user speed and remains unchanged when the SBS data rate and SBS communication ranges change. This work yielded promising results for the formulation of cache placement as a classification problem. Further experiments were conducted to understand the relationship between system performance and the robustness in our framework. As the results showed, the SSST algorithm was more robust and improved the performance of the cache placement problem.

7.2 Future Work

This research work yielded promising results for the formulation of cache placement. Therefore, continuous investigation and more dataset accumulation are suggested to model the trade-off between cache placement and the change in system parameters and the latency and energy consumption caused by downloading the contents by requested users. It has been noticed that with the changes in system parameters such as SBS cache size, user speed, SBS data rate, SBS communication range, and content popularity, there were a few limitations and weaknesses in each classifier model used in this research. Therefore, generating an automatic algorithm selection that decides which algorithm is executed in each wireless network cell depending on the size of the dataset, the number of input features, the available resources, and values of system parameters is suggested for future work. Furthermore, the automatic selection of the algorithm can be a rule-based technique aiming to combine different learning algorithms.

For mobile edge caching, it is clear that machine learning will play a significant role, particularly in terms of content request prediction and cache replacement tasks prediction. Wireless networks generate huge amounts of data every second. The data contains information about the content, its popularity, the user's preferences, the location of the user, and the mobility of the user. To achieve the best prediction of cache contents and to minimize energy consumption and latency, big data analytics should be used for cache placement.

Bibliography

[1] E. Baştuğ, M. Bennis, and M. Debbah, "Proactive caching in 5g small cell networks," *Towards 5G: Applications, Requirements and Candidate Technologies*, pp. 78–98, 2016.

[2] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting Caching and Multicast for 5G Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 15, pp. 2995–3007, April 2016.

[3] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[4] E. Bastug, M. Bennis, and M. Debbah, "Social and spatial proactive caching for mobile data offloading," *2014 IEEE International Conference on Communications Workshops, ICC 2014*, pp. 581–586, 2014.

[5] M. K. Kiskani and H. R. Sadjadpour, "Multihop Caching-Aided Coded Multicasting for the Next Generation of Cellular Networks," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 2576–2585, March 2017.

[6] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," *Mobile Networks and Applications*, vol. 19, pp. 133–143, April 2014.

[7] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[8] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

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

[10] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[11] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, 2009.

[12] Z. Pang, L. Sun, Z. Wang, E. Tian, and S. Yang, "A survey of cloudlet based mobile computing," in *Cloud Computing and Big Data (CCBD), 2015 International Conference on*, pp. 268–275, IEEE, 2015.

[13] X. Zhang and Q. Zhu, "Distributed mobile devices caching over edge computing wireless networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2017 IEEE Conference on*, pp. 127–132, IEEE, 2017.

[14] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[15] M. Marjanović, A. Antonić, and I. P. Žarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, 2018.

[16] T. Subramanya, L. Goratti, S. N. Khan, E. Kafetzakis, I. Giannoulakis, and R. Riggio, "A practical architecture for mobile edge computing," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017 IEEE Conference on*, pp. 1–4, IEEE, 2017.

[17] X. Wei, S. Wang, A. Zhou, J. Xu, S. Su, S. Kumar, and F. Yang, "Mvr: An architecture for computation offloading in mobile edge computing," in *Edge Computing (EDGE), 2017 IEEE International Conference on*, pp. 232–235, IEEE, 2017.

[18] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, "A mobile edge computing-assisted video delivery architecture for wireless heterogeneous networks," in *Computers and Communications (ISCC), 2017 IEEE Symposium on*, pp. 534–539, IEEE, 2017.

[19] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[20] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, "A mobile edge computing-based architecture for improved adaptive http video delivery," in *Standards for Communications and Networking (CSCN), 2016 IEEE Conference on*, pp. 1–6, IEEE, 2016.

[21] E. Cau, M. Corici, P. Bellavista, L. Foschini, G. Carella, A. Edmonds, and T. M. Bohnert, "Efficient exploitation of mobile edge computing for virtualized 5g in epc architectures," in *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2016 4th IEEE International Conference on*, pp. 100–109, IEEE, 2016.

[22] S.-C. Huang, Y.-C. Luo, B.-L. Chen, Y.-C. Chung, and J. Chou, "Application-aware traffic redirection: A mobile edge computing implementation toward future 5g networks," in *Cloud and Service Computing (SC2), 2017 IEEE 7th International Symposium on*, pp. 17–23, IEEE, 2017.

[23] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, 2018.

[24] E. Pencheva, I. Atanasov, K. Kassev, and V. Trifonov, "Location service in mobile edge computing," in *Ubiquitous and Future Networks (ICUFN), 2017 Ninth International Conference on*, pp. 617–622, IEEE, 2017.

[25] X. Huang, Z. Zhao, and H. Zhang, "Cooperate caching with multicast for mobile edge computing in 5g networks," in *Vehicular Technology Conference (VTC Spring), 2017 IEEE 85th*, pp. 1–6, IEEE, 2017.

[26] T. S. Darwish and K. A. Bakar, "Fog based intelligent transportation big data analytics in the internet of vehicles environment: Motivations, architecture, challenges and critical issues," *IEEE Access*, 2018.

[27] F. Bonomi, S. Poledna, and W. Steiner, "The role of fog computing in the future of the automobile," *Fog for 5G and IoT*, p. 191, 2017.

[28] D. Roca, J. V. Quiroga, M. Valero, and M. Nemirovsky, "Fog function virtualization: A flexible solution for iot applications," in *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on*, pp. 74–80, IEEE, 2017.

[29] A. A. Khan, M. Abolhasan, and W. Ni, "5g next generation vanets using sdn and fog computing framework," in *Consumer Communications & Networking Conference (CCNC), 2018 15th IEEE Annual*, pp. 1–6, IEEE, 2018.

[30] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, 2018.

[31] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, 2017.

[32] E. Yigitoglu, M. Mohamed, L. Liu, and H. Ludwig, "Foggy: A framework for continuous automated iot application deployment in fog computing," in *AI & Mobile Services (AIMS), 2017 IEEE International Conference on*, pp. 38–45, IEEE, 2017.

[33] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE access*, vol. 5, pp. 9882–9910, 2017.

[34] A. Jain and P. Singhal, "Fog computing: Driving force behind the emergence of edge computing," in *System Modeling & Advancement in Research Trends (SMART), International Conference*, pp. 294–297, IEEE, 2016.

[35] A. Enayet, M. A. Razzaque, M. M. Hassan, A. Alamri, and G. Fortino, "A mobility-aware optimal resource allocation architecture for big data task execution on mobile cloud in smart cities," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 110–117, 2018.

[36] A. B. Lazreg, A. B. Arbia, and H. Youssef, "A synchronized offline cloudlet architecture," in *Engineering & MIS (ICEMIS), 2017 International Conference on*, pp. 1–6, IEEE, 2017.

[37] X. Sun and N. Ansari, "Green cloudlet network: A sustainable platform for mobile cloud computing," *IEEE Transactions on Cloud Computing*, 2017.

[38] Y. Wu and L. Ying, "A cloudlet-based multi-lateral resource exchange framework for mobile users," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pp. 927–935, IEEE, 2015.

[39] Y. Jararweh, L. Tawalbeh, F. Ababneh, and F. Dosari, "Resource efficient mobile computing using cloudlet infrastructure," in *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, pp. 373–377, IEEE, 2013.

[40] H. Xingye, L. Xinming, and L. Yinpeng, "Research on resource management for cloud computing based information system," in *Computational and Information Sciences (ICCIS), 2010 International Conference on*, pp. 491–494, IEEE, 2010.

[41] J. Poderys, M. Artuso, C. M. O. Lensbøl, H. L. Christiansen, and J. Soler, "Caching at the mobile edge: A practical implementation," *Ieee Access*, vol. 6, pp. 8630–8637, 2018.

[42] X. Jiang, Z. Liu, L. Zhong, Y. Cui, and Y. Ji, "Coordinated edge-caching for content delivery in future internet architecture," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2017.

[43] A. Kalla and S. Sharma, "Exploring off-path caching with edge caching in information centric networking," in *Computational Techniques in Information and Communication Technologies (ICCTICT), 2016 International Conference on*, pp. 630–635, IEEE, 2016.

[44] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5g systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[45] W. P. Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021," 2017.

[46] I. Parvez, A. Rahmati, I. Güvenç, A. I. Sarwat, and H. Dai, “A survey on low latency towards 5g: Ran, core network and caching solutions,” *CoRR*, vol. abs/1708.02562, 2017.

[47] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, “Mobile edge computing potential in making cities smarter,” *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[48] H. Ko, J. Lee, and S. Pack, “Spatial and temporal computation offloading decision algorithm in edge cloud-enabled heterogeneous networks,” *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2018.

[49] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, “A survey of computation offloading for mobile systems,” *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[50] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoniem, “Energy efficient task caching and offloading for mobile edge computing,” *IEEE Access*, 2018.

[51] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[52] Y. Cui and D. Jiang, “Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks,” *IEEE transactions on Wireless Communications*, vol. 16, no. 1, pp. 250–264, 2017.

[53] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, “A survey on low latency towards 5g: Ran, core network and caching solutions,” *IEEE Communications Surveys & Tutorials*, 2018.

[54] W. Ding, W. Qi, J. Wang, and B. Chen, “Openscaas: an open service chain as a service platform toward the integration of sdn and nfv,” *IEEE Network*, vol. 29, no. 3, pp. 30–35, 2015.

[55] C.-H. Hung, Y.-C. Hsieh, and L.-C. Wang, “Control plane latency reduction for service chaining in mobile edge computing system,” in *Network and Service Management (CNSM), 2017 13th International Conference on*, pp. 1–5, IEEE, 2017.

[56] M. Chen, Y. Qian, Y. Hao, Y. Li, and J. Song, "Data-driven computing and caching in 5g networks: Architecture and delay analysis," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 70–75, 2018.

[57] S. Ranadheera, S. Maghsudi, and E. Hossain, "Computation offloading and activation of mobile edge computing servers: A minority game," *IEEE Wireless Communications Letters*, 2018.

[58] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017 IEEE 28th Annual International Symposium on*, pp. 1–6, IEEE, 2017.

[59] H. Guo, J. Liu, H. Qin, and H. Zhang, "Collaborative computation offloading for mobile-edge computing over fiber-wireless networks," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2017.

[60] C. Luo, S. Salinas, M. Li, and P. Li, "Energy-efficient autonomic offloading in mobile edge computing," in *Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence & Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 2017 IEEE 15th Intl*, pp. 581–588, IEEE, 2017.

[61] M. Deng, H. Tian, and X. Lyu, "Adaptive sequential offloading game for multi-cell mobile edge computing," in *Telecommunications (ICT), 2016 23rd International Conference on*, pp. 1–5, IEEE, 2016.

[62] B. Perabathini, E. Baştuğ, M. Kountouris, M. Debbah, and A. Conte, "Caching at the edge: a green perspective for 5g networks," in *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pp. 2830–2835, IEEE, 2015.

[63] M.-C. Lee and A. F. Molisch, "Individual preference aware caching policy design for energy-efficient wireless d2d communications," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–7, IEEE, 2017.

[64] J. Zhang, X. Zhang, Z. Yan, Y. Li, W. Wang, and Y. Zhang, “Social-aware cache information processing for 5g ultra-dense networks,” in *Wireless Communications & Signal Processing (WCSP), 2016 8th International Conference on*, pp. 1–5, IEEE, 2016.

[65] M. Gregori, J. Gómez-Vilardebó, J. Matamoros, and D. Gündüz, “Wireless content caching for small cell and d2d networks,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1222–1234, 2016.

[66] Y. Cui, W. He, C. Ni, C. Guo, and Z. Liu, “Energy-efficient resource allocation for cache-assisted mobile edge computing,” *arXiv preprint arXiv:1708.04813*, 2017.

[67] C. Liang, Y. He, F. R. Yu, and N. Zhao, “Energy-efficient resource allocation in software-defined mobile networks with mobile edge computing and caching,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2017 IEEE Conference on*, pp. 121–126, IEEE, 2017.

[68] S. Mrad, S. Hamouda, and H. Rezig, “Graph theory based multicast caching for better energy saving in dense small cell networks,” in *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*, pp. 2015–2020, IEEE, 2017.

[69] J. Zhang, W. Xia, F. Yan, and L. Shen, “Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing,” *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2018.

[70] D. Wu, Q. Liu, H. Wang, D. Wu, and R. Wang, “Socially aware energy-efficient mobile edge collaboration for video distribution,” *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2197–2209, 2017.

[71] J. Liu and Q. Zhang, “Offloading schemes in mobile edge computing for ultra-reliable low latency communications,” *Ieee Access*, no. 99, pp. 1–13, 2018.

[72] C.-F. Liu, M. Bennis, and H. V. Poor, “Latency and reliability-aware task offloading and resource allocation for mobile edge computing,” *arXiv preprint arXiv:1710.00590*, 2017.

[73] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[74] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, 2017.

[75] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2017.

[76] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," in *Edge Computing (EDGE), 2017 IEEE International Conference on*, pp. 47–54, IEEE, 2017.

[77] Z. Zhu, J. Peng, X. Gu, H. Li, K. Liu, Z. Zhou, and W. Liu, "Fair resource allocation for system throughput maximization in mobile edge computing," *IEEE Access*, vol. 6, pp. 5332–5340, 2018.

[78] M. E. C. Santos, A. Chen, T. Taketomi, G. Yamamoto, J. Miyazaki, and H. Kato, "Augmented reality learning experiences: Survey of prototype design and evaluation," *IEEE Transactions on learning technologies*, vol. 7, no. 1, pp. 38–56, 2014.

[79] M. Erol-Kantarci and S. Sukhmani, "Caching and computing at the edge for mobile augmented reality and virtual reality (ar/vr) in 5g," in *Ad Hoc Networks*, pp. 169–177, Springer, 2018.

[80] 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*, pp. 945–953, IEEE, 2012.

[81] A. Ravi and S. K. Peddoju, "Mobile computation bursting: An application partitioning and offloading decision engine," in *Proceedings of the 19th International Conference on Distributed Computing and Networking*, p. 46, ACM, 2018.

[82] J. Niu, S. Wang, W. Niu, and M. Atiquzzaman, "User-aware partitioning algorithm for mobile cloud computing based on maximum graph cuts," *Computer Networks*, vol. 129, pp. 193–206, 2017.

[83] Y. Jararweh, A. Doulat, A. Darabseh, M. Alsmirat, M. Al-Ayyoub, and E. Benkhelifa, "Sd-mec: Software defined system for mobile edge computing," in *Cloud Engineering Workshop (IC2EW), 2016 IEEE International Conference on*, pp. 88–93, IEEE, 2016.

[84] F. Messaoudi, A. Ksentini, G. Simon, and P. Bertin, "Performance analysis of game engines on mobile and fixed devices," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 4, p. 57, 2017.

[85] H. Tanaka, M. Yoshida, K. Mori, and N. Takahashi, "Multi-access edge computing: A survey," *Journal of Information Processing*, vol. 26, pp. 87–97, 2018.

[86] N. Takahashi, H. Tanaka, and R. Kawamura, "Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing," in *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on*, pp. 233–234, IEEE, 2015.

[87] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Towards edge-caching for image recognition," in *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pp. 593–598, IEEE, 2017.

[88] A. Anjum, T. Abdullah, M. Tariq, Y. Baltaci, and N. Antonopoulos, "Video stream analysis in clouds: An object detection and classification framework for high performance video analytics," *IEEE Transactions on Cloud Computing*, 2016.

[89] X. Wang, M. Chen, T. T. Kwon, L. Yang, and V. C. Leung, "Ames-cloud: A framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 811–820, 2013.

[90] P. Rost, I. Berberana, A. Maeder, H. Paul, V. Suryaprakash, M. Valenti, D. Wübben, A. Dekorsy, and G. Fettweis, "Benefits and challenges of virtualization in 5g radio access networks," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 75–82, 2015.

[91] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green internet of things," *IEEE Network*, vol. 32, no. 1, pp. 54–60, 2018.

[92] R. K. Barik, H. Dubey, and K. Mankodiya, "Soa-fog: Secure service-oriented edge computing architecture for smart health big data analytics," *arXiv preprint arXiv:1712.09098*, 2017.

[93] H. El-Sayed and M. Chaqfeh, "The deployment of mobile edges in vehicular environments," in *Information Networking (ICOIN), 2018 International Conference on*, pp. 322–324, IEEE, 2018.

[94] R. Munoz, J. Mangues-Bafalluy, R. Vilalta, C. Verikoukis, J. Alonso-Zarate, N. Bartzoudis, A. Georgiadis, M. Payaro, A. Perez-Neira, R. Casellas, *et al.*, "The cttc 5g end-to-end experimental platform: integrating heterogeneous wireless/optical networks, distributed cloud, and iot devices," *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 50–63, 2016.

[95] X. Chen, Q. Shi, L. Yang, and J. Xu, "Thriftyedge: Resource-efficient edge computing for intelligent iot applications," *IEEE Network*, vol. 32, no. 1, pp. 61–65, 2018.

[96] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[97] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani, "Internet-of-things-based smart cities: Recent advances and challenges," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 16–24, 2017.

[98] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[99] M. Al-Jabi, "Toward an iot-enabled adaptive interactive bus transportation system," in *Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS), 2017 2nd International Conference on the*, pp. 1–4, IEEE, 2017.

[100] Y. Liu, "Big data technology and its analysis of application in urban intelligent transportation system," in *Intelligent Transportation, Big Data & Smart City (ICITBS), 2018 International Conference on*, pp. 17–19, IEEE, 2018.

[101] T. Anagnostopoulos, A. Zaslavsky, K. Kolomvatsos, A. Medvedev, P. Amirian, J. Morley, and S. Hadjieftymiades, "Challenges and opportunities of waste management in iot-enabled smart cities: a survey," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 3, pp. 275–289, 2017.

[102] Y. Cao, H. Song, O. Kaiwartya, B. Zhou, Y. Zhuang, Y. Cao, and X. Zhang, "Mobile edge computing for big-data-enabled electric vehicle charging," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 150–156, 2018.

[103] Intel, "Intel 5g: A network transformation imperative." December 2015.

[104] G. mobile Suppliers Association *et al.*, "The road to 5g: Drivers, applications, requirements and technical development," 2015.

[105] Y. Qi, M. Hunukumbure, M. Nekovee, J. Lorca, and V. Sgardoni, "Quantifying data rate and bandwidth requirements for immersive 5g experience," in *Communications Workshops (ICC), 2016 IEEE International Conference on*, pp. 455–461, IEEE, 2016.

[106] P. Popovski, "Ultra-reliable communication in 5g wireless systems," in *5G for Ubiquitous Connectivity (5GU), 2014 1st International Conference on*, pp. 146–151, IEEE, 2014.

[107] F. R. Yu, T. Huang, and Y. Liu, *Integrated Networking, Caching, and Computing.* CRC Press, 2018.

[108] S. Kim, "5g network communication, caching, and computing algorithms based on the two-tier game model," *ETRI Journal*, vol. 40, no. 1, pp. 61–71, 2018.

[109] M. Chen, Y. Hao, L. Hu, K. Huang, and V. K. Lau, "Green and mobility-aware caching in 5g networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 12, pp. 8347–8361, 2017.

[110] A. A. Zewail and A. Yener, "Fundamental limits of secure device-to-device coded caching," in *Signals, Systems and Computers, 2016 50th Asilomar Conference on*, pp. 1414–1418, IEEE, 2016.

[111] E. Baştuğ, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 549–557, 2015.

[112] M. K. Kiskani, S. Vakilinia, and M. Cheriet, "Popularity based file categorization and coded caching in 5g networks," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017 IEEE 28th Annual International Symposium on*, pp. 1–5, IEEE, 2017.

[113] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2017.

[114] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, April 2016.

[115] S. Mller, O. Atan, M. van der Schaar, and A. Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, May 2016.

[116] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers," *IEEE Transactions on Information Theory*, vol. 59, pp. 8402–8413, Dec 2013.

[117] N.-S. Vo, T. Q. Duong, and M. Guizani, "Qoe-oriented resource efficiency for 5g two-tier cellular networks: A femtocaching framework," in *Global Communications Conference (GLOBECOM), 2016 IEEE*, pp. 1–6, IEEE, 2016.

[118] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, 2013.

[119] N. S. Vo, T. Q. Duong, and M. Guizani, “Qoe-oriented resource efficiency for 5g two-tier cellular networks: A femtocaching framework,” in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Dec 2016.

[120] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, “Learning distributed caching strategies in small cell networks,” in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, pp. 917–921, August 2014.

[121] S. Müller, O. Atan, M. van der Schaar, and A. Klein, “Smart caching in wireless small cell networks via contextual multi-armed bandits,” in *Communications (ICC), 2016 IEEE International Conference on*, pp. 1–7, IEEE, 2016.

[122] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, “Learning distributed caching strategies in small cell networks,” in *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*, pp. 917–921, IEEE, 2014.

[123] S. Mller, O. Atan, M. van der Schaar, and A. Klein, “Context-aware proactive content caching with service differentiation in wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 16, pp. 1024–1036, Feb 2017.

[124] S. Li, J. Xu, M. Van Der Schaar, and W. Li, “Popularity-driven content caching,” in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pp. 1–9, IEEE, 2016.

[125] Y. Wang, Y. Chen, H. Dai, Y. Huang, and L. Yang, “A learning-based approach for proactive caching in wireless communication networks,” in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, Oct 2017.

[126] Z. Zheng, L. Song, Z. Han, G. Y. Li, and H. V. Poor, “A stackelberg game approach to proactive caching in large-scale mobile edge networks,” *IEEE Transactions on Wireless Communications*, pp. 1–1, 2018.

[127] K. N. Doan, T. Van Nguyen, T. Q. Quek, and H. Shin, “Content-aware proactive caching for backhaul offloading in cellular network,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3128–3140, 2018.

[128] F. Pantisano, M. Bennis, W. Saad, and M. Debbah, "Cache-aware user association in backhaul-constrained small cell networks," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2014 12th International Symposium on*, pp. 37–42, IEEE, 2014.

[129] E. Bastug, M. Bennis, and M. Debbah, "Social and spatial proactive caching for mobile data offloading," in *Communications Workshops (ICC), 2014 IEEE International Conference on*, pp. 581–586, IEEE, 2014.

[130] J. Rao, H. Feng, C. Yang, Z. Chen, and B. Xia, "Optimal caching placement for d2d assisted wireless caching networks," in *Communications (ICC), 2016 IEEE International Conference on*, pp. 1–6, IEEE, 2016.

[131] X. Zhang, Y. Wang, R. Sun, and D. Wang, "Clustered device-to-device caching based on file preferences," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, Sep.

[132] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 2275–2284, Aug 2016.

[133] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 5, pp. 1444–1462, 2014.

[134] S. S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive scheme for caching youtube content in a cellular network: Machine learning approach," *Ieee Access*, vol. 5, pp. 5870–5881, 2017.

[135] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[136] T. Wang, L. Song, and Z. Han, "Dynamic femtocaching for mobile users," in *Wireless Communications and Networking Conference (WCNC), 2015 IEEE*, pp. 861–865, IEEE, 2015.

[137] K. Poularakis and L. Tassiulas, “Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 675–687, 2017.

[138] Y. Guan, Y. Xiao, H. Feng, C.-C. Shen, and L. J. Cimini, “Mobicacher: Mobility-aware content caching in small-cell networks,” in *Global Communications Conference (GLOBECOM), 2014 IEEE*, pp. 4537–4542, IEEE, 2014.

[139] X. Liu, J. Zhang, X. Zhang, and W. Wang, “Mobility-aware coded probabilistic caching scheme for mec-enabled small cell networks,” *IEEE Access*, vol. 5, pp. 17824–17833, 2017.

[140] R. Lan, W. Wang, A. Huang, and H. Shan, “Device-to-device offloading with proactive caching in mobile cellular networks,” in *Global Communications Conference (GLOBECOM), 2015 IEEE*, pp. 1–6, IEEE, 2015.

[141] R. Wang, J. Zhang, S. Song, and K. B. Letaief, “Mobility-aware caching in d2d networks,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 5001–5015, 2017.

[142] R. Wang, X. Peng, J. Zhang, and K. B. Letaief, “Mobility-aware caching for content-centric wireless networks: Modeling and methodology,” *IEEE Communications Magazine*, vol. 54, no. 8, pp. 77–83, 2016.

[143] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, “Cooperative content caching in 5g networks with mobile edge computing,” *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, 2018.

[144] R. Wang, J. Zhang, S. Song, and K. B. Letaief, “Exploiting mobility in cache-assisted d2d networks: Performance analysis and optimization,” *arXiv preprint arXiv:1806.04069*, 2018.

[145] T. Deng, G. Ahani, P. Fan, and D. Yuan, “Cost-optimal caching for d2d networks with user mobility: Modeling, analysis, and computational approaches,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3082–3094, 2018.

[146] M. Chen, Y. Hao, M. Qiu, J. Song, D. Wu, and I. Humar, "Mobility-aware caching and computation offloading in 5g ultra-dense cellular networks," *Sensors*, vol. 16, no. 7, p. 974, 2016.

[147] Y. Wang, Y. Chen, H. Dai, Y. Huang, and L. Yang, "A learning-based approach for proactive caching in wireless communication networks," in *Wireless Communications and Signal Processing (WCSP), 2017 9th International Conference on*, pp. 1–6, IEEE, 2017.

[148] Q. N. Nguyen, M. Arifuzzaman, K. Yu, and T. Sato, "A context-aware green information-centric networking model for future wireless communications," *IEEE Access*, vol. 6, pp. 22804–22816, 2018.

[149] H. Abou-Zeid and H. Hassanein, "Toward green media delivery: location-aware opportunities and approaches," *IEEE Wireless Communications*, vol. 21, no. 4, pp. 38–46, 2014.

[150] X. Huang and N. Ansari, "Content caching and distribution in smart grid enabled wireless networks," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 513–520, 2017.

[151] X. Peng, Y. Shi, J. Zhang, and K. B. Letaief, "Layered group sparse beamforming for cache-enabled green wireless networks," *IEEE Transactions on Communications*, vol. 65, no. 12, pp. 5589–5603, 2017.

[152] P. Duan, Y. Jia, L. Liang, J. Rodriguez, K. M. S. Huq, and G. Li, "Space-reserved cooperative caching in 5g heterogeneous networks for industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, p. 2715, 2018.

[153] F. Gabry, V. Bioglio, and I. Land, "On energy-efficient edge caching in heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3288–3298, 2016.

[154] F. Guo, H. Zhang, X. Li, H. Ji, and V. C. Leung, "Joint optimization of caching and association in energy harvesting powered small cell networks," *IEEE Transactions on Vehicular Technology*, 2018.

[155] X. Zhang, T. Lv, W. Ni, J. M. Cioffi, N. C. Beaulieu, and Y. J. Guo, “Energy-efficient caching for scalable videos in heterogeneous networks,” *arXiv preprint arXiv:1805.11875*, 2018.

[156] A. Sengupta, R. Tandon, and O. Simeone, “Fog-aided wireless networks for content delivery: Fundamental latency trade-offs. 2016,” 2017.

[157] J. Kwak, Y. Kim, L. B. Le, and S. Chong, “Hybrid content caching in 5g wireless networks: Cloud versus edge caching,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3030–3045, 2018.

[158] Y. Wang, X. Tao, X. Zhang, and G. Mao, “Joint caching placement and user association for minimizing user download delay,” *IEEE Access*, vol. 4, pp. 8625–8633, 2016.

[159] W. Jiang, G. Feng, and S. Qin, “Optimal cooperative content caching and delivery policy for heterogeneous cellular networks,” *IEEE Transactions on Mobile Computing*, no. 1, pp. 1–1, 2017.

[160] R. Amer, M. M. Butt, M. Bennis, and N. Marchetti, “Inter-cluster cooperation for wireless d2d caching networks,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 9, pp. 6108–6121, 2018.

[161] Z. Chen, N. Pappas, and M. Kountouris, “Probabilistic caching in wireless d2d networks: Cache hit optimal versus throughput optimal,” *IEEE Communications Letters*, vol. 21, no. 3, pp. 584–587, 2017.

[162] P. Cheng, C. Ma, M. Ding, Y. Hu, Z. Lin, Y. Li, and B. Vucetic, “Localized small cell caching: A machine learning approach based on rating data,” *IEEE Transactions on Communications*, 2018.

[163] M. K. Kiskani and H. R. Sadjadpour, “Throughput analysis of decentralized coded content caching in cellular networks,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 1, pp. 663–672, 2017.

[164] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, “Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution,” *IEEE Communications Magazine*, vol. 51, pp. 142–149, April 2013.

[165] N. Golrezaei, P. Mansourifard, A. F. Molisch, and A. G. Dimakis, “Base-station assisted device-to-device communications for high-throughput wireless video networks,” *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3665–3676, 2014.

[166] M. Polese, R. Jana, V. Kounev, K. Zhang, S. Deb, and M. Zorzi, “Machine learning at the edge: A data-driven architecture with applications to 5g cellular networks,” *arXiv preprint arXiv:1808.07647*, 2018.

[167] S. J. Reddi, *New Optimization Methods for Modern Machine Learning.* PhD thesis, Stanford University, 2017.

[168] A. Lee, P. Taylor, J. Kalpathy-Cramer, and A. Tufail, “Machine learning has arrived!,” 2017.

[169] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[170] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[171] Q. Cheng, R. Niu, A. Sundaresan, and P. K. VARSHNEY, “Distributed detection and decision fusion with applications to wireless sensor networks,” *INTEGRATED TRACKING, CLASSIFICATION, AND SENSOR MANAGEMENT*, p. 619, 2013.

[172] L. Roux, “An application of possibility theory information fusion to satellite image classification,” in *International Workshop on Fuzzy Logic in Artificial Intelligence*, pp. 166–179, Springer, 1997.

[173] Z. Chen and B. Liu, “Lifelong machine learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 10, no. 3, pp. 1–145, 2016.

[174] Y. Zhang, P. Sun, Y. Yin, L. Lin, and X. Wang, "Human-like autonomous vehicle speed control by deep reinforcement learning with double q-learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1251–1256, IEEE, 2018.

[175] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5g using reinforcement learning of space-time popularities," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, 2018.

[176] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, "Content popularity prediction and caching for icn: A deep learning approach with sdn," *IEEE access*, vol. 6, pp. 5075–5089, 2018.

[177] H. Hao, C. Xu, M. Wang, H. Xie, Y. Liu, and D. O. Wu, "Knowledge-centric proactive edge caching over mobile content distribution network," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 450–455, IEEE, 2018.

[178] M. A. Kader, E. Bastug, M. Bennis, E. Zeydan, A. Karatepe, A. S. Er, and M. Debbah, "Leveraging big data analytics for cache-enabled wireless networks," in *Globecom Workshops (GC Wkshps), 2015 IEEE*, pp. 1–6, IEEE, 2015.

[179] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, 2016.

[180] S. E. Hajri and M. Assaad, "Energy efficiency in cache-enabled small cell networks with adaptive user clustering," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 955–968, 2018.

[181] A. Sadeghi, F. Sheikholeslami, A. G. Matrques, and G. B. Giannakis, "Reinforcement learning for 5g caching with dynamic cost," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6653–6657, IEEE, 2018.

[182] S. K. Mishra, P. Pandey, P. Arya, and A. Jain, "Efficient proactive caching in storage constrained 5g small cells," in *Communication Systems & Networks (COMSNETS), 2018 10th International Conference on*, pp. 291–296, IEEE, 2018.

[183] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," *International Journal of Communication Systems*, vol. 31, no. 11, p. e3706, 2018.

[184] L. Lei, L. You, G. Dai, T. X. Vu, D. Yuan, and S. Chatzinotas, "A deep learning approach for optimizing content delivering in cache-enabled hetnet," in *Wireless Communication Systems (ISWCS), 2017 International Symposium on*, pp. 449–453, IEEE, 2017.

[185] Q. Tang, R. Xie, T. Huang, and Y. Liu, "Hierarchical collaborative caching in 5g networks," *IET Communications*, vol. 12, no. 18, pp. 2357–2365, 2018.

[186] L. Mohammed, M. Jaseemuddin, and A. Anpalagan, "Fuzzy soft-set based approach for femtocaching in wireless networks," in *IEEE International Conference on High Performance Computing and Communications.*, IEEE, 2018.

[187] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[188] Y. Tan, Y. Yuan, T. Yang, Y. Xu, and B. Hu, "Femtocaching in wireless video networks: Distributed framework based on exact potential game," in *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–6, IEEE, July 2016.

[189] L. Yang, T. Liu, Q. Hu, S. Liu, and H. Huang, "Empirical analysis on temporal statistics of pairwise contact patterns in dynamic human networks," in *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017 IEEE International Conference on*, pp. 9–16, IEEE, 2017.

[190] L. Chen, Y. Su, W. Luo, X. Hong, and J. Shi, "Explicit content caching at mobile edge networks with cross-layer sensing," *Sensors*, vol. 18, no. 4, p. 940, 2018.

[191] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, pp. 151–160, ACM, 1998.

[192] X. Zhang, Y. Wang, R. Sun, and D. Wang, "Clustered device-to-device caching based on file preferences," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016 IEEE 27th Annual International Symposium on*, pp. 1–6, IEEE, 2016.

[193] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "5g backhaul challenges and emerging research directions: A survey," *IEEE access*, vol. 4, pp. 1743–1766, 2016.

[194] M. D. Bastug, Ejder, Mehdi Bennis, "Proactive Caching in 5G Small Cell Networks," 2015.

[195] S. Manzoor, S. Mazhar, A. Asghar, A. Noor Mian, A. Imran, and J. Crowcroft, "Leveraging mobility and content caching for proactive load balancing in heterogeneous cellular networks," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, p. e3739, 2020.

[196] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *2012 Proceedings IEEE INFOCOM*, pp. 1107–1115, IEEE, March 2012.

[197] M. McNett and G. M. Voelker, "Access and mobility of wireless pda users," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 2, pp. 40–55, 2005.

[198] S. Krishnendu, B. Bharath, and V. Bhatia, "Cache enabled cellular network: Algorithm for cache placement and guarantees," *IEEE Wireless Communications Letters*, vol. 8, no. 6, pp. 1550–1554, 2019.

[199] J. Song, H. Song, and W. Choi, "Optimal content placement for wireless femto-caching network," *IEEE Transactions on Wireless Communications*, vol. 16, no. 7, pp. 4433–4444, 2017.

[200] R. Amer, M. M. Butt, and N. Marchetti, "Caching at the edge in low latency wireless networks," *Wireless Automation as an Enabler for the Next Industrial Revolution*, pp. 209–240, 2020.

[201] X. Li, X. Wang, P.-J. Wan, Z. Han, and V. C. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1768–1785, 2018.

[202] Y. Wu, S. Yao, Y. Yang, T. Zhou, H. Qian, H. Hu, and M. Hamalainen, "Challenges of mobile social device caching," *IEEE Access*, vol. 4, pp. 8938–8947, 2016.

[203] J. Song and W. Choi, "Mobility-aware content placement for device-to-device caching systems," *IEEE Transactions on Wireless Communications*, vol. 18, no. 7, pp. 3658–3668, 2019.

[204] R. Zhang, F. R. Yu, J. Liu, T. Huang, and Y. Liu, "Deep reinforcement learning (drl)-based device-to-device (d2d) caching with blockchain and mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6469–6485, 2020.

[205] T. Zhang, X. Fang, Y. Liu, G. Y. Li, and W. Xu, "D2d-enabled mobile user edge caching: A multi-winner auction approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12314–12328, 2019.

[206] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: Moving from cloud to edge," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 36–42, 2016.

[207] J. Ahn, S. H. Jeon, and H.-S. Park, "A novel proactive caching strategy with community-aware learning in comp-enabled small-cell networks," *IEEE Communications Letters*, vol. 22, no. 9, pp. 1918–1921, 2018.

[208] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[209] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, D. Niyato, and D. I. Kim, "Distributed deep learning at the edge: A novel proactive and cooperative caching framework for mobile edge networks," *IEEE Wireless Communications Letters*, vol. 8, no. 4, pp. 1220–1223, 2019.

[210] Z. Piao, M. Peng, Y. Liu, and M. Daneshmand, "Recent advances of edge cache in radio access networks for internet of things: techniques, performances, and challenges," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1010–1028, 2018.

[211] L. B. Mohammed, A. Anpalagan, A. S. Khwaja, and M. Jaseemuddin, "Mobility-aware latency-efficient cache placement in mobile edge networks," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 778–785, IEEE, 2020.

[212] S. Dash, B. J. Sahu, and N. Saxena, "Social network aware caching for 5g radio access network," *IETE Technical Review*, vol. 34, no. sup1, pp. 52–61, 2017.

[213] A. Quinn, A. Stranieri, and J. Yearwood, "Classification for accuracy and insight: A weighted sum approach," in *Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70*, pp. 203–208, Citeseer, 2007.

[214] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective.* Academic press, 2015.

[215] A. DasGupta, *Probability for statistics and machine learning: fundamentals and advanced topics.* Springer Science & Business Media, 2011.

[216] D. Tran, P. Toulis, and E. M. Airoldi, "Stochastic gradient descent methods for estimation with large data sets," *arXiv preprint arXiv:1509.06459*, 2015.

[217] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[218] T. Van Gestel, J. A. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle, "Benchmarking least squares support vector machine classifiers," *Machine learning*, vol. 54, no. 1, pp. 5–32, 2004.

[219] C. Panagiotakopoulos and P. Tsampouka, “The stochastic gradient descent for the primal l1-svm optimization revisited,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 65–80, Springer, 2013.

[220] T. Kolosova and S. Berestizhevsky, *Supervised Machine Learning: Optimization Framework and Applications with SAS and R.* CRC Press, 2020.

[221] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[222] R. Cheruku, D. R. Edla, and V. Kuppili, *Soft Computing Techniques for Type-2 Diabetes Data Classification.* Chapman and Hall/CRC, 2020.

[223] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[224] H. Chai, Y. Liang, S. Wang, and H.-w. Shen, “A novel logistic regression model combining semi-supervised learning and active learning for disease classification,” *Scientific reports*, vol. 8, no. 1, pp. 1–10, 2018.

[225] L. B. Mohammed, A. Anpalagan, A. S. Khwaja, and M. Jaseemuddin, “Semi-supervised learning with self-training for cache placement in mobile edge networks,” in *30th Biennial Symposium on Communications*, IEEE, 2021.

[226] X. Zhu and Z. Ghahramani, “Learning from labeled and unlabeled data with label propagation,” 2002.

[227] L. Tang, B. Tang, L. Tang, F. Guo, and J. Zhang, “Reliable mobile edge service offloading based on p2p distributed networks,” *Symmetry*, vol. 12, no. 5, p. 821, 2020.