8-2019

Edge Caching for Small Cell Networks

Md Ferdous Pervej
Utah State University

EDGE CACHING FOR SMALL CELL NETWORKS

by

Md Ferdous Pervej

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Rose Qingyang Hu, Ph.D. Major Professor	Don Cripps, Ph.D. Committee Member
Ziqi Song, Ph.D. Committee Member	Richard S. Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2019

ABSTRACT

Edge Caching for Small Cell Networks

by

Md Ferdous Pervej, Master of Science

Utah State University, 2019

Major Professor: Rose Qingyang Hu, Ph.D.
Department: Electrical and Computer Engineering

Lately, edge caching has been applied extensively into next-generation wireless communication systems to improve bandwidth utilization and to reduce latency. One of the most challenging tasks in such a platform is to predict what contents to cache in the nearby nodes so that high caching content availability can be achieved. In most of the legacy systems, a static estimation on the content demands is made by considering Zipf distribution. However, modeling this enormously critical parameter in such a way may not be sufficient enough to capture the dynamic behaviors of the content popularity. Furthermore, users make requests for contents based on their own preferences. Therefore, modeling dynamic user preferences for the contents makes content caching more challenging. However, to forecast user preferences ahead of time, one can cache the probable contents that are likely to be requested. Motivated by this, to capture the dynamic preference of the users, this thesis proposes a long short-term memory (LSTM) based time series model. Thanks to LSTM [1], the temporal dynamics of the users' preferences can be well captured for the available content in the content library. Furthermore, to incorporate the goal of edge caching in a content delivery network (CDN), this thesis considers collaborations among different nodes residing (a) in the same cell as well as (b) in the same cluster. Based on the time-ahead forecast, the optimization problem is formulated to minimize the cost of content sharing among these

nodes. Besides, unlike the legacy models which stores the contents following homogeneous caching placement strategy, this thesis exploits both heterogeneous caching placement and homogeneous caching placement strategies. However, the formulated problem is non-convex and contains a large number of optimization parameters. Therefore, it is extremely hard to get an optimal solution for a distributed platform like the proposed one. This thesis, thus, proposes necessary greedy and meta-heuristic algorithms to jointly optimize and solve the problems while achieving reasonable sub-optimal solutions. By using the mathematical analysis and simulation results, this thesis also validates that the heterogeneous caching placements strategy outperforms the homogeneous caching placement strategy. Furthermore, the proposed caching algorithms perform better than other available baseline caching schemes.

(107 pages)

PUBLIC ABSTRACT

Edge Caching for Small Cell Networks

Md Ferdous Pervej

An idea of storing contents, such as media files, music files, movie clips, etc. is simple yet challenging in terms of required effort to make it count. Some of the benefits of pre-storing the contents are reduced delay of accessing/downloading a content, reduced load to the centralized servers and of course, a higher data rate. However, several challenges need to be addressed to achieve these benefits. Among many, some of the fundamentals are limited storage capacity, storing the right content and minimizing the costs. This thesis aims to address these challenges. First, a framework for predicting the proper contents that need to be stored to the limited storage capacity is presented. Then, the cost is minimized considering several real-world scenarios. While doing that, all possible collaborations among the local nodes are performed to ensure high performance. Therefore, the goal of this thesis is to come up with a solution to the content storing problems so that the network cost is minimized.

To my family members, for their unconditional support and sacrifices!

ACKNOWLEDGMENTS

Finally, I am writing this acknowledgment! I have never thought coming to this point would be any easier, yet the efforts it took were way beyond my realization. There were many nights when I felt I would never have finished the works. I am entirely indebted to many living human-like angels who have been supporting me like a true - friend, brother, and guardian. A simple 'thank you' will not suffice to express my gratitude to them. Yet again, from the bottom of my heart, I would like to say thanks to those human-looking angels.

I am genuinely thankful to my supervisor - Prof. Rose Qingyang Hu. Without her direction, I would have stuck in the wrong routes. My heartiest thanks to her for her time and patience with this work. I would also sincerely like to thank Dr. Ziqi Song and Dr. Don Cripps, for serving as my thesis committee members. My sincere thanks to Prof. Gunther for the occasional discussions about solving optimization problems. I would also like to thank Prof. Moon for teaching his classes so brilliantly. Taking the courses he offered were some of my best experiences - though I suffered a lot, during my entire stay in USU. Undoubtedly, those courses built the necessary mathematical foundation of my wireless communication research. Thanks to all past and current members of the Communications Network Innovation Lab. A special thanks to Qun for tolerating my naive questions. This work was funded by the NSF grants ECCS-1308006, NeTS- 1423348, EARS- 1547312. My sincere gratitude to the funding agencies, your supports led me to finish this work timely. Thanks to Tricia and Diane from the ECE Department, you guys were super helpful.

Dr. Le Thanh Tan deserves more than my sincere gratitude. From the start of this work, he has been supporting me like a real guardian. Thank you, Dr. Tan, you have helped and rescued me in many possible ways. The helpful discussions along the way indeed led me to finish this work. I have learned so many things from you for which I am truly indebted to you. I also sincerely convey my heartfelt thanks and gratitude to all current and past Bangladeshi students in USU for their loves and supports along the way. Thanks to Dr.

Mehedi Hasan for the helpful discussions, especially for the coffee in each of those informal sessions. Thanks to Shaju Saha, Md Munibun Billah, Abrar Zahin, Md Abdullah Al Sarfin, Nazmus Sakib, Rakin Muhammad Shadab, for your countless help and supports.

Last but not least, I would like to thank my family members who are miles apart yet supporting me from the beginning. Living without any of you have not been easy so far. Your inspirations and loves have brought me here today. Without you and your sacrifices, it would not have been possible to travel this long route. Thanks to my wife - Mst Sujtanta Razia, for taking care of our child - Ayan Ferdous. Raising a child is in no way an easy task to do. You have proved me wrong! You are just doing fine. Sometimes, I wonder, was it harder to finish this thesis or taking care of Ayan in my absence!

Md Ferdous Pervej

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ACRONYMS

AI	Artificial Intelligence
ADMM	Alternating direction method of multipliers
CDN	Content delivery network
D2D	Device to device
HD	Half duplex
EM	Expectation maximization
HetNet	Heterogeneous network
HPPP	Homogeneous Poisson point process
IBFD	In-band full-duplex
ICN	Information centric network
LSTM	Long short term memory
MAB	Multi-arm bandit
MBS	Macro base station
ML	Machine learning
MLE	Maximum likelihood estimation
MZipf	Mandelbrot-Zipf distribution
pLSA	Probabilistic latent semantic analysis
PPP	Poisson point process
PSO	Particle swarm optimization
QCQP	Quadratically constrained quadratic program
RL	Reinforcement learning
RNN	Recurrent Neural Network
sBS	Small base station
SDR	Semidefinite relaxation
SINR	Signal-to-interference-plus-noise ratio
SPMAB	Single player MAB
UE	User equipment
QoS	Quality of service

CHAPTER 1

Introduction

Wireless communications have experienced dramatic changes over the past few decades. As the wireless user penetration keeps increasing and the new applications have been consistently emerging, wireless technologies have evolved from one generation to the next generation to provide more and more capacity and better and better user quality of service (QoS). The general thrust for these changes is the crave for enhanced data rate, latency, energy efficiency, and QoS. While the existing wireless network has already ensured auspicious performance delivery, the new and challenging-to-deal-with demands on capacity and other performance have never ceased to emerge [2]. Furthermore, the inadequacy of the existing technology has become apparent with the inclusion of the Internet of things (IoT) [3], leading us to embrace new technologies on top of the existing ones. To guarantee the required performances, researchers are, therefore, in a toiled search for new technologies to incorporate in the fifth generation (5G) and beyond.

Among many other imposing ideas, moving towards user-centric network infrastructure instead of the traditional base station (BS) centric infrastructure is a promising one [4, 5]. A user-centric network platform can significantly reduce delay, computation, and energy consumption and enhance the utilization of the much-needed spectrum [6–8]. Therefore, the concept of edge caching becomes a desirable approach. This allows a user to retract requested contents from local nodes instead of accessing the distant cloud servers. Furthermore, it also more efficiently utilizes the network bandwidth and reduces the congestion on the access to the centralized cloud server.

The main intuition of edge caching is to store the contents closer to the user end so that if required, a user can directly extract the requested content from the nearby node without accessing far way centralized cloud server. This increases the data rate, which is, among many others, perhaps the significant one that researchers have always desired for [9].

Besides, video traffic has been deemed one of the main reasons for wireless mobile data traffic volume escalation, imposing tremendous pressure to expand the network capacity [10–12]. One possible solution to this bottleneck is to store the popular video content in the local nodes to alleviate the bandwidth demand in the centralized network segments. Thus, during rush hours, mobile video traffic can be efficiently handled with the inclusion of edge caching in a content delivery network (CDN) platform. Furthermore, in various mission-critical delay-sensitive applications such as online surgery, edge caching can perform much better than the cloud caching approach due to the significant reduction on delay [13].

It has been a while since content caching has been explored in the information-centric network (ICN) [14–16]. As the essence of caching is to store popular contents at the local nodes, there has to be a way to measure the popularity of the contents. In other words, the selection of the to-be-stored contents has to be accurate. Furthermore, the volatile natures of the users' preferences make the problem much more challenging. Therefore, before making a decision, an extensive study of the users' tastes, both locally and globally, have to be performed. Furthermore, in a distributed platform, learning popularity profiles using conventional tools may not be feasible. The system administrator may require to predict what content the users will need in the future ahead of a long time. Therefore, instead of modeling the content popularity such as the Zipf type distribution, sophisticated machine learning (ML) algorithms are likely to perform better.

Having found a way to measure the dynamic content popularity and user preferences, the next thing to consider is the distribution of the cache-enabled nodes. Distribution of these edge nodes may play a critical role in CDN. As the existing network is already dense, before making it more thicker - by deploying cache enabled devices, a thorough study could be beneficial. Furthermore, an integrated platform may change the existing performance metric, such as - coverage/outage probability, energy efficiency, area spectral efficiency. Therefore, performance analysis should be made jointly using both stochastic geometry and proper networking protocol.

Then, there come the questions - what content to store and at what node to store

it? That is, essentially, the placement of the contents. Since the cache storages of the edge nodes are limited, given the popularity measures, a sophisticated way is required to place the contents at these nodes. Numerous studies have shown that the optimal caching problem is NP-hard [17–19]. Furthermore, a long term caching probabilities may help the system administrator to understand the performance of the network evaluating different performance metrics. Also, in general, two caching placements strategies, namely - heterogeneous caching placement and homogeneous caching placements, are widely used in a CDN [20]. In the former one, the stored contents at different nodes may not be the same while, in the later one, similar copy of the contents are stored at all of the identical tier nodes.

Moreover, after placing the contents at the edge nodes, if a collaboration is established so that the cache-enabled nodes also can share content with each other, while this increase the system performance, the complexity also escalates. It will be beneficial if the content placements are performed, reminding such collaboration. However, the performance of the overall network may vary on what strategy the system administrator is following for the content placement, content delivery, and collaboration phase.

The **organization** of this thesis is as follows - Chapter 2 discusses the comprehensive literature reviews and significant contributions of this thesis. Chapter 3 presents the first fundamental technical part of this thesis by modeling dynamic user behaviors using long short-term memory (LSTM) model and aiming to minimize content sharing cost. Then, in Chapter 4, a more practical system model with real-world communication scenario has been considered for analyzing the performance of edge caching in a real network. Finally, Chapter 5 concludes this thesis followed by the probable future research directions.

CHAPTER 2

Literature Review and Contributions

2.1 Literature Review

In [21], the authors considered both coded and uncoded cases for caching content at the helper's nodes to minimize the downloading time for content. The authors, however, did not consider the collaboration among the helper nodes. In [22], to learn the popularity of content over time, the authors proposed a multi-arm bandit (MAB) framework based model that only considers a single BS. If collaboration among BSs is not counted, more required contents will have to be fetched from the far away cloud. In [23], without considering any collaboration among the BSs, a reinforcement learning (RL) based system model was proposed to determine the optimal caching placement for both coded and uncoded cases showing a way of maximizing the instantaneous reward for an individual user. Similar to [22], this is, therefore, not idle for a CDN platform. Making collaboration among different local nodes such as - users and BSs is more practical and beneficial in such a platform.

The authors of [24] proposed a more dynamic approach. In this paper, the authors considered that the base stations have different cache sizes and considering collaboration between the BSs, they have proposed single player MAB (SPMAB) and multiple players MAB (MPMAB) approach to maximizing the caching rewards. For the SPMAB approach, they formulated semidefinite relaxation (SDR) based quadratically constrained quadratic program (QCQP) problem, which is more complex and increases system complexity. In the case of the MPMAB approach, the authors considered the alternating direction method of multipliers (ADMM) problem formulation and shown how to maximize the caching rewards. However, in comparison to the proposed system model of [24], this thesis considers a device to device (D2D) based CDN platform. Besides, it is doubtful that the implementation of such a model may not be computationally efficient due to its complexity.

Perhaps [25, 26] are more related papers with the proposed framework in this thesis. However, the proposed approach in [25] is not computationally efficient for a vast distributed network. In contrast to [25], the observation time-slots and then predicting based on that in this thesis are, presumably, more practical. In any real networks, it is impractical to update the cache storage in each time-slot if not nearly impossible. In such a system, replacement of the content occurs when the network has lower traffic to ensure no interruption in service [27–29]. The authors in [7, 8, 30] have shown some promising techniques for video caching and edge computing. In [7], the authors have shown a brilliant idea of incorporating video caching and processing for mobile-edge computing. However, in these papers, the content popularity is considered, either as known or can be modeled based on least recently used contents only.

Caching gain has been studied while modeling individual user preference in [31]. The authors have considered global content popularity and personal preference as two different concepts. Furthermore, the authors have modeled this using an expectation-maximization (EM) algorithm. However, the authors only consider estimating for a single time slot. In other words, following their notion, the popularity and preference can only be modeled for a single time slot. They do not present a time series alike analysis. Besides, the system model adopted in the paper is simple, where the users can only collaborate with the neighbors. Consideration of collaboration between base stations or access points has not been considered in their work.

Caching policy and cooperative distance have been designed in [32, 33] considering clustered D2D networks. While the authors have shown some brilliant concepts for the caching policy design aiming to maximize (a) energy efficiency and (b) throughput, they only consider collaboration among the D2D users. In their system model, they have divided a squared region into multiple squared cells and deploy D2D users in the area. However, their collaboration scheme is straightforward in a sense that, in real-world CDN deployment, there should be multiple access points (which can be considered as the low powered sBS or relay) underlying users. Besides, the authors in [32, 33] model the content popularity and

user preferences following a Mandelbrot − Zipf distribution (MZipf) type distribution [34]. However, this thesis intends to capture the temporal dynamics of the users considering a historical dataset. Therefore, apart from a more sophisticated system model, the works presented in this thesis, is distinct in terms of user preference modeling, than the authors' considerations.

Another clustered based system model has been considered in [35, 36]. While both of the works, form the same research group, have considered collaboration among the clustered cell users, their method of clustering is straightforward. The authors have simply divided a macro base station (MBS) into multiple clusters where users are distributed. Users in such a group can collaborate with the neighbors. Naming their works as inter-cluster cooperation, the authors have presented a delay (latency) and a throughput per requests analysis in these papers. No consideration of low powered wireless access points, relay or sBS have been considered in their works. A more practical collaborative caching model is presented in [37]. Here, the authors have considered collaboration among different participating base stations. However, their framework does not review any underlying D2D communication, meaning that they are only collaborating among the base stations. Using Zipf distribution, the authors have modeled the content popularity followed by a Poisson process governed request model. However, although the network traffic pattern follows the Poisson process, the request for specific content may not necessarily follow this distribution.

Furthermore, the authors in [38] have adopted a collaboration based caching model in a heterogeneous network model. While some brilliant concept of relaying and collaboration have been introduced to maximize the network throughput, only homogeneous caching placement strategies have been incorporated. Besides, while they introduced low-powered relay for the throughput maximization, their collaboration schemes only limit to D2D users. A mobility aware probabilistic edge caching approach has been explored in [39]. While the proposed model considers a noble idea of collaboration by considering spatial node distribution and user mobility, again, the proposed solution is only for the homogeneous caching case.

2.2 Research Contributions

In contrary to these works, to reflect the proper picture of users' choices, this thesis, first in Chapter 3, adopts a LSTM based model to predict the users' preferences sequentially. The underlying assumption made in this chapter is - given some historical observation, what content the user will request in the future unseen time slot. Furthermore, here, a fixed number of users and small base stations (sBS) are assumed to be distributed uniformly. A fixed number of contents are also considered in the content catalog in this chapter. Users, in the same cell, can collaborate with each other while the base stations in the same clusters also can collaborate with each other. Then using the historical dataset, the prediction has been made for future time slots. Note that the preferences of the user vary dynamically over time. Thus, the user of LSTM allows capturing the temporal dynamics very well. Furthermore, instead of only considering the global content popularity while modeling the user preferences using the Zipf distribution, a real scenario is considered where the choice of a user does not depend on the global content popularity or other users preferences.

Using the prediction and forecast from the proposed LSTM model, then first the content placement - at the cache-enabled nodes, is performed. Several noble and practical algorithms for this content placement are also introduced in Chapter 3. Note that the algorithms help to place the contents at the local nodes for all possible future time slots for which the forecast has been performed with the LSTM model. The number of time slots is not fixed, and the proposed model has the flexibility to determine this duration based on the choice of the system administrator. Note that the algorithms work for both the heterogeneous and homogeneous caching placement cases. This chapter also shows the comparison of performances between heterogeneous caching placement case and homogeneous caching placement case.

After performing the caching placements, for both heterogeneous and homogeneous cases, the long term probabilities of storing the contents at the local nodes are calculated based on their respective equations. Finally, the performances of the proposed algorithms are compared with similar existing and baseline algorithms in Chapter 3.

Furthermore, a spatial real network deployment may need to rethink about the collaboration among nodes. That is, if a particular node does have a requested content, but not in the communication range of the requester; this is essentially the case of a cache miss. While deploying a real network in a 2D plane, the performance of edge caching may vary. Hence, in Chapter 4, an actual network model has been considered while performing edge caching. Unlike the existing models, a practical heterogeneous request model and heterogeneous caching placement model have been considered in this chapter. Due to a large number of system parameters, it is extremely difficult to come up with the optimal solution. Thus, artificial intelligence (AI) based solution is also proposed in Chapter 4 to efficiently solve the original problem.

In summary, the contributions made in this thesis are listed as follows:

1. A LSTM model is used to capture short-temporal user dynamics for forecasting user preferences ahead of time.

2. Users' preferences are forecasted multi-time scales ahead using the proposed LSTM model.

3. To fully exploit the advantages of edge caching, a collaborative communication framework, in which different nodes in the same cluster can share contents among each other, is proposed.

4. The optimization problems are formulated to minimize the content sharing costs under the constraints of limited and dynamic storage capacities at both the users and the BSs for both heterogeneous caching placement and homogeneous caching placement cases.

5. The content sharing costs are analyzed, and both greedy and meta-heuristic collaborative edge caching algorithms are developed to configure the parameters of caching placement.

6. Numerical results are presented to illustrate the performance of the proposed algorithms by using the optimal parameter configuration for the caching strategy.

CHAPTER 3

User Preference Learning Aided Collaborative Edge Caching for Small Cell Networks

This chapter focuses on user preference modeling first. Then, the optimization problem is formulated to minimize the cost function while ensuring collaboration among different edge nodes. Unlike the existing direct preference modeling based on Zipf type distribution, this chapter considers the temporal dynamics of the user preferences as well as its activity level by incorporating long-short-term-memory (LSTM) based model.

3.1 System Model

In this section, first, the system model is introduced briefly. A concise description of the node's distributions and various assumptions made for the contents in the content catalog is presented next. Then, different content access protocols that need to be used while performing the system performance evaluation are introduced. Finally, a concise explanation of the purpose of the problem formulation is presented.

3.1.1 Local Node Distributions

A set of D2D users $\mathcal{U} = \{i\}$, where $i \in \{1, 2, \ldots, U\}$, are distributed in the coverage area of some sBSs. Considering a clustered based system model, it is assumed that there exist a small number of small base stations (sBS) in a single cluster[1]. In each cluster, there are an equal number of sBSs with equal cache storage capacity. Here, $\mathcal{B} = \{j\}$, where $j \in \{1, 2, \ldots, B\}$, and C_b represent the set of sBS and its the cache storage size, respectively. For simplicity, it is assumed that each sBS cell has an equal number of users uniformly distributed in its coverage region. A D2D requesting node and the serving sBS are denoted as the tagged D2D node and the tagged sBS[2], respectively. An equal number of sBSs are also uniformly distributed in all clusters. Furthermore, for successful communication

[1] By clustering, a group of cells with no common frequency within the group is referred.

[2] Throughout this chapter, the name serving sBS and tagged sBS are used interchangeably.

among different nodes, the channel quality has to satisfy a certain pre-defined threshold. For simplicity, all D2D nodes in a single cell are assumed to be in the communication range of each other. It is also assumed that all the D2D nodes have the same cache size of $\mathcal{C}_d$. Moreover, all the D2D users are in the communication range with a serving sBS.

3.1.2 Content Catalog

A fixed $|\mathcal{F}|$ number of contents, denoted by $\mathcal{F} = \{k\}$, where $k \in \{1, 2, \ldots, F\}$, are considered in the content catalog. Furthermore, the size of all content is considered to be equal, denoted by S_f. In reality, this is widely used. Even if the content sizes are not similar, the contents can be divided into equal chunks of packets and then those equal size chunks can be stored into the edge nodes cache storage [40, 41]. In the proposed model, the users are restricted to make requests for only those contents that are in the content catalog.

3.1.3 Content Access Protocols

The primary goal is to deliver the requested contents from the local cache as much as possible. The proposed system model thus can contribute significantly to latency critical real-time communication, as it can avoid unnecessary computations and delays with enhanced resource utilization.

The Proposed Communication Flow

If a tagged user needs to access the desired content, before sending the request to other nodes, it first checks its own cache storage. The tagged user sends the content request to the neighboring D2D nodes that are residing in the same cell and are within its communication range if the requested content is not found in its self cache storage. If the content is available in one of the neighboring D2D nodes, the content can be directly served from that node to the tagged user. If none of the D2D nodes has the requested content, the request is then forwarded to the serving BS, which delivers content to the tagged user if it is found its storage. If the requested content does not exist in the serving BS's cache, the serving BS forwards the request to the neighboring BSs residing in the same cluster. If the content is

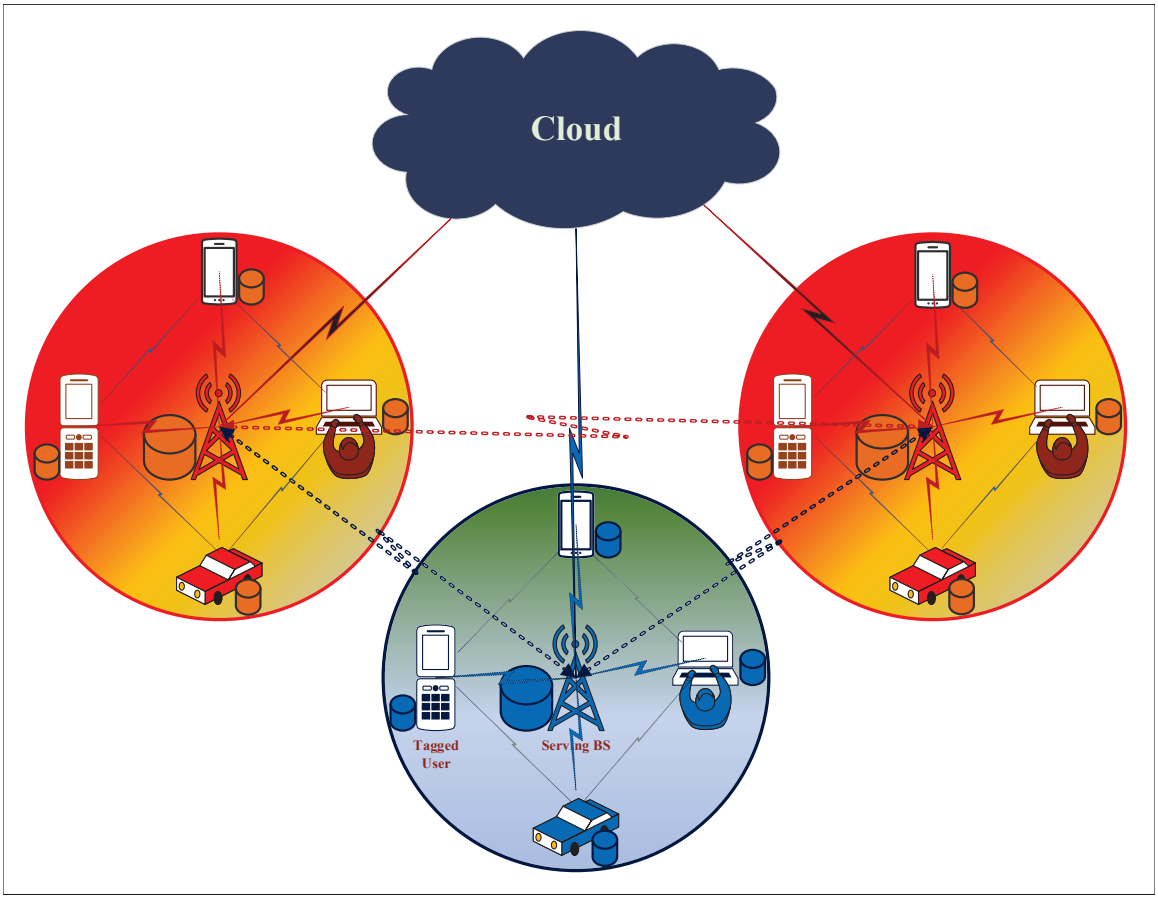


Fig. 3.1: System Model for Collaborative Caching

not available in any of these local stores, it can be downloaded from the cloud, which is considered time and bandwidth consuming.

3.1.4 Problem Definition

The ultimate aim is to model the problem in a twin scale fashion. First, the dynamic content preferences of the users are modeled. Here, for clarification, it is assumed that complete historical data is available on hand. Based on historical data, a time-series analysis is performed to model the dynamic user preference for the contents. Note that 'content popularity' and 'user preference' for a particular content are considered as two different terms. While content popularity can be modeled based on the frequency count on a global scale, it is not that simple to model the user preference for particular content. Though the

Table 3.1: List of Symbols

Symbol	**Description**
$\mathcal{F}$	Content catalog / library
$\mathcal{U}$	Set of user nodes (D2D Nodes)
u_i	i^{th} D2D Node, u_0 represents tagged node
$\mathcal{B}$	Set of BSs
b	Element of the BS set, b_0 is the serving BS
C_b	Cache storage of BS b
$\mathcal{C}_d$	Cache storage of D2D node d
$\rho_{f_k}^{u_i}$	Probability that user u_i requests content f_k
$a_{f_k}^{u_i}$	Probability that user u_i stores content f_k
$\eta_{f_k}^{b}$	Probability that BS b stores the content f_k
ϕ_l	Content sharing cost from node l

global content popularity may affect a user's inclination to watch/view a content, in reality, it largely depends on the personal choice of the user. Therefore, the noble intention of this thesis is to model the per-user content preference in a dynamic way. Further, the aim is to predict this in a time series manner. The majority of the legacy system model does not consider modeling this in a dynamic time-series analysis framework. Motivated by this, given the historical data of multiple time slots, the goal is to predict which content could be requested by the user in the next time slot considering heterogeneous user preferences.

Based on the prediction, the next goal is to model the caching policy. The objective is to store the most probable to-be-requested contents in future time slots before the actual requests occur. Recall that the system model considered in this thesis aims is to store the contents at two levels - D2D nodes and sBSs, and make collaborations among these local nodes. The proposed model is thus highly practical to that of a CDN platform. Based on the model of the actual request, the optimization model is then presented with the final goal of minimizing the content sharing costs.

In the next section, the prediction model is presented. Fig. 3.1 and Table 3.1 represent the proposed system model and symbols used in this thesis, respectively. Throughout this thesis, bold and uppercase ($\mathbf{A}$) represents a matrix, bold and lowercase ($\mathbf{a}$) represents a vector, regular symbol or letter (a or θ) represents a scalar-valued symbol.

3.2 Dynamic User Preference Prediction

This is the first technical part of this chapter. In this section, to avoid confusions of cross-domain nomenclatures, first, different terms that are to be used throughout this thesis are defined. Then the proposed approach for modeling dynamic user preferences is presented.

3.2.1 Motivation for Introducing Different Terms

Modeling content popularity and heterogeneous preferences in a system model as presented in Fig. 3.1 is always challenging. If content popularity is only taken into account, it dynamically varies over time and locations, let alone user preferences. Motivated by this, to quickly learn different aspects, different terms are clearly defined in the following. Before diving into the details, the meaning of these terms are introduced as follows:

Content Popularity

In a general sense, content popularity defines the fondness of a content. More formally, content popularity is the probability distribution of the content, $k \in \{\mathcal{F}\}$, which expresses the global demand of that content. In other words, it defines the probability distribution of the requests coming from all the users. If only a small geographic region, such as a small cell, is considered this may also be termed as the local/regional popularity.

In most of the legacy networks, Zipf distribution has been widely used to model the content popularity [21, 42]. The probability mass function (pmf) of this Zipf distribution is represented by (3.1).

$$\mathrm{P}_{f_k}(k = i) = \frac{i^{-\gamma}}{\sum_{i=1}^{F} i^{-\gamma}}, \tag{3.1}$$

where F denotes the number of content, while γ denotes the skewness of the content popularity. This distribution depends on the parameter γ i.e., the skewness. If γ is large, only a few contents will be requested most of the time, while a low valued γ means that the users will prefer a large number of contents.

Content Preference of Users

This is the preference of an individual user. Rather than considering the fondness of a content to all users, when per-user basis fondness is considered, the term is denoted as the content preference of that user. More formally, content preference of a user is the term which defines the probability of requesting a content f_k by a user u_i given that the user actually makes a request. Let us define this as follows:

$$\mathbf{q}_{f|u_i}(t) = \left[q_{f_1|u_i}(t), q_{f_2|u_i}(t), \ldots, q_{f_F|u_i}(t)\right], \quad \forall i \in \{\mathcal{U}\}, \tag{3.2}$$

where $q_{f_k|u_i}(t)$ represents the probability that user u_i request content f_k at time slot t given that she actually makes a request.

It is readily comprehensible that at a time slot t, $\sum_{k=1}^{F} q_{f_k|u_i}(t) = 1$. Furthermore, this is equivalent to $q_{f_k|u_i}(t) \triangleq P_t(f_k|r_i)$, where $P_t(r_i(t))$ denotes the probability of making a request by user u_i at time slot t. In the time slot t, the user preference can be stacked in a matrix for the ease of convenience as follows:

$$\mathbf{Q}_{u|f}^{U\times F}(t) = \begin{bmatrix} \mathbf{q}_{f|u_1}(t) \\ \mathbf{q}_{f|u_2}(t) \\ \vdots \\ \mathbf{q}_{f|u_U}(t) \end{bmatrix} = \begin{bmatrix} q_{f_1|u_1}(t) & q_{f_2|u_1}(t) & \ldots & q_{f_F|u_1}(t) \\ q_{f_1|u_2}(t) & q_{f_2|u_2}(t) & \ldots & q_{f_F|u_2}(t) \\ \vdots & \vdots & \ldots & \vdots \\ q_{f_1|u_U}(t) & q_{f_2|u_U}(t) & \ldots & q_{f_F|u_U}(t) \end{bmatrix}. \tag{3.3}$$

Activity Level of User

As the name suggests, the probability that a user sends a content request is denoted as the activity level of her. This is denoted by $r_i(t) \triangleq P_t(r_i(t))$, where $\sum_{i=1}^{U} P_t(r_i(t)) = 1, \forall u_i \in \{\mathcal{U}\}$. Furthermore, this can be staked into a vector for all users which is denoted as $\mathbf{r}(t) = [r_1(t), r_2(t), \ldots, r_U(t)]^T$.

Before deriving the mathematical equation of $r_i(t)$, for the ease of describing different terms, the following matrix is denoted as the user-content matrix for the time slot t.

$$\mathbf{N}_{uf}^{U\times F}(t) = \begin{bmatrix} \mathbf{n}_{u_1,f}(t) \\ \mathbf{n}_{u_2,f}(t) \\ \vdots \\ \mathbf{n}_{u_U,f}(t) \end{bmatrix} = \begin{bmatrix} n_{u_1,f_1}(t) & n_{u_1,f_2}(t) & \dots & n_{u_1,f_F}(t) \\ n_{u_2,f_1}(t) & n_{u_2,f_2}(t) & \dots & n_{u_2,f_F}(t) \\ \vdots & \vdots & \dots & \vdots \\ n_{u_U,f_1}(t) & n_{u_U,f_2}(t) & \dots & n_{u_U,f_F}(t) \end{bmatrix}, \tag{3.4}$$

where $n_{u_i,f_k}(t)$ denotes the number of incidents in which user u_i has requested content f_k in time slot t. Then the following can be written accordingly to represent the total number of requests made by all users for all contents in that time slot.

$$q(t) = \sum_{i=1}^{U}\sum_{k=1}^{F} n_{u_i,f_k}(t). \tag{3.5}$$

From the user-content matrix in equation (3.4), the following two equations are also introduced for the ease of explanations. Let $n_{u_i}(t)$ denotes the summation of the content requests made by a user u_i for all the content $f_k \in \{\mathcal{F}\}$ in time slot t. Also, let $n_{f_k}(t)$ denotes the total number of requests for a particular content f_k in that time slot by all users $u_i \in \{\mathcal{U}\}$. Then the following can be written:

$$n_{u_i}(t) = \sum_{k=1}^{F} n_{u_i,f_k}(t), \quad \forall i \in \{\mathcal{U}\}. \tag{3.6}$$

$$n_{f_k}(t) = \sum_{i=1}^{U} n_{u_i,f_k}(t), \quad \forall k \in \{\mathcal{F}\}. \tag{3.7}$$

Then, the activity level of a user is expressed by the following equation:

$$r_i(t) = \frac{n_{u_i}(t)}{q(t)}, \tag{3.8}$$

where $q(t) = \sum_{i=1}^{U}\sum_{k=1}^{F} n_{u_i,f_k}(t)$. Therefore, the activity level is introduced to know the actual load coming from each user.

Furthermore, the conditional probability that a user request content f_k - given that she actually makes a request, is presented by the following equation.

$$q_{f_k|u_i}(t) = \frac{n_{u_i,f_k}(t)}{n_{u_i}(t)}. \tag{3.9}$$

Notice that from equations (3.8-3.9), at that time slot, the joint probability that a user u_i actually makes a request and the requested content is f_k is expressed as follows:

$$q_{u_i,f_k}(t) = r_i(t)q_{f_k|u_i}(t) \triangleq P_t(r_i(t))q_{f_k|u_i}(t). \tag{3.10}$$

From equation (3.10), it is clear that two things, namely - the activity level of a user, followed by which content she requests, need to be predicted to know the successful joint probability for all users. Furthermore, considering the complete data is known, using the definitions, the following interrelation among the above terms can be observed:

$$\sum_{i=1}^{U} r_i(t)q_{f_k|u_i}(t) = p_{f_k}(t), \tag{3.11}$$

where $p_{f_k}(t)$ essentially represents the global content popularity confined on that region for time slot t.

Proof. The RHS of (3.11) is defined as $RHS = \frac{n_{f_k}(t)}{q(t)}$. From (3.8) and (3.9), the LHS of (3.11) can be derived as

$$LHS = \sum_{i=1}^{U} \frac{n_{u_i}(t)}{q(t)} \frac{n_{u_i,f_k}(t)}{n_{u_i}(t)} = \sum_{i=1}^{U} \frac{n_{u_i,f_k}(t)}{q(t)} = \frac{n_{f_k}(t)}{q(t)}. \tag{3.12}$$

So the proof is completed. ■

3.2.2 Modeling Dynamic User Behavior

Notice that one may try to estimate the parameter that governs the distributions using

maximum likelihood estimation (MLE). However, if the complete data[3] of (3.4) is not known, the direct use of MLE is not applicable. In that particular case, one may estimate the complete data before applying MLE. Thanks to probabilistic latent semantic analysis (pLSA), one may use expectation maximization (EM) algorithm to estimate the data as shown in [31]. However, only predicting the data, or the parameter of the distributions that govern the data, based on historical data set availability may not suffice the requirements of content caching. Another critical aspect of this is that the distribution of the user requests model is not known. In reality, the system administrator only stores, evicts, or re-flash the stored contents after certain times depending on the performances and traffic flows. Therefore, predicting the user preferences and content popularity for the future periods are inevitable for a reasonable and practical CDN design. Thus, the well-known concepts of machine learning (ML) is used in this chapter for predicting these critical parameters for the future time slots on a time series manner.

Essentially, the interest lies in predicting the user-content matrix for future time slots. That is, given the historical data for $t \in \{1, 2, \ldots, N\}$ time slots, the aim is to predict $\hat{\mathbf{N}}_{uf}^{U\times F}(N+1)$. Before diving into the proposed LSTM based prediction model, first, different terms related to LSTM models are introduced concisely. Then the proposed prediction model is presented.

Long Short-Term Memory (LSTM) Model

To avoid the long term dependencies in the recurrent neural network (RNN), LSTM - a special kind of RNN is used in this chapter. Note that RNN is widely used to predict sequential data. It takes historical data as input and predicts what data will come in the next step [43]. However, it is extremely tough to train the model and learn from it when long term dependencies come into the picture. Furthermore, the well-known problems of vanishing gradients and exploding gradients can significantly impact the performance of RNN if the number of time steps is larger due to necessary backpropagation [44]. To

[3]The complete data also contains the latent class from which the content comes from. Besides, the data could have some missing points.

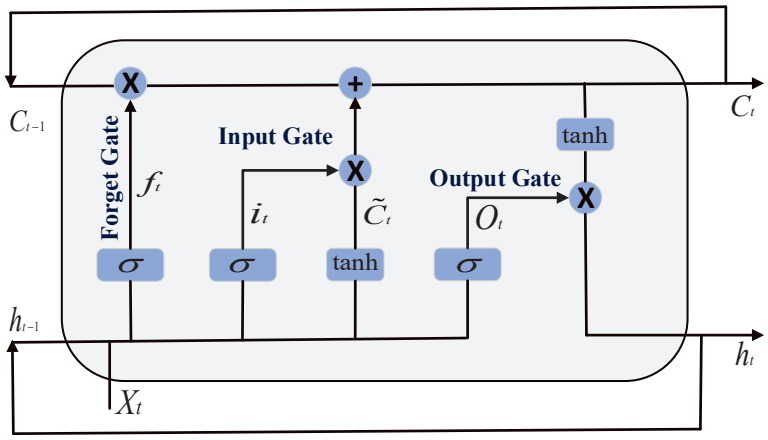


Fig. 3.2: LSTM Model

overcome these problems LSTM is modeled magnificently by truncating the gradient where it will not affect [1].

A simple schematic diagram is presented in Fig. 3.2. As it can be seen from the figure, there are three simple gates, namely - (1) forget gate, (2) input gate and (3) output gate, while a cell state C_t governs the LSTM model. The functionalities of the cell state and each of these gates are briefly discussed in the following.

First of all, the previous output h_{t-1} and current input value x_t are taken as the present input. Note that this is a sequential model i.e., the output of the previous time step $t-1$ has to be fed at the current step t to train the model. In the forget gate, these values are taken as the gate's input which then produces the output f_t. Mathematically, it can be expressed as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f), \tag{3.13}$$

where W_f and b_f are weight matrix and bias vector, respectively. Note that these parameters are learned during the training process.

In the input gate, first the previous output h_{t-1} and current input X_t are passed through a Sigmoid and a tanh functions. This produces the output denoted by i_t and $\tilde{C}_t$,

respectively.

$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i), \tag{3.14}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, X_t] + b_c), \tag{3.15}$$

where W_i, W_c are weight matrices and b_i, b_c are bias vectors in equations (3.14-3.15).

Multiplication of equations (3.14), (3.15) and adding that to the output of the multiplication of the forget gate (f_t) with previous cell state C_{t-1} provide the current cell state, C_t as shown in equation (3.16).

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t. \tag{3.16}$$

In the next pass, the current input, X_t and previous output h_{t-1} are passed through a Sigmoid function to obtain intermediate output, O_t.

$$O_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_0). \tag{3.17}$$

Finally, multiplying the current cell state, C_t with the intermediate output, O_t gives the current output h_t as shown in equation (3.18).

$$h_t = O_t \tanh C_t. \tag{3.18}$$

Predicting Dynamic User Preferences Using LSTM

To model dynamic user preferences, a historical dataset is required. Unfortunately, there is no real dataset for modeling individual user behavior [31]. Hence, at first, a synthetic dataset is generated for modeling dynamic user behavior. Notice that, usually, content popularity is modeled using Zipf distribution in which the parameter - skewness, controls the distribution. However, modeling the content popularity at each individual user level is different than modeling global content popularity. Therefore, a random skewness is used for all users while generating the dataset. First, a random content index order is

generated for each user. Then, the number of requests to make for the user is generated using random skewness in between a minimum and a maximum allowable skewness. The governing equation for this is given below.

$$\gamma = \left(\gamma^{max} - \gamma^{min}\right) \times \text{Uniform}(0, 1) + \gamma^{min}, \tag{3.19}$$

where γ^{max} and γ^{min} are the maximum and minimum skewness of the Zipf distribution.

Notice that, for each user, different content order and skewness are considered. While the content's order is taken randomly for each user, the skewness is controlled with equation (3.19). Furthermore, taking this value only as an initial value, afterward, correlated data are generated for the next time slots. The detailed procedures of generating the synthetic dataset are presented in Alg. 1.

Having found the historical dataset for $t \in \{1, 2, \ldots, N\}$ time slots, the next focus is on the LSTM based prediction model. A time ahead forecast of the probability of making a content request - activity level, and what content a user will request at the $t \in \{N+1\}$s time slots are also conducted. The model needs to be trained using the available historical data of the first N time slots. Then, $(N+1)^{th}$ time slot's data needs to be predicted. The detailed procedure of that is discussed in what follows.

For the training, at each time slot t, an entire row is fed to the input of the LSTM block meaning that the input X_t of the LSTM is an entire row of the user-content matrix obtained from Alg. 1. Note that from the shape of the equation (3.4), it is also visible that, at a time slot, the data of an individual user is fed to the LSTM model. Therefore, this has to be performed for all users $u_i \in \{U\}$. After training the model, the value of each row of the user-content matrix, $\mathbf{N}_{uf}^{U \times F}$ is forecasted for $t = (N+1)^{th}$ time slot. These forecast values of time slot $t = N+1$ is next used to forecast the next time slot's - $(N+2)$, data. Note that as the generated number are the prediction of how many times a user will request a content, the predicted value cannot be non-positive or fractional. However, the forecast results following the LSTM model may contain fractional values. Therefore, the necessary rounding is performed in the same way as it has been performed while generating

Algorithm 1 Generating Synthetic Data Set

select total number of users U, total number of content F and total number of historical time slots, t
for each user, $i \in \{U\}$ **do**
  select random content index order, $\forall\ f_k \in \{\mathcal{F}\}$
  $\gamma \leftarrow \text{Uniform}(\gamma_{\min}, \gamma_{\max})$ ▷ use equation (3.19)
  $\mathcal{N}_{\text{int}}^{\text{req}} \leftarrow \text{Uniform}(\mathcal{N}_{\min}^{\text{req}}, \mathcal{N}_{\max}^{\text{req}})$
  $\text{P}_{int} \leftarrow \text{Uniform}\left((0,1), \mathcal{N}_{\text{int}}^{\text{req}}\right)$ ▷ $\mathcal{N}_{\text{int}}^{\text{req}}$ Uniform random numbers $\in \{0,1\}$
  calculate $P_{f_k}(k=i)$ using equation (3.1)
  $P_{f_k} \leftarrow \sum_{j=1}^{k} P_{f_j}$ ▷ cumulative sum of (3.1)
  generate $\mathcal{N}^{\text{req}}$ using Histcounts$(\text{P}_{int}, P_{f_k})$ ▷ P_{f_k} of step (8), $\mathcal{N}^{\text{req}} \in \mathbb{R}^F$
  for $\forall\ f_k \in \{\mathcal{F}\}$ **do**
    sort $\mathcal{N}^{\text{req}}$ as per initial generated index in step (3) ▷ generation in steps (7 - 8) is not sorted
  end for
end for
return: $\mathbf{N}_{uf}^{U\times F}(t) = \mathcal{N}^{\text{req}}$, ▷ initial user content matrix, $t=1$
for each $t > 1$ **do**
  for all user **do**
    $\mathcal{N}_{\text{new}}^{\text{req}}(t)$ using the initial generated number in step (14)
    if $\mathcal{N}_{\text{new}}^{\text{req}}(t) \neq INT$ **then** ▷ INT refers to integer
      $\mathcal{N}_{\text{new}}^{\text{req}}(t) \leftarrow \text{round}(\mathcal{N}_{\text{new}}^{\text{req}}(t))$
    end if
  end for
  return: synthetic user-content data matrix $\mathbf{N}_{uf}^{U\times F}(t)$
end for

Algorithm 2 Predicting Sequential Data Using LSTM

for each cell, $j \in \mathcal{B}$ **do**
  for each user, $u_i \in \{U_{b_j}\}$ **do**
    take generated historical dataset from Alg. 1
    process the data to take the entire row for the time slot as input elements of the LSTM input
    divide the dataset into training, validation and test part
    feed the data to the LSTM model
    using the model forecast the value of the entire row for $(N+1)^{th}$ time slot
    save the trained model and store the values
  end for
end for
return: predicted value for $(N+1)$ time slot

the dataset. The detailed procedure is noted in Alg. 2.

After running Alg. 2, now $\hat{r}_i(t)$ and $\hat{q}_{f_k|u_i}(t)$, where $t = N+1$, are calculated. Remember that $r_i(t)$ denotes the probability that a user actually makes the request and $q_{f_k|u_i}(t)$ denotes the conditional probability that user u_i request for content f_k conditioned on $r_i(t)$. The predicted activity levels of the users are calculated as follows:

$$\hat{r}_i(t) = \frac{\hat{n}_{u_i}(t)}{\hat{q}(t)}, \tag{3.20}$$

where $\hat{n}_{u_i}(t) = \sum_{k=1}^{F} \hat{n}_{u_i,f_k}(t)$, $\hat{q}(t) = \sum_{i=1}^{U}\sum_{k=1}^{F} \hat{n}_{u_i,f_k}(t)$ and $t = N+1$.

The predicted conditional probability that a user request for content f_k given that she actually makes a request is calculated as follows:

$$\hat{q}_{f_k|u_i}(t) = \frac{\hat{n}_{u_i,f_k}(t)}{\hat{n}_{u_i}(t)}. \tag{3.21}$$

Thus, the predicted joint probability that a user will make a request for content f_k is calculated as follows:

$$\hat{q}_{f_k,u_i}(t) = \hat{r}_i(t)\hat{q}_{f_k|u_i}(t). \tag{3.22}$$

Without loss of generality, the preference probability of a user for the next time slots, $t = (N+1)$s, are calculated from this joint probability. Since $\hat{q}_{f_k,u_i}(t)$ is the joint probability, necessary normalization may require to keep the summation of these probabilities to be equal to 1.

Note that following the notion of the forecast model, the preference probability, $\hat{q}_{f_k,u_i}(t)$, can be modeled for all the future time slots, $\forall\ t \geq N+1$. This forecast period is completely on the system administrator's hand. Based on the requirements, it can be set to any reasonable time window. Furthermore, if the per time scale analysis is required, it can be easily modeled by considering the per time slot user's preference probabilities. However, in this works, the long term content caching probabilities are analyzed. As placing the content for each forecast time window may not be cost-efficient, the long term request probability is considered. Thus, without loss of any generality, assuming the forecast window as fixed, the future content preferences of the users are considered as the average of the predicted

$\hat{q}_{f_k,u_i}(t)$, $\forall\ t \geq N+1$.

Let N^{opt} denote this fixed time window chosen by the network administrator. Then, the average $\hat{q}_{f_k,u_i}(t_o)$, $\forall\ t_o \in \{N+1, N+2, \ldots, N+N^{\text{opt}}\}$ is considered as the preference probability of the user u_i for evaluating the system performance[4]. Let $\rho_{f_k}^{u_i}$ denote this preference probability that has to be used for the performance evaluation. This then can be calculated as follows:

$$\rho_{f_k}^{u_i} = \frac{\sum_{t_0=N+1}^{N+N^{\text{opt}}} \hat{q}_{f_k,u_i}(t_0)}{N^{\text{opt}}} \quad ,\forall\ u_i \in \{\mathcal{U}\} \quad \text{and} \quad f_k \in \{\mathcal{F}\}. \tag{3.23}$$

Since the predicted value of what content a user will request in the next time slot and what load she will create for the network are already known at this point, this chapter thus now focuses on the caching model.

3.3 Caching Model and Content Sharing Cost

In this section, at first, the caching model is discussed considering both heterogeneous and homogeneous caching placement strategies. Then, the content sharing cost is introduced. Finally, the objective functions, for both cases, are presented.

3.3.1 Caching Models

A probabilistic caching model for the content caching at the edge nodes, i.e., at both the D2D users and the sBSs are considered. Without any loss of generality, the probabilities that the sBS b_j and the user u_i store the content, f_k are defined by $\eta_{f_k}^{b_j}$ and $a_{f_k}^{u_i}$, respectively. The storage capacities of each sBS and each user are denoted by C_b and $\mathcal{C}_d$, respectively. Hence, the storage constraints of $\sum_{k=1}^{F} \eta_{f_k}^{b_j} \leq C_b$ and $\sum_{k=1}^{F} a_{f_k}^{u_i} \leq \mathcal{C}_d$, $\forall\ f_k$, b_j, and u_i are considered. In the following, the tagged user is defined as u_i and the sBS associated with the tagged user is named as tagged sBS (or serving sBS) which is denoted as b_j. Therefore, the remaining sBSs are $b_{j'}$, where $j' \in \mathcal{B} \backslash \{j\}$. Furthermore, the set of users in the coverage of b_j is defined as $\mathcal{U}_{b_j} = \{1, \ldots, U_{b_j}\}$, where U_{b_j} is the number of users including the tagged

[4] t_o represents only the optimization time slots, while t represent all time slot.

user. The probabilities that the tagged user obtains a requested content from (a) the local nodes and (b) the cloud are respectively denoted as P_l^u and P_c^u.

Heterogeneous caching model

In the heterogeneous caching placement case, heterogeneous user preferences, as well as heterogeneous caching placement strategies, are considered. In other words, the caching strategy at node i is considered to be different from that of node j. In the following, first, the probabilities of storing contents in the local nodes, P_l^u is introduced. Later on, the probability of extracting content from the cloud, P_c^u is calculated.

Firstly, the probability that the tagged user receives the requested content from one of the local D2D nodes is calculated as follows:

$$\mathrm{P}_o^{u_i} = \Pr\{\text{content } f_k \text{ is in user's own storage}\} = a_{f_k}^{u_i}, \tag{3.24}$$

where $a_{f_k}^{u_i}$ represents the probability of storing the content in the tagged user's storage. Note that collaborations among users, serving BS and neighboring BSs staying in a single cluster are considered in this chapter. However, as the D2D communication has a very short communication range, collaborations among the users are only considered within the cell. For the base stations, collaboration among the BSs that are in the same cluster is considered.

Then, the probability of obtaining a requested content from the neighbor D2D nodes of the same cell is defined as the probability that content f_k is not in the tagged user's storage but is available in at least one neighboring D2D nodes. Mathematically, this can be expressed as follows:

$$\mathrm{P}_d^{u_i} = \left(1 - a_{f_k}^{u_i}\right)\left[1 - \prod_{i' \in \mathcal{U}_{b_j} \setminus i} \left(1 - a_{f_k}^{u_{i'}}\right)\right], \tag{3.25}$$

where the term of $1 - a_{f_k}^{u_i}$ denotes the probability of the event that the requested content cannot be served by the local cache storage of the tagged node. So the content has to be found

and delivered from at least one of the set $\mathcal{U}_{b_j}\backslash i$ of D2D nodes. The term $\prod_{i'\in\mathcal{U}_{b_j}\backslash i}\left(1-a_{f_k}^{u_{i'}}\right)$ represents that none of nodes in $\mathcal{U}_{b_j}\backslash i$ caches the content. The complement of that term (i.e. $1-\prod_{i'\in\mathcal{U}_{b_j}\backslash i}\left(1-a_{f_k}^{u_{i'}}\right)$) expresses the probability of the event that at least one of the nodes in $\mathcal{U}_{b_j}\backslash i$ caches the requested content.

Next, the considered case is that the content is not available at the tagged user self cache storage as well as at any of the other $\mathcal{U}_{b_j}$ nodes' storage. Hence, the requester obtains the requested content from the serving BS. The probability that the tagged user receives the content, f_k from the serving BS is calculated as follows:

$$\mathrm{P}_{b_j}^{u_i}=\left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}_{b_j}\backslash i}\left(1-a_{f_k}^{u_{i'}}\right)\eta_{f_k}^{b_j}, \tag{3.26}$$

where the quantity of $\left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}_{b_j}\backslash i}\left(1-a_{f_k}^{u_{i'}}\right)$ is denoted by the probability that none of the D2D nodes in $\mathcal{U}_{b_j}$ caches the requested content, while $\eta_{f_k}^{b_j}$ is the probability that the serving BS stores the content.

The next case is that the requested content is not available at the local nodes as well as at the serving BS's storage. Therefore, the request is forwarded to the neighboring BSs in the same cluster. The probability that the tagged user receives the requested content from one of the neighboring BSs is calculated as follows:

$$\mathrm{P}_{\mathrm{B}}^{u_i}=\left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}_{b_j}\backslash i}\left(1-a_{f_k}^{u_{i'}}\right)\left(1-\eta_{f_k}^{b_j}\right)\left[1-\prod_{j'\in\mathcal{B}\backslash j}\left(1-\eta_{f_k}^{b_{j'}}\right)\right], \tag{3.27}$$

where the term of $\left(1-\eta_{f_k}^{b_j}\right)$ is the probability that the tagged/serving BS does not cache the content, while the term of $\prod_{j'\in\mathcal{B}\backslash j}\left(1-\eta_{f_k}^{b_{j'}}\right)$ represents the probability that none of the $(B-1)$ BSs within the cluster store the content. Note that the complement of that quantity, i.e. $\left[1-\prod_{j'\in\mathcal{B}\backslash j}\left(1-\eta_{f_k}^{b_{j'}}\right)\right]$, presents the probability that at least one of the neighboring BSs caches the content. Note that the complement of that quantity, i.e. $\left[1-\prod_{j=1}^{B-1}\left(1-\eta_{f_k}^{b_j}\right)\right]$, presents the probability that at least one of the BSs caches the content.

Thus, the total local cache access probability can be calculated as follows:

$$\begin{aligned} \mathrm{P}_l^{u_i} &= \mathrm{P}_o^{u_i} + \mathrm{P}_d^{u_i} + \mathrm{P}_{b_j}^{u_i} + \mathrm{P}_{\mathrm{B}}^{u_i} \\ &= 1 - \left(1 - a_{f_k}^{u_i}\right) \prod_{i' \in \mathcal{U}_{b_j} \setminus i} \left(1 - a_{f_k}^{u_{i'}}\right) \left(1 - \eta_{f_k}^{b_j}\right) \prod_{j' \in \mathcal{B} \setminus j} \left(1 - \eta_{f_k}^{b_{j'}}\right). \end{aligned} \tag{3.28}$$

Finally, in the case that all of the above cases fail and hence, the request needs to be forwarded to the cloud is considered. This means that the requested content is not available at the user's local storage, at any of the other users' storage and at any of the BSs' storage within the cluster. The probability, P_c^u, that the tagged user obtains the requested content from the cloud is determined as follows:

$$\begin{aligned} \mathrm{P}_c^{u_i} &= 1 - \mathrm{P}_l^{u_i} \\ &= \left(1 - a_{f_k}^{u_i}\right) \prod_{i' \in \mathcal{U}_{b_j} \setminus i} \left(1 - a_{f_k}^{u_{i'}}\right) \left(1 - \eta_{f_k}^{b_j}\right) \prod_{j' \in \mathcal{B} \setminus j} \left(1 - \eta_{f_k}^{b_{j'}}\right). \end{aligned} \tag{3.29}$$

Homogeneous caching model

In a homogeneous caching model, cache-enabled nodes store the same set of contents. This can be referred to as storing the same copy of the content to all same tier cache-enabled nodes [45]. Thus, the probability of storing a content into the cache-enabled nodes are equal for the same tier local nodes. That is the probability of storing a content f_k is same for all D2D nodes. Therefore, $a_{f_k}^{u_1} = a_{f_k}^{u_2} = \cdots = a_{f_k}^{u_{U_{b_j}}}$ and $\eta_{f_k}^{b_1} = \eta_{f_k}^{b_2} = \cdots = \eta_{f_k}^{b_B}$, where $a_{f_k}^{u_i} \neq \eta_{f_k}^{b_j}, \forall i, j$. For simplicity of expressing the equations, the superscripts are dropped off and the storing probability are denoted as a_{f_k} and η_{f_k} for D2D nodes and base stations, respectively. Furthermore, the number of users in a cell is denoted by U_c.

In homogeneous case, using the definitions, equations (3.24-3.27) are rewritten as follows:

$$\mathrm{P}_o^{\mathrm{hom}} = a_{f_k}. \tag{3.30}$$

$$\mathrm{P}_d^{\mathrm{hom}} = (1 - a_{f_k}) \left[1 - (1 - a_{f_k})^{U_c - 1}\right]. \tag{3.31}$$

$$\mathrm{P}_{b_0}^{\mathrm{hom}} = (1 - a_{f_k})^{U_c} \eta_{f_k}. \tag{3.32}$$

$$\mathrm{P}_{\mathrm{B}}^{\mathrm{hom}} = (1 - a_{f_k})^{U_c} (1 - \eta_{f_k}) \left[1 - (1 - \eta_{f_k})^{B-1}\right]. \tag{3.33}$$

Then, the local access probability can be calculated as follows:

$$\begin{aligned} \mathrm{P}_{l}^{\mathrm{hom}} =& 1 - (1 - a_{f_k})^{U_c} + (1 - a_{f_k})^{U_c} \eta_{f_k} + (1 - a_{f_k})^{U_c} (1 - \eta_{f_k}) \left[1 - (1 - \eta_{f_k})^{B-1}\right] \\ =& 1 - (1 - a_{f_k})^{U_c} (1 - \eta_{f_k})^{B}. \end{aligned} \tag{3.34}$$

Finally, the probability of accessing a content from the cloud is derived as follows:

$$\mathrm{P}_{c}^{\mathrm{hom}} = 1 - \mathrm{P}_{l}^{\mathrm{hom}} = (1 - a_{f_k})^{U_c} (1 - \eta_{f_k})^{B}. \tag{3.35}$$

As both heterogeneous and homogeneous caching placement models are introduced this far, next this chapter introduces content sharing cost.

3.3.2 Content Sharing Cost

In this subsection, the cost of collaborating and sharing the contents among different nodes are described. Two types of costs, namely - (a) storage cost and (b) communication cost are considered in this chapter. For the communication cost, per bit per meter transmission cost is considered. For an example, if a content has a size of S_f bits, then the transmission cost between D2D nodes residing d meters apart is calculated as follows:

$$\Lambda_{d}^{com} = S_f \times \delta_d \times d, \tag{3.36}$$

where δ_d is the cost of per byte transmission in case of D2D transmission.

For simplicity, it is not irrational to consider equal storage cost for all nodes. In this chapter, therefore, storage cost is considered to be equal for all nodes. This cost is denoted by Λ_{i}^{stor}.

The costs of receiving the contents from the cloud, the nearby BS, the tagged BS, and the nearby D2D node are denoted as in the following general equation. The cost of

obtaining the content from a node i is expressed as follows:

$$\phi_i = \Lambda_i^{com} + \Lambda_i^{stor}, \tag{3.37}$$

where $\Lambda_i^{com} = S_f \times \delta_i \times d_i$ is the transmission cost from node i to the tagged user and Λ_i^{stor} is the cost of storing the content at node i. Here, $i \in \{\mathrm{C}, \mathrm{BS}, \mathrm{b}_0, \mathrm{d}\}$. Therefore, $\phi_C, \phi_{BS}, \phi_{b_0}$ and ϕ_d represent the costs of extracting a content from the cloud, other BS in the same cluster, serving BS and other D2D nodes in the same cell, respectively. Furthermore, it is assumed that the transmission cost is zero if the requested content is available in the storage of the requester node itself. However, in that particular case, the storage cost needs to be considered. The relationship of the costs are presented in Proposition 3.3.1.

Proposition 3.3.1. *In general, it is assumed that the costs of receiving the required contents from different nodes satisfy the following constraint*

$$\phi_C >> \phi_{BS} >> \phi_{b_0} >> \phi_d. \tag{3.38}$$

After introducing the cost terms, the cost function for accessing a content f by a tagged user can be expressed as follows:

$$\Xi_c = \Lambda^{stor}\mathrm{P}_o^{u_i} + \phi_d \mathrm{P}_d^{u_i} + \phi_b \mathrm{P}_{b_0}^{u_i} + \phi_{BS}\mathrm{P}_{\mathrm{B}}^{u_i} + \phi_C \mathrm{P}_c^{u_i}. \tag{3.39}$$

In equation (3.39), the first term is considered because of the fact that a requested content might need to be stored at the requesting user's own node. The second, third, fourth and fifth terms are considered for the case of accessing the requested content from the - neighbor D2D nodes, serving base station, other base stations of the cluster and cloud, respectively.

Recall that the preference probability, $\rho_{f_k}^{u_i}$, is the long term probability that the user $u_i \in \{\mathcal{U}\}$ requests the content f_k. It is calculated in equation (3.23) based on the proposed LSTM model. Then, the average cost for accessing the content among all the users and all the contents in a cluster is the weighted average of Ξ_c in (3.39). This quantity is calculated

as follows:

$$\Xi_{\pi} = \sum_{j=1}^{B} \sum_{i=1}^{U_{b_j}} \sum_{k=1}^{F} \rho_{f_k}^{u_i} \frac{\Xi_c}{U}, \tag{3.40}$$

where $j \in \{1, 2, \ldots, B\}$ and U represent the small cells and total number of users in a cluster, respectively, while U_{b_j} represents the number of users in the coverage of BS b_j.

Heterogeneous caching model case

In the case of heterogeneous caching placement, from (3.39) and (3.40), Ξ_{π} can be rewritten as follows:

$$\begin{aligned} \Xi_{\pi}^{\text{het}} = \frac{1}{U} \sum_{j=1}^{B} \sum_{i=1}^{U_{b_j}} \sum_{k=1}^{F} \rho_{f_k}^{u_i} \Bigg\{ & \Lambda^{stor} a_{f_k}^{u_i} + \phi_d \left(1 - a_{f_k}^{u_i}\right) - \\ & A_1 \left[\phi_d - \phi_b \eta_{f_k}^{b_j} - \phi_{BS} \left(1 - \eta_{f_k}^{b_j}\right) + A_2 \left(\phi_{BS} - \phi_C\right) \right] \Bigg\}, \end{aligned} \tag{3.41}$$

where u_i and b_j represent tagged user and serving base station, respectively. A_1 and A_2 are as shown in equations (3.42-3.43). The detailed derivation of equation (3.41) is shown in Appendix A.1 where (A.1a) - (A.1b) are obtained using the cache storage probabilities at different nodes. (A.1b) - (A.1e) are obtained using algebraic manipulations.

$$A_1 = \left(1 - a_{f_k}^{u_i}\right) \prod_{i' \in \mathcal{U}_{b_j} \setminus i} \left(1 - a_{f_k}^{u_{i'}}\right). \tag{3.42}$$

$$A_2 = \left(1 - \eta_{f_k}^{b_j}\right) \prod_{j' \in \mathcal{B} \setminus j} \left(1 - \eta_{f_k}^{b_{j'}}\right). \tag{3.43}$$

Next, the focus is to determine the optimal cache placement (i.e. $a_{f_k}^{u_i}$ and $\eta_{f_k}^{b_j}$) considering the BS assisted caching mechanism. This is called the base station assisted mechanism, because the contents may be received from other BSs in the same cluster and/or from the cloud via the serving BS. In particular, the optimization problem is formulated next aiming to minimize the content sharing cost. The formulated optimization problem is given as

follows:

$$\mathrm{P}_1: \quad \underset{a_{f_k}^{u_i}, \eta_{f_k}^{b_j}}{\text{minimize}} \quad \Xi_{\pi}^{\text{het}} \tag{3.44a}$$

$$\text{s. t.} \quad \sum_{k=1}^{F} a_{f_k}^{u_i} \leq \mathcal{C}_d, \quad \forall\ u_i, f_k \tag{3.44b}$$

$$\sum_{k=1}^{F} \eta_{f_k}^{b_j} \leq C_b, \quad \forall\ b_j, f_k \tag{3.44c}$$

$$0 \leq a_{f_k}^{u_i} \leq 1,\ 0 \leq \eta_{f_k}^{b_j} \leq 1,\ \forall\ u_i,\ b_j\ \&f_k. \tag{3.44d}$$

In problem P_1, the constraints in (3.44b) and (3.44c) indicate that the total the contents cached at the node (i.e. a D2D node and a BS) must not excess the node's storage capacity. The constraint in (3.44d) simply states that caching probabilities have to be in the range of $[0, 1]$. Moreover, the cost function, Ξ_{π}^{het}, is given in (3.41).

Homogeneous caching model case

In the case of homogeneous caching placement, from (3.39) and (3.40), Ξ_{π} is rewritten as follows:

$$\Xi_{\pi}^{\text{hom}} = \frac{1}{U} \sum_{j=1}^{B} \sum_{i=1}^{U_c} \sum_{k=1}^{F} \rho_{f_k}^{u_i} \Bigg\{ \Lambda^{stor} a_{f_k} + \phi_d \left(1 - a_{f_k}\right) - B_1 \Bigg[\phi_d - \phi_b \eta_{f_k} - \phi_{BS} \left(1 - \eta_{f_k}\right) + B_2 \left(\phi_{BS} - \phi_C\right) \Bigg] \Bigg\}, \tag{3.45}$$

where u_i represents the tagged user, B_1 and B_2 are as shown in equations (3.46-3.47). The detailed derivation of equation (3.45) is simply some algebraic manipulation of equation (3.41).

$$B_1 = \left(1 - a_{f_k}\right)^{U_c}. \tag{3.46}$$

$$B_2 = \left(1 - \eta_{f_k}\right)^{B}. \tag{3.47}$$

Here, it should be stressed out that all edge nodes (D2D nodes and BSs) are assumed to have an equal caching policy in the homogeneous caching placement case [45]. Further, recall that $\rho_{f_k}^{u_i}$, calculated in equation (3.23), denotes the probability that user u_i requests content f_k at the future time slot. Although the equal caching policy is considered for all cache enabled nodes, heterogeneous content preferences of the users are still being considered. Following the homogeneous notion, the optimization problem in P_1 is reformulated as follows:

$$\mathrm{P}_2: \quad \underset{a_{f_k},\eta_{f_k}}{\text{minimize}} \quad \Xi_{\pi}^{\text{hom}} \tag{3.48a}$$

$$\text{s. t.} \quad \sum_{k=1}^{F} a_{f_k} \leq \mathcal{C}_d, \quad \forall\ u, f_k \tag{3.48b}$$

$$\sum_{k=1}^{F} \eta_{f_k} \leq C_b, \quad \forall\ b, f_k \tag{3.48c}$$

$$0 \leq a_{f_k} \leq 1,\ 0 \leq \eta_{f_k} \leq 1,\ \forall\ u,\ b\ \&f_k. \tag{3.48d}$$

The constraints (3.48b) - (3.48d) are used for the same reasons as in problem P_1.

3.4 Observations and Joint Solver for the Objective Functions

In this section, the objective functions for both heterogeneous and homogeneous caching placement cases are analyzed. To efficiently solve the optimization problems, necessary algorithms are also proposed in this section.

3.4.1 Observations of the Objective Functions

To gain insights into the parameter configuration of the caching placement, at first the optimization with respect to the caching placement at the BSs, $\eta_{f_k}^{b_j}$s and at the users' nodes, $a_{f_k}^{u_i}$ are studied.

Observations in Heterogeneous Caching Policy Case

First, the properties of the cost function, Ξ_{π}^{het} with respect to $\eta_{f_k}^{b_j}$ for a given $a_{f_k}^{u_i}$ are characterized in Proposition 3.4.1.

Proposition 3.4.1. *The objective function Ξ_{π}^{het} of problem P_1 is monotonically non-increasing with respect to $\eta_{f_k}^{b_j}$, for all $b_j \in \{B\}$ (i.e. the cache placement for both serving BS and other BSs in the same cluster).*

Proof. The proof is left in Appendix A.2. ■

Similarly, the optimization with respect to the caching placement at the users, $a_{f_k}^{u_i}$ for a given the caching placement at the BSs, $\eta_{f_k}^{b_j}$ is studied. The properties of cost function, Ξ_{π}^{het} with respect to $a_{f_k}^{u_i}$ for a given $\eta_{f_k}^{b_j}$ are characterized in Proposition 3.4.2.

Proposition 3.4.2. *The monotonicity of objective function Ξ_{π}^{het} of problem P_1 with respect to $a_{f_k}^{u_i}$ is inconclusive. It cannot be determined whether the cost function is monotonically non-decreasing or non-increasing with respect to $a_{f_k}^{u_i}$, $\forall a_{f_k}^{u_i} \in \mathrm{dom}(a_{f_k}^{u_i})$.*

Proof. The proof is left in Appendix A.3. ■

Based on these observations, it is clear that the objective function P_1 is intractable and hard to solve. Besides, some authors claim that it may not be efficient to optimize at each user node in a larger network platform like CDN due to the complexity [46, 47]. Thus, the homogeneous case may benefit in some occasions. This thesis, therefore, also investigate the performance of the homogeneous case. In the next section, analysis for the homogeneous caching placement case is presented.

Observations in Homogeneous Caching Policy Case

Problem P_2 i.e. the optimization problem for the homogeneous caching placement case, is characterized in Proposition 3.4.3 and Proposition 3.4.4.

Proposition 3.4.3. *Propositions 3.4.1 and 3.4.2 hold in homogeneous caching placement scenario.*

Proof. The proof is straight forward, and thus the proof is left for brevity. ■

The convexity of problem P_2 is now investigated. It is characterized by the following Proposition 3.4.4.

Proposition 3.4.4. *The optimization problem* P_2 *is not convex.*

Proof. The proof is left in appendix A.4. ■

As the Hessian of Ξ_{π}^{hom} in (3.45) is indefinite and problem P_2 is not convex, it is very hard to achieve an optimal solution. In what follows, necessary algorithms are proposed to solve the problem while achieving a sub-optimal solution.

3.4.2 Algorithm and Solver for the Joint Optimizations

In this subsection, the proposed algorithms are presented to efficiently solve problems P_1 and P_2. As per the above analysis and observations, the optimization problems P_1 and P_2 both has two variables, i.e. $a_{f_k}^{u_i}$ and $\eta_{f_k}^{b_j}$. The joint optimization problems are not convex as well. Further, notice that user preferences vary dynamically over different time slots which are captured using the LSTM model. Considering these dynamics, this thesis intends to capture the long term caching placement probabilities at the cache-enabled nodes. The significance of doing this is that a system administrator may need to know multiple time slots forecasts for the to-be-requested contents. If the binary cases[5], are considered, the obtained results are only for a single time slot. Instead, the goal of this thesis is to optimize the caching placement probabilities for multi-time scale cases. In doing that, two indicator functions, $\mathbb{I}_{f_k}^{u_i}(t_o)$ and $\mathbb{I}_{f_k}^{b_j}(t_o)$, are considered to denote the cache placement indicator at user node and base station, respectively for time slot t_0. $\mathbb{I}_{f_k}^{u_i}(t_o) = 0$ and $\mathbb{I}_{f_k}^{u_i}(t_o) = 1$ indicates that content f_k is not placed and placed into the cache storage of user u_i, respectively for time slot t_o. This is essentially the binary case. This has to be considered for all optimization time slots and then finally, the cache placement probabilities are required to be calculated.

[5]A binary case considers only 0 or 1. For example, if $a_{f_k}^{u_i} = 0$, the content f_k is not cached at the user node u_i.

Considering the above facts, the cache placement probabilities are calculated as follows:

$$a_{f_k}^{u_i} = \frac{\sum_{t_o=N+1}^{N+N^{\text{opt}}} \mathbb{I}_{f_k}^{u_i}(t_o)}{N^{\text{opt}}}, \ \forall \ u_i \text{ and } f_k, \tag{3.49}$$

where N^{opt} is the total number of time slots for the optimization.

$$\eta_{f_k}^{b_j} = \frac{\sum_{t_0=N+1}^{N+N^{\text{opt}}} \mathbb{I}_{f_k}^{b_j}(t_o)}{N^{\text{opt}}}, \forall \ b_j \text{ and } f_k. \tag{3.50}$$

Now, to efficiently optimize the problems, necessary algorithms are proposed in what follows.

Algorithm and solver for heterogeneous caching placement

In reality, solving the optimization problem in the case of heterogeneous caching placement strategy is more interesting and beneficial for a CDN. However, the optimization problem P_1 is very challenging and contains a large number of system parameters. Since the problem is not convex, it is extremely hard to get the optimal solutions. Therefore, heuristic algorithms are proposed to efficiently solve the joint optimization problem P_1. Moreover, three scenarios are considered for placing the contents at the nodes for the heterogeneous case. The three sub-cases are - (a) *collaborative greedy caching - base station first (non-overlapping)* (b) *collaborative greedy caching - user first (non-overlapping)* and (c) collaborative greedy overlapping caching.

Collaborative greedy caching - base station first (non-overlapping): One way to think about this is to store as much content as possible. Therefore, the aim is to store the commonly preferred contents, f_{com}^{C} into the base stations cache storage first. No overlapping is considered in this case. In other words, unique content is stored at each cache-enabled nodes. First, the contents f_{com}^{C} are stored at the base stations. After that, if there is any place left, other preferred contents of the users (that are not already stored into the users' cache storage) of that respective cells are stored later on. Next, the users' preferred contents are placed into their cache storage. While doing so, it needs to be assured that there is no

Algorithm 3 Collaborative greedy caching - base station first (non-overlapping)

for each time slot, t_o of the optimization of P_1 **do**
  Input: user content preference, $\hat{q}_{f_k,u_i}(t_o),\ \forall\ u_i$
  calculate: $f^{C}_{\text{com}} = f^{\text{pref}}_{b_1} \cap f^{\text{pref}}_{b_2} \cap f^{\text{pref}}_{b_3}$ and $f^{b_j b_l}_{\text{com}} = f^{\text{pref}}_{b_j} \cap f^{\text{pref}}_{b_l}, \forall\ U\ \&\ F$ ▷ $f^{\text{pref}}_{b_j}$ represents the common contents that are preferred by the users in cell b_j
  $f_{\text{stored}} = []$, $f^{C_{\text{rest}}}_{\text{com}} = []$, $C^{b}_{\text{avail}} = []$, $f^{U_{\text{rest}}}_{\text{pref}} = []$
  for each cell, $j \in \mathcal{B}$ **do**
    store the common contents at first ▷ f^{C}_{com} and $f^{b_j b_l}_{\text{com}}$
    update f_{stored}, $f^{C_{\text{rest}}}_{\text{com}}$ and C^{b}_{avail} ▷ based on content popularity
  end for
  return $\mathbb{I}^{b_j}_{f_k}(t_o)$, f_{stored} and C^{b}_{avail}
  for each cell, $j \in \mathcal{B}$ **do**
    for $\forall\ u_i \in \{U_{b_j}\}$ **do**
      for $\forall\ f_k \in \{f^{\text{pref}}_{u_i}\}$ **do** ▷ $f^{\text{pref}}_{u_i}$ represents the contents that are preferred by user u_i
        if $f^{\text{pref}}_{u_i} \notin f_{\text{stored}}$ && $C_d \neq$ full **then**
          $\mathbb{I}^{u_i}_{f_k}(t_o) \leftarrow 1$
          update f_{stored}, $f^{U_{\text{rest}}}_{\text{pref}}$
        end if
      end for
    end for
    if $C^{b}_{\text{avail}} \neq 0$ **then**
      fill out the storage with $f^{U_{\text{rest}}}_{\text{pref}}$
    end if
  end for
  return $\mathbb{I}^{u_i}_{f_k}(t_o)$ and $\mathbb{I}^{b_j}_{f_k}(t_o)$
end for
calculate $\bar{a}^{u_i}_{f_k}$ and $\bar{\eta}^{b_j}_{f_k}, \forall\ u_i\ \&\ b_j$ using equations (3.49-3.50)
Return Ξ^{het}_{π}

overlapping of similar content. After completing storing the contents at the UE level, the base station's cache storage - given that there is actually some space left in its (BS) storage - is updated. The detailed procedures are listed in Alg. 3.

Collaborative greedy caching - user first (non-overlapping): In this case, a non-overlapping cache placement strategy is considered. Here, user cache storage is filled with the most requested and popular content first. Then, the residual contents are placed at the base stations. It is worth mentioning that it is very similar to the *collaborative greedy caching - base station first (non-overlapping)* case. However, the difference is - the contents are placed at the user level first. For brevity, the algorithm is not presented here.

Collaborative greedy overlapping caching: In this case, a completely greedy caching mechanism is adopted. As the cost of getting the requested content from other nodes is higher than storing the content at the requester node, the aim of this algorithm is to place as many to-be-requested content as possible into the requester cache storage. Recall that the prediction model can predict what content a user will request ahead of time. Therefore, it makes sense to polish the caching policy based on the user's preferences. Using the forecast information, the to-be-requested content by the users is placed into their cache storage for each time slot. This gives the indicator functions $\mathbb{I}_{f_k}^{u_0}(t_o)$s. Finding the indicator functions then gives the long term cache placement probabilities. For the base station's cache storage, the remaining contents are placed based on their popularity profile. Finally, the caching placement probabilities $a_{f_k}^{u_i}$ and $\eta_{f_k}^{b_j}$ are calculated using equations (3.49) and (3.50), respectively. The detailed algorithm for this case is presented in Alg. 4.

Algorithm and solver for homogeneous caching placement

To tackle the complexity, one may consider homogeneous caching placement. Recall that in the homogeneous caching policy, all nodes in the same tier place the same content into their cache stores. As the problem P_2 is not a convex problem, it is hard to get the optimal solution. Considering that, again a heuristic algorithm is proposed to efficiently solve the joint optimization problem. In this case, all D2D nodes in the same cell are assumed to follow homogeneity while placing the content. Similarly, all base stations in the same cluster are considered to follow homogeneity. However, the preferences of the users are not homogeneous. Each user has a different preference than others. Therefore, it is expected that the system performance will degrade while the complexity will definitely reduce. That is, there is a trade-off between performance and complexity.

In the homogeneous caching model, for all optimization time slots, contents are placed at the user level first. Then, the residual contents are stacked and sorted (based on their popularity). Finally, the base stations' cache stores are filled out with the most popular content. The detailed procedure is presented in Algorithm 5.

Algorithm 4 Collaborative Greedy overlapping Caching

for each time slot, t_o of the optimization of P_1 **do**
  input: predicted user content preference, $\hat{q}_{f_k,u_i}(t_o)$
  for each cell, $j \in \mathcal{B}$ **do**
    $f^u_{\text{stored}} = []$, $f^{\text{rest}}_u = []$, $C^{\text{avail}}_d = []$, $Checksum = 0$
    for each user, $u_i \in \{U_{b_j}\}$ **do**
      find f_{pref} and sort f_{pref} based on $\hat{q}_{f_k,u_i}(t_o)$
      if $len(f_{\text{pref}}) > C_d$ **then**
        $\mathbb{I}^{u_i}_{f_k}(t_o) \leftarrow index(f_{\text{pref}}[0 : C_d])$
        f^u_{stored}.append$(index(f_{\text{pref}}[0 : C_d]))$
        $f^{\text{rest}}_u \leftarrow index(f_{\text{pref}}[C_d : end])$
      else ▷ $len(f_{\text{pref}}) \leq C_d$
        $\mathbb{I}^{u_i}_{f_k}(t_o) \leftarrow index(f_{\text{pref}})$
        f^u_{stored}.append$(index(f_{\text{pref}}))$
        $S_{\text{avail}} = C_d - len(f_{\text{pref}})$
        C^{avail}_d.append(S_{avail})
      end if
    end for
    find the index of f^{rest}_u and $\hat{q}_{f_k,u_i}(t_o)$
    $f^{\text{rest}_{\text{up}}}_u \leftarrow sort(f^{\text{rest}}_u)$ ▷ descending order
    if $len(f^{\text{rest}_{\text{up}}}_u) > \sum_{i=1}^{U_{b_j}}(C^{\text{avail}}_d)$ **then**
      for $\forall\ u_i$ in which $C^{avail}_d \neq 0$ **do**
        $\mathbb{I}^{u_i}_{f_k}(t_o)$.extend$(f^{\text{rest}_{\text{up}}}_u[0 : C^{\text{avail}}_d])$, $\forall$ item in $f^{\text{rest}_{\text{up}}}_u \notin f^u_{\text{stored}}$ ▷ if in f^u_{stored}, store the next popular one and delete it from $f^{\text{rest}_{\text{up}}}_u$
        $f^{\text{rest}_{\text{up}}}_u = f^{\text{rest}_{\text{up}}}_u[C^{\text{avail}}_d : end]$
      end for
      set $Checksum+ = 1$
    else
      repeat steps (21-24), if any storage is yet left consider storing the most popular content in that cell
    end if
    if $Checksum \neq 0$ **then**
      if $len(f^{\text{rest}_{\text{up}}}_u) > C_b$ **then**
        $\mathbb{I}^{b_j}_{f_k}(t_o) \leftarrow 1, \forall\ f_k \in f^{\text{rest}_{\text{up}}}_u[0 : C_b]$
      else
        $\mathbb{I}^{b_j}_{f_k}(t_o) \leftarrow 1, \forall\ f_k \in f^{\text{rest}_{\text{up}}}_u$
        fill out the BS storage (if any space left after step 33) with the most popular content of the cell
      end if
    else
      repeat step (34)
    end if
  end for
end for
calculate $\bar{a}^{u_i}_{f_k}$ and $\bar{\eta}^{b_j}_{f_k}$, $\forall\ u_i$ & b_j using equations (3.49-3.50)
Return Ξ^{het}_{π}

Algorithm 5 Collaborative Edge Caching Algorithm: Homogeneous Case

for each time slots, t_o of the optimization of P_2 **do**
  for each cell, $j \in \mathcal{B}$ **do**
    calculate $\Omega^{j}_{f_k} = \sum_{i=1}^{U_{b_j}} \hat{q}_{f_k,u_i}(t_o)$
    find and sort $f^{\text{pref}}_{b_j}$ using $\Omega^{j}_{f_k}$ ▷ descending order
    for $\forall\ u_i \in \{U_{b_j}\}$ **do**
      $\mathbb{I}^{u_i}_{f_k}(t_o) \leftarrow 1,\ \forall\ f_k \in\ f^{\text{pref}}_{b_j}[0 : C_d]$
      $f^{\text{stored}}_{b_j}$.append$[index(f^{\text{pref}}_{b_j}[0 : C_d])]$
      $f^{\text{residual}}_{b_j}$.append$[index(f^{\text{pref}}_{b_j}[C_d : end])]$
    end for
    $f^{\text{residual}}_{\text{Cell}}$.append$(f^{\text{residual}}_{b_j})$
  end for
  calculate $\Omega^{\text{Cell}}_{f_k} = \sum_{j=1}^{B} \sum_{i=1}^{U_{b_j}} \hat{q}_{f_k,u_i}(t_o)$
  sort $f^{\text{residual}}_{\text{Cell}}$ based on $\Omega^{\text{Cell}}_{f_k}$ ▷ descending order
  for $\forall\ j \in \mathcal{B}$ **do**
    $\mathbb{I}^{b_j}_{f_k}(t_o) \leftarrow 1,\ \forall\ f_k \in\ f^{\text{residual}}_{\text{Cell}}[0 : C_b]$
    $f^{\text{stored}}_{b_j}$.append$[index(f^{\text{residual}}_{\text{Cell}}[0 : C_b])]$
  end for
end for
calculate $\bar{a}_{f_k}$ and $\bar{\eta}_{f_k}$ using equations (3.49-3.50)
calculate and **return**: Ξ^{hom}_{π}

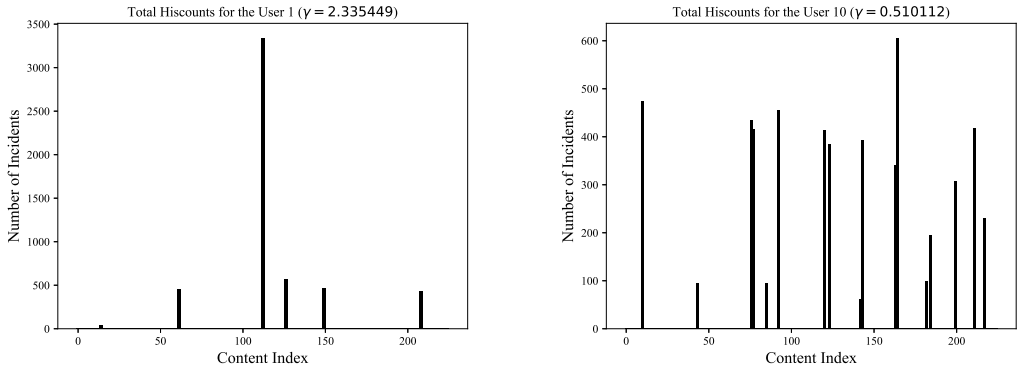
3.5 Results and Discussion

Following the proposed models and algorithms, the obtained results are discussed in this section. First, the user preferences prediction model of the proposed LSTM based model is discussed. After that, the result of it will be used for analyzing the caching performance. Note that it is assumed that all D2D nodes (users) are uniformly distributed and in the coverage region of a serving base station. For simplicity, it is also assumed that all user nodes have good SINR and channel quality. In this chapter, it is assumed that all users satisfy SINR $\geq \mathcal{T}$, where $\mathcal{T}$ is the predefined threshold [48]. Here, it is also clarified that as per Proposition 3.3.1, a fixed cost for all nodes is considered for the convenience. The simulation parameters that are used for the performance evaluations are presented in Table 3.2.

3.5.1 User Preference Prediction

Table 3.2: Simulation Parameters

Symbol	**Value**
Total number of content, F	225
Total number of users, U	45
Total number of sBS in a cluster, B	3
Total number of users under a serving BS	15
BS cache size, C_b	5 : 14
UE cache size, $\mathcal{C}_d$	1 : 4
Number of historical time slots, $t \in \{1, 2, \ldots, N\}$	$N = 250$
Number of optimization time slots, $t_0 = \{N+1, N+2, \ldots, N+N^{\text{opt}}\}$	$N^{\text{opt}} = 50$
Storage cost, Λ^{stor}	2000
Communication cost, $\{\Lambda_d^{com}, \Lambda_{b_0}^{com}, \Lambda_{BS}^{com}, \Lambda_C^{com}\}$	$\{100, 500, 1000, 5000\}$



(a) Histogram Count of User 1

(b) Histogram Count of User 10

Fig. 3.3: Histogram count of user 1 and user 10

Different skewness of the Zipf distributions are considered for all the users. Using Alg. 1, the initial content requests are generated. Based on that, correlated request numbers are generated using equation (3.51).

$$n_{u_i f_k}(t) = n_{u_i f_k}(t_{\text{int}}) + \sum_{n=1}^{\infty} A_n \sin(nt) + \epsilon(t), \tag{3.51}$$

where $n_{u_i f_k}(t_{\text{int}})$ represents initial generated number for time slot 1, t represents rest of the time slots for which the correlated data are being generated, A represents amplitude and $\epsilon(t)$ is Normal random variable with mean 0 and variance 1. $A_n = 1$ where $n = 1, 2, 3$ and

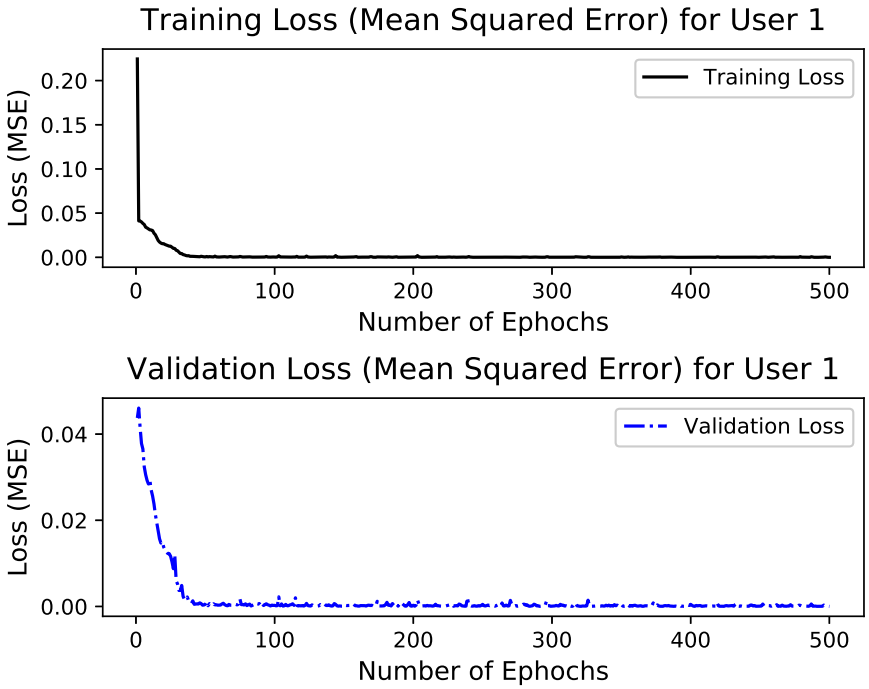


Fig. 3.4: Loss for user 1

$t = 2, 3, \ldots, 250$ are considered for the simulation of this chapter. Also, as the requested incident number is non-negative and integer valued, necessary replacement of any negative number with 0 and rounding are performed.

Following this notion, the histogram counts of the requests made by the users are presented in Fig. 3.3. As the figure shows, the skewness, γ is not similar for user 1 and user 10. Note that Zipf distribution depends on this parameter. Therefore, this chapter is essentially considering heterogeneous user preferences for all the users. Furthermore, recall that in each time slot, the generated number varies. This can easily be visualized from the sample figure in Fig. 3.5.

Next, using the proposed prediction model in Alg. 2, the contents that will be requested in the next time slot by the users are sequentially predicted. While implementing the LSTM model, 80% of the data is taken as training data and rest 20% is used for the testing purpose. It is also assured that the model is not overfitting or underfitting by considering 10% validation data set on the training part. The loss (mean squared error) when the model

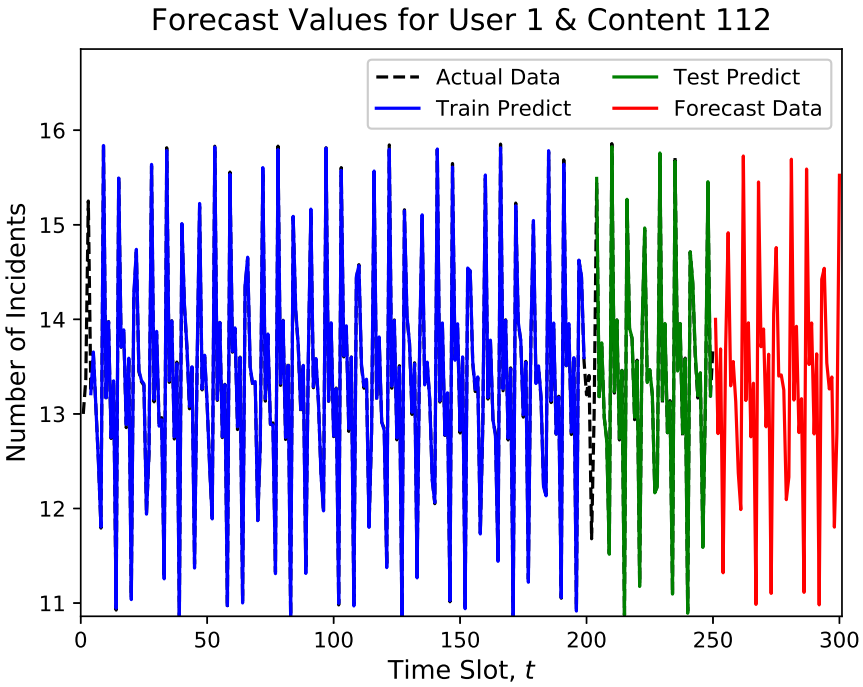


Fig. 3.5: Predicted value for user 1 and content 112

fits is shown in Fig. 3.4. The mean squared error losses for both the training and validation part are close to zero after only a minimal number of epochs. The figure also validates that the loss of both cases is nearly identical. This, thus, means that the proposed model works perfectly. Therefore, this model is used for the required sequential prediction.

The prediction made by this model for the most popular content of user 1 is shown in Fig. 3.5. First, the historical data of time slots $t \in \{1, 2, \ldots, 250\}$ are used to predict for $t = 251^{st}$ time slot. A $look\_back = 3$, 100 LSTM blocks and Sigmoid activation function are used. As shown in the figure, it is readily visible that the proposed model is predicting accurately. Therefore, the model is used for forecasting the t_o time slot value. Then using these results, the caching policy is designed. While doing that, the data of $\{t = 249, t = 250, t = 251\}$ are used to predict for $t = 252$. Similarly, the prediction up to $t = 300$ is forecasted. Here it needs to be clarified that the generated results or the original dataset have not been rounded to show the actual effectiveness of the proposed model. Doing the necessary rounding is pretty straightforward, though. Using the model, the following

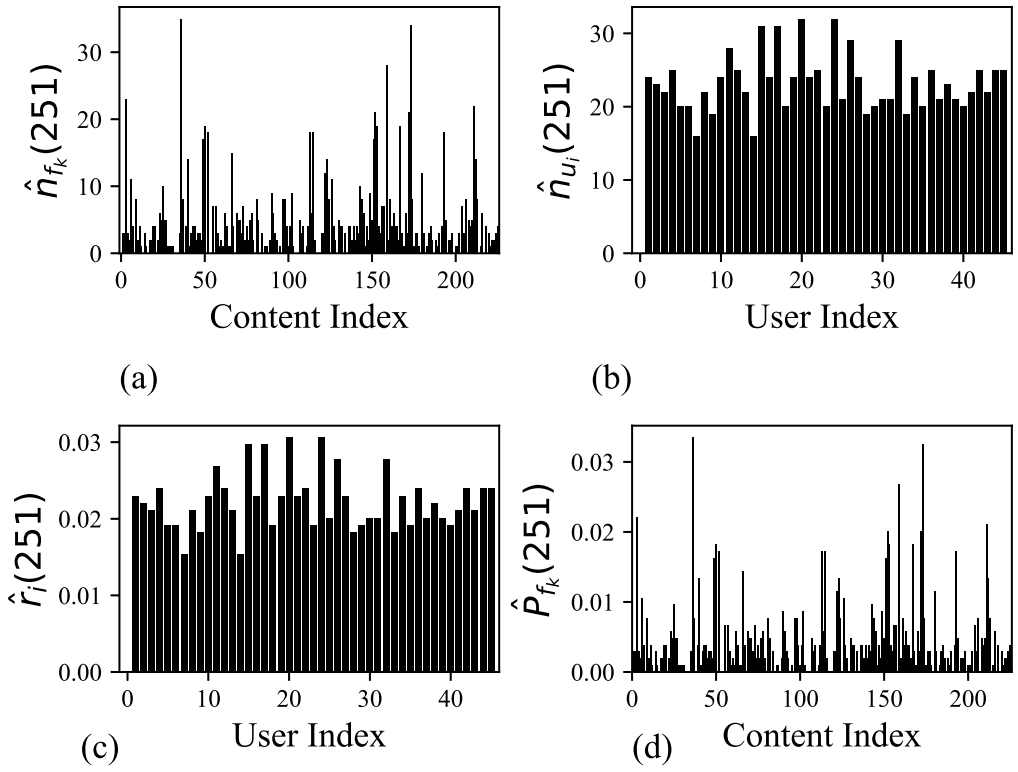


Fig. 3.6: (a) Histogram counts: total number of requests made for the content by all users, (b) histogram counts: total number of requests made by the users for all content in the catalog (c) predicted activity levels in $t_o = 251$ and (d) content popularity in $t_o = 251$

forecasted values of time slot $t = 251$ for user u_1 of cell c_1: $[1, 2, 14, 3, 2, 2]$ for the content $[14, 61, 112, 126, 149, 208]$, respectively, are obtained. Remind that these values need to be calculated for all users.

Next, using the proposed algorithm and prediction model, the heterogeneous content preferences of the users residing in all cells are captured. Thanks to LSTM, it can capture both the regional and global content popularity for all available content in the catalog. First, the histogram count, the activity level of the users, and the content popularity in a particular time slot are presented. Fig. 3.6 presents a sample result of this. Note that these values are captured for all optimization time slots. The results of some of the selected users from all cells are presented next to show the temporal dynamics over the time slots. Note

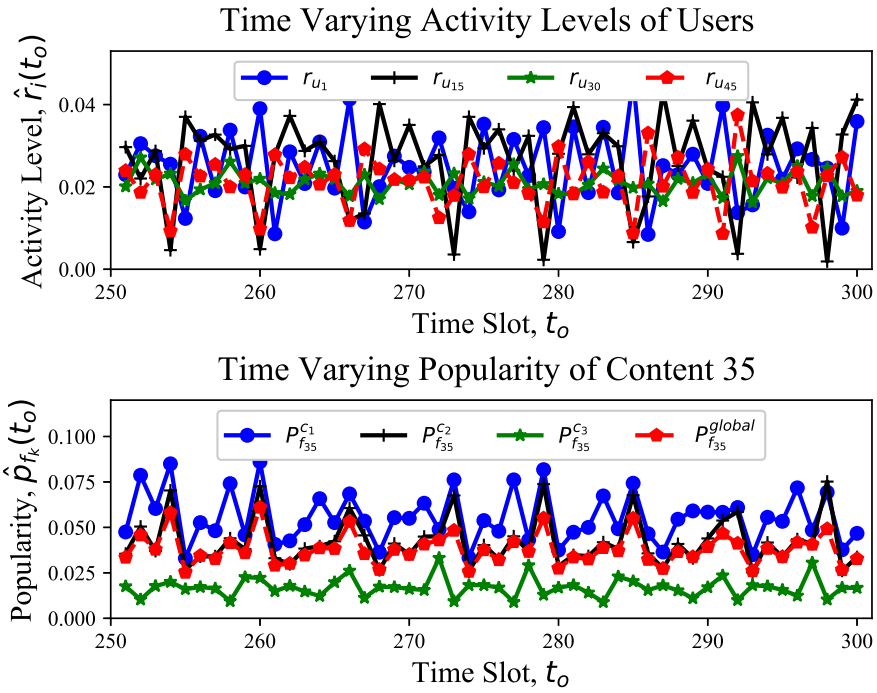


Fig. 3.7: Time varying nature of the content preferences and activity levels of the users

that the model capture all of these dynamics for all users and all contents in all time slots. Here, only a sample of how the popularity of the content and activity of the users changes over time are presented in Fig. 3.7.

The content preference probabilities $\left(\rho^{u_i}_{f_k}\right)$ of the users are first measured using these values. Then, the caching policy is designed using these results in the next sub-section.

3.5.2 Cache Placement

Using the time slot basis preferences (the joint probability that a user will request for a particular content given that she actually makes a request), first, the cache placement indicator functions are measured. Then, from that, the overall user preferences are calculated based on the equation (3.23). The caching placement into the local nodes is discussed in what follows.

Heterogeneous Caching Placement

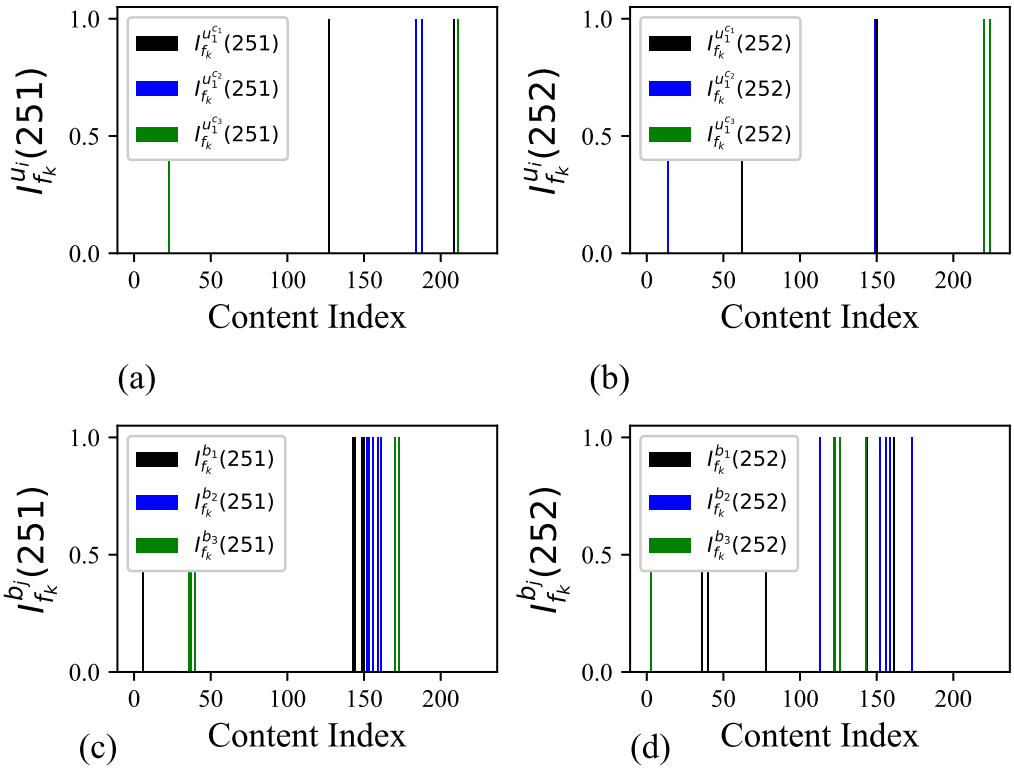


Fig. 3.8: Cache placement indicator functions in collaborative greedy caching - base station first (non-overlapping) case for $\mathcal{C}_d = 2$ and $C_b = 5$: (a) $\mathbb{I}_{f_k}^{u_i}(251)$, (b) $\mathbb{I}_{f_k}^{u_i}(252)$, (c) $\mathbb{I}_{f_k}^{b_j}(251)$ and (d) $\mathbb{I}_{f_k}^{b_j}(252)$

In the heterogeneous caching placement case, the cache enabled nodes stores different contents based on user preferences. First, the proposed algorithms are analyzed and evaluated in the following.

Collaborative greedy caching - base station first (non-overlapping): At first, the caching placement is presented based on Alg. 3. Recall that in this case, overlapping is not allowed for the stored content. The intention is to save as many unique contents as possible into the cache storage. The commonly preferred contents are stored fist. Then, the residuals - based on popularity. In a sense, thus, in this case, the popularity of content is compromised over the uniqueness and commonness of the contents. Fig. 3.8 presents the cache placement indicator functions for two different time slots at both, user level and base

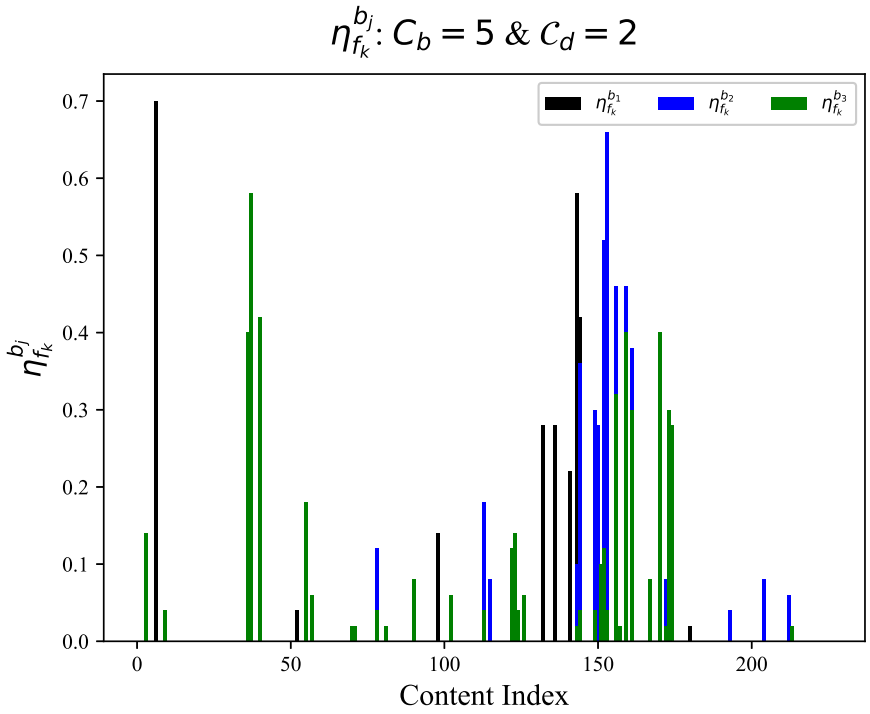


Fig. 3.9: Probability of storing content at the base stations in collaborative greedy caching - base station first (non-overlapping) for $C_b = 5$

station level. The figures suggest that similar content is not stored in two different places. That is, greedy caching - base station first (non-overlapping) ensured a maximum number of stored contents.

Then, using these indicator functions, with the help of equations (3.49-3.50), the long term caching probabilities are plotted. The result of that are presented in Fig. 3.9 and Fig. 3.10.

Collaborative greedy caching - user first (non-overlapping): In this case, the contents are placed at the user nodes first. The process is similar to the *Collaborative greedy caching - base station first (non-overlapping)* case. The difference is, however, the user nodes are given priority first. After storing the content at the UE level, the non-cached popular contents are stored at the base stations. In all cells, a similar approach is performed. Figs. 3.11, 3.12 and 3.13 represent the cache placement indicator functions, caching placement probabilities at the UE level and caching placement probabilities at the

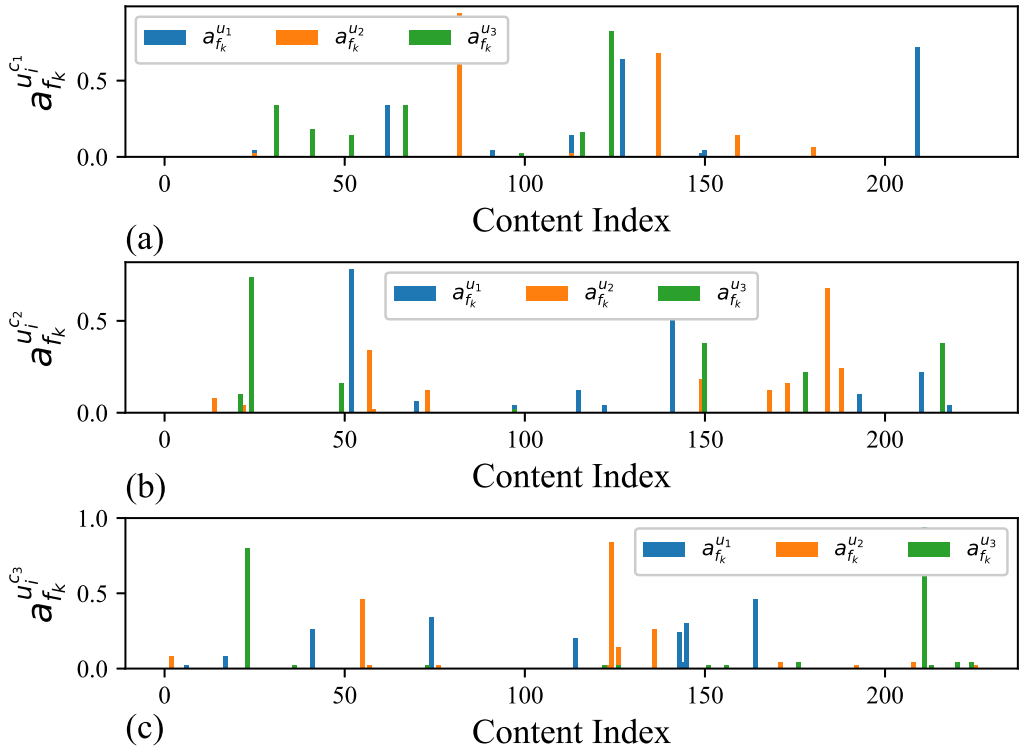


Fig. 3.10: Probability of storing content at the user level in collaborative greedy caching - base station first (non-overlapping) for $\mathcal{C}_d = 2$: (a) $a_{f_k}^{u_i^{c_1}}$ for user $u_1^{c_1}$, $u_2^{c_1}$ and $u_3^{c_1}$, (b) $a_{f_k}^{u_i^{c_2}}$ for user $u_1^{c_2}$, $u_2^{c_2}$ and $u_3^{c_2}$, and (c) $a_{f_k}^{u_i^{c_3}}$ for user $u_1^{c_3}$, $u_2^{c_3}$ and $u_3^{c_3}$

user level, respectively for the greedy caching - user first (non-overlapping) case.

Collaborative greedy overlapping caching: In this case, the preferred contents of the users are placed into its cache store first. Recall that the preference of a user is chosen randomly while the original request number is generated. Therefore, the choices of the users are different. However, it is possible to have a similar taste. As the cost of storing the content at different nodes are assumed to be equal, the noble intention of this proposed algorithm is to minimize the overall cost. Remind that the total cost for getting a content consists of the storage cost and the transmission cost. Therefore, the preferred contents of users are stored into its cache stores first. Then, if she has any space left, she will save the residual popular contents. After running this process for all the users in a cell, the

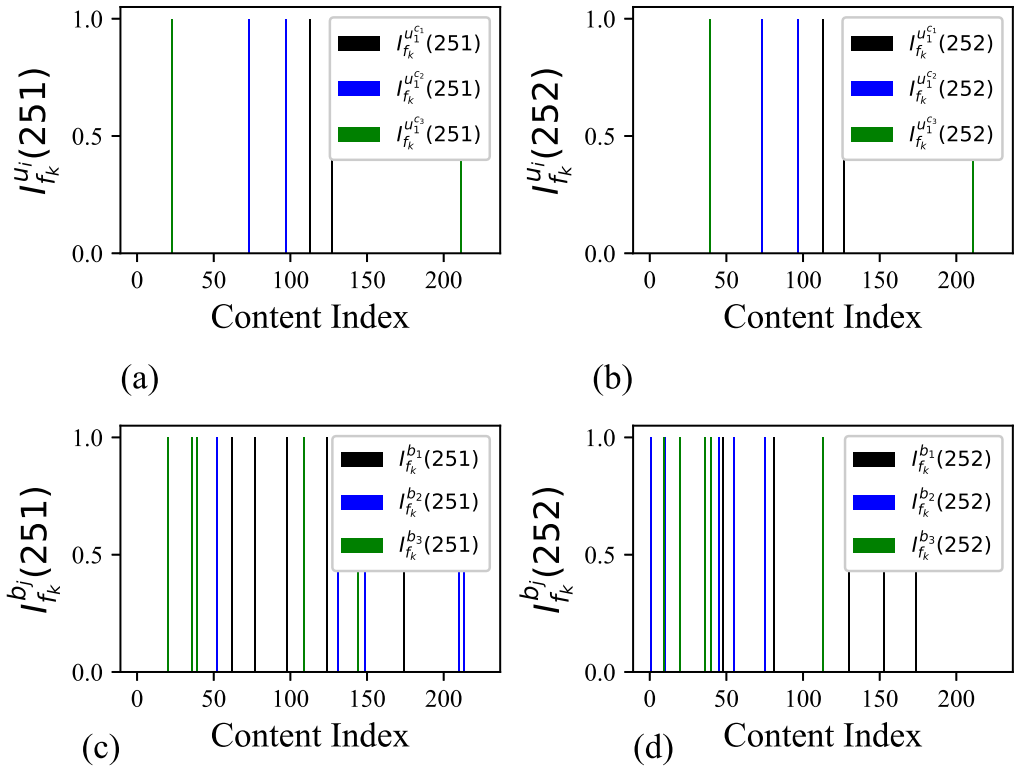


Fig. 3.11: Cache placement indicator functions in collaborative greedy caching - user first (non-overlapping) case for $\mathcal{C}_d = 2$ and $C_b = 5$: (a) $\mathbb{I}^{u_i}_{f_k}(251)$, (b) $\mathbb{I}^{u_i}_{f_k}(252)$, (c) $\mathbb{I}^{b_j}_{f_k}(251)$ and (d) $\mathbb{I}^{b_j}_{f_k}(252)$

cache storage of the base station is filled out with the remaining sorted contents. A similar thing is performed for all cell. Figs. 3.14, 3.15 and 3.16 represent the cache placement indicator functions, caching placement probabilities at the UE level and caching placement probabilities at the user level, respectively for the proposed greedy overlapping caching case.

Homogeneous Caching Placement

In the homogeneous caching placement case, all user nodes in the same cell store the same contents. The same copies of the contents are stored in all UE's cache storage, considering the cell (regional) content popularity, in this case. All base stations in a cluster also follow a similar fashion while placing the content. However, since all base stations

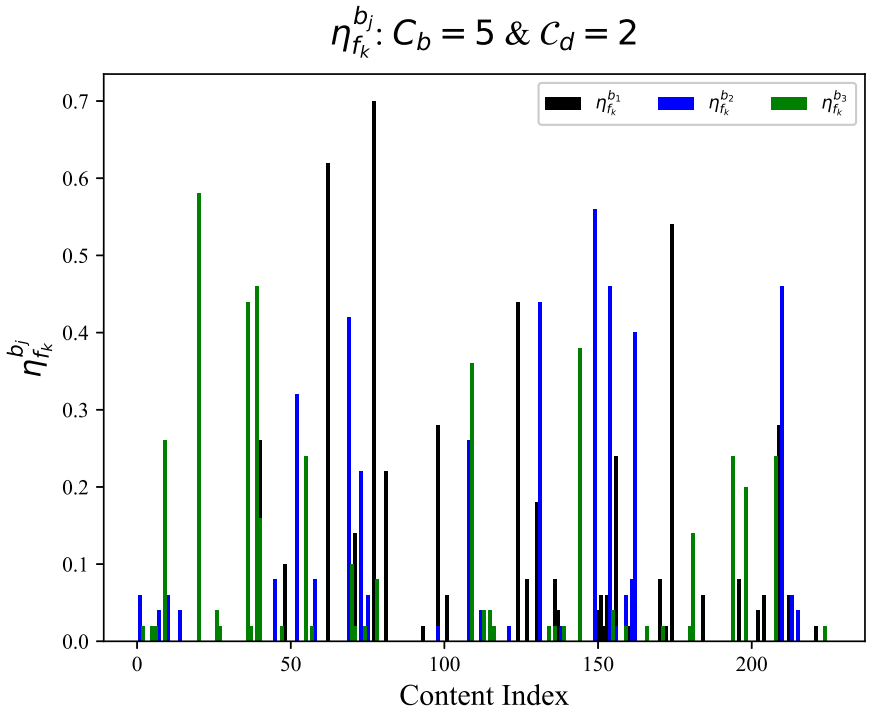


Fig. 3.12: Probability of storing content at the base stations in collaborative greedy caching - user first (non-overlapping) for $C_b = 5$

have to store the same copy of the contents, in the homogeneous caching placement, the global content popularity is chosen. Figs. 3.17, 3.18 and 3.19 present the cache placement indicator functions, long term storing probability at the user level, and base station level, respectively.

Next, the results obtained from both heterogeneous and homogeneous caching placements using the proposed algorithms are used to observe the cost functions in the following subsection.

3.5.3 Performance Analysis

Before going into the performance analysis of the proposed algorithms, first, a clear comparison between the legacy system's static estimation and the proposed LSTM based dynamic prediction has been performed. Note that if a static case is considered, for all time slots t_o essentially, there is no information about the temporal dynamics of the user pref-

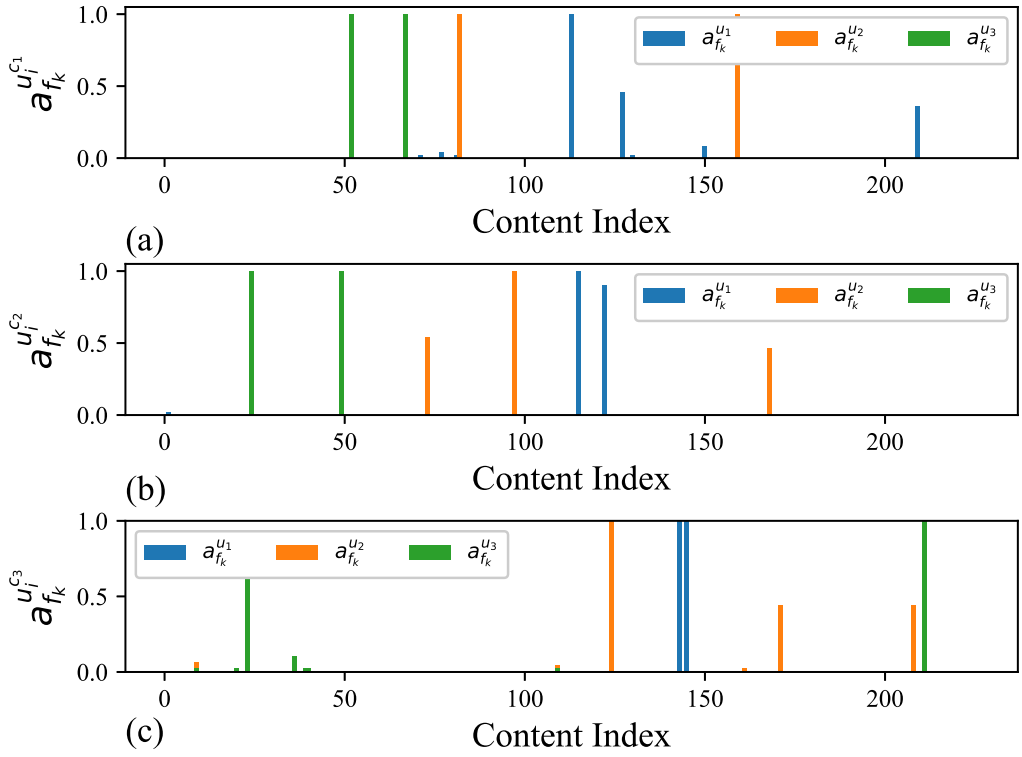


Fig. 3.13: Probability of storing content at the user level in collaborative greedy caching - user first (non-overlapping) for $\mathcal{C}_d = 2$: (a) $a_{f_k}^{u_i^{c_1}}$ for user $u_1^{c_1}$, $u_2^{c_1}$ and $u_3^{c_1}$, (b) $a_{f_k}^{u_i^{c_2}}$ for user $u_1^{c_2}$, $u_2^{c_2}$ and $u_3^{c_2}$, and (c) $a_{f_k}^{u_i^{c_3}}$ for user $u_1^{c_3}$, $u_2^{c_3}$ and $u_3^{c_3}$

erences and activity levels. Therefore, in all time slots, the caching placement probabilities are the same. However, if the proposed scheme is used, all of the temporal dynamics are well captured. Therefore, the system administrator knows precisely at what time, what contents might be requested by the users. Furthermore, the load coming from all of the users are also known to the system administrator. Therefore, the optimal caching placement can be performed based on the requirement. Fig. 3.20 shows a comparison between the static estimation and the proposed dynamic prediction. Note that, for simplicity and complexity, the comparison has been performed in the homogeneous case only. A similar trend should also be attained in the heterogeneous case.

From Fig 3.20, it is quite apparent that performing time slot based prediction performs

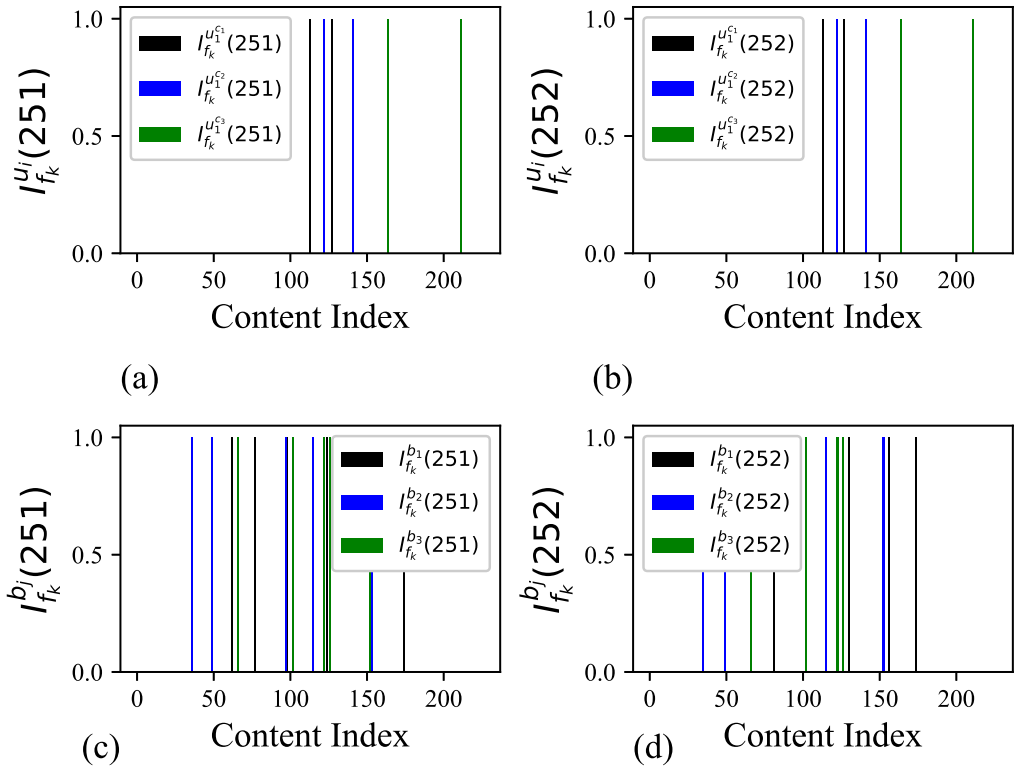


Fig. 3.14: Cache placement indicator functions in collaborative greedy overlapping caching case for $\mathcal{C}_d = 2$ and $C_b = 5$: (a) $\mathbb{I}_{f_k}^{u_i}(251)$, (b) $\mathbb{I}_{f_k}^{u_i}(252)$, (c) $\mathbb{I}_{f_k}^{b_j}(251)$ and (d) $\mathbb{I}_{f_k}^{b_j}(252)$

better than the static case. Therefore, the proposed LSTM model, and results obtained from it will be used for conducting various performance analysis from here on.

Now, the performances of the proposed algorithms are analyzed in the following. At first, the cost function is observed by keeping the UE cache size fixed and varying the BS cache size. As getting a requested content from the neighbor D2D nodes is lower than extracting it from the other nodes (BS or cloud), the performance will get affected if the most popular content is not stored at the user level first. Hence, it is expected that the proposed heterogeneous greedy overlapping caching policy to perform better than the other strategies. However, it is worth mentioning here that if the cache storage of the UEs is significantly small, the performance of the proposed greedy overlapping caching and greedy caching - user first, might be pretty similar. As in both cases, the most popular and

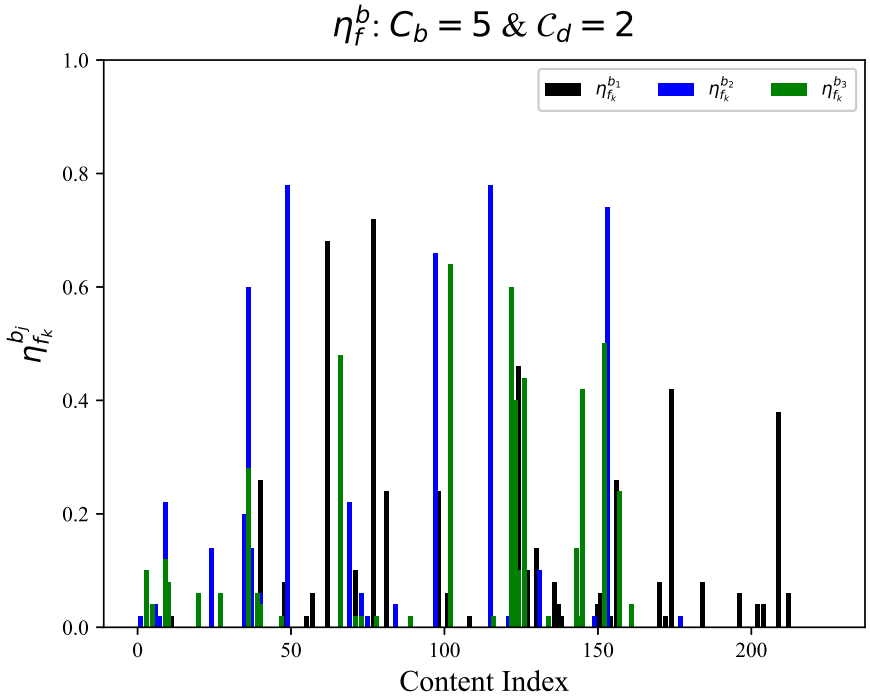


Fig. 3.15: Probability of storing content at the base stations in collaborative greedy overlapping caching for $C_b = 5$

common contents for the users are stored into either the requester self cache storage or nearby neighbor nodes first. This does mean that the cost, is only due to the storage cost (if stored in the own cache store) or a small transmission cost for obtaining from the neighbors.

The obtained results from the simulation also validate this point. In Fig. 3.21, a fixed $\mathcal{C}_d = 2$ is used while the BS cache size is varied. As expected, both of the proposed greedy overlapping and greedy - user first, results in considerably similar performances. While the performance of the greedy caching - BS first is the lowest among these three proposed algorithms. However, notice that if the BS cache size is supposed to keep increasing, at some point, it is visible that the performance gaps in these three cases are becoming quite similar. On the contrary, the performance of the homogeneous cache placement is, as expected, pretty lower than all of the considered heterogeneous caching placement cases.

In Fig. 3.22, a fixed $\mathcal{C}_d = 4$ is used while the BS cache size is varied in between 4

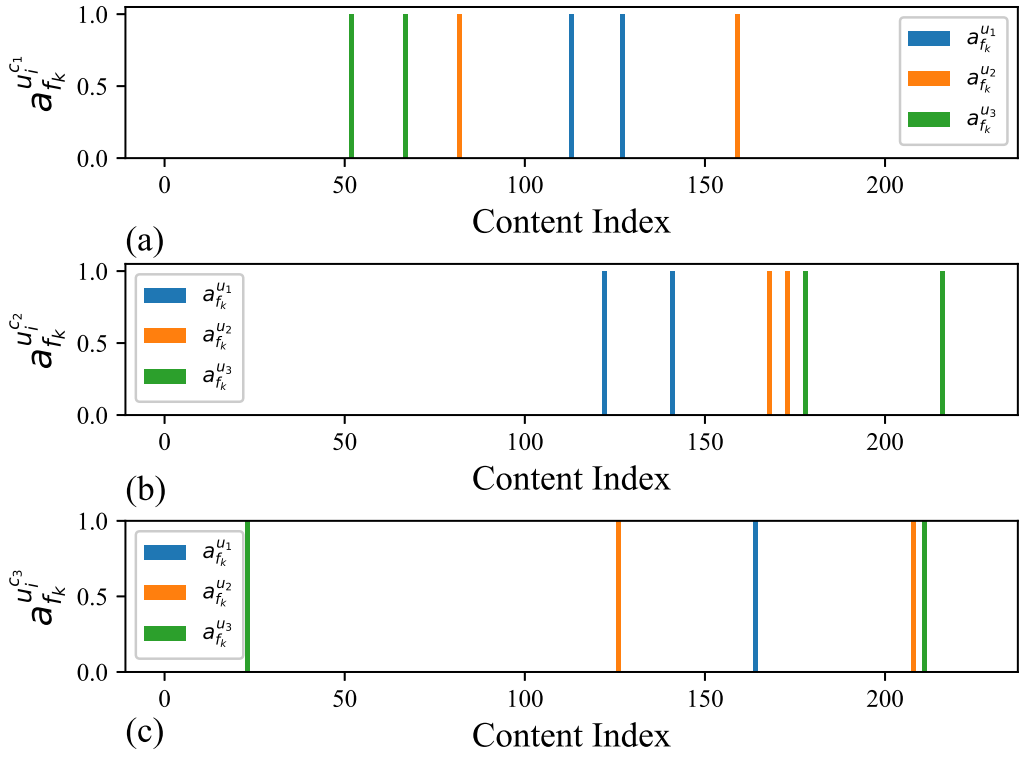


Fig. 3.16: Probability of storing content at the user level in collaborative greedy overlapping caching for $\mathcal{C}_d = 2$: (a) $a_{f_k}^{u_i^{c_1}}$ for user $u_1^{c_1}$, $u_2^{c_1}$ and $u_3^{c_1}$, (b) $a_{f_k}^{u_i^{c_2}}$ for user $u_1^{c_2}$, $u_2^{c_2}$ and $u_3^{c_2}$, and (c) $a_{f_k}^{u_i^{c_3}}$ for user $u_1^{c_3}$, $u_2^{c_3}$ and $u_3^{c_3}$

to 14. As mentioned earlier, the proposed greedy overlapping caching placement performs significantly better than all the other cases. However, a critical observation from this figure is when BS cache size increases, the performance of the collaborative greedy caching - base station first (non-overlapping) is better than that of collaborative greedy caching - user first (non-overlapping). For example, when $C_b = 13$, the performance of these two algorithms are crossing over. When $C_b = 14$, the performance of the collaborative greedy caching - base station first (non-overlapping) case is visibly better than the later one. This is because of the fact that when the C_b is increasing, more contents can be stored at the BS level first. Then, the rest of the popular contents are stored at the user node. While performing the cache placement, notice that in Alg. 3, the user's preferred leftover contents are being

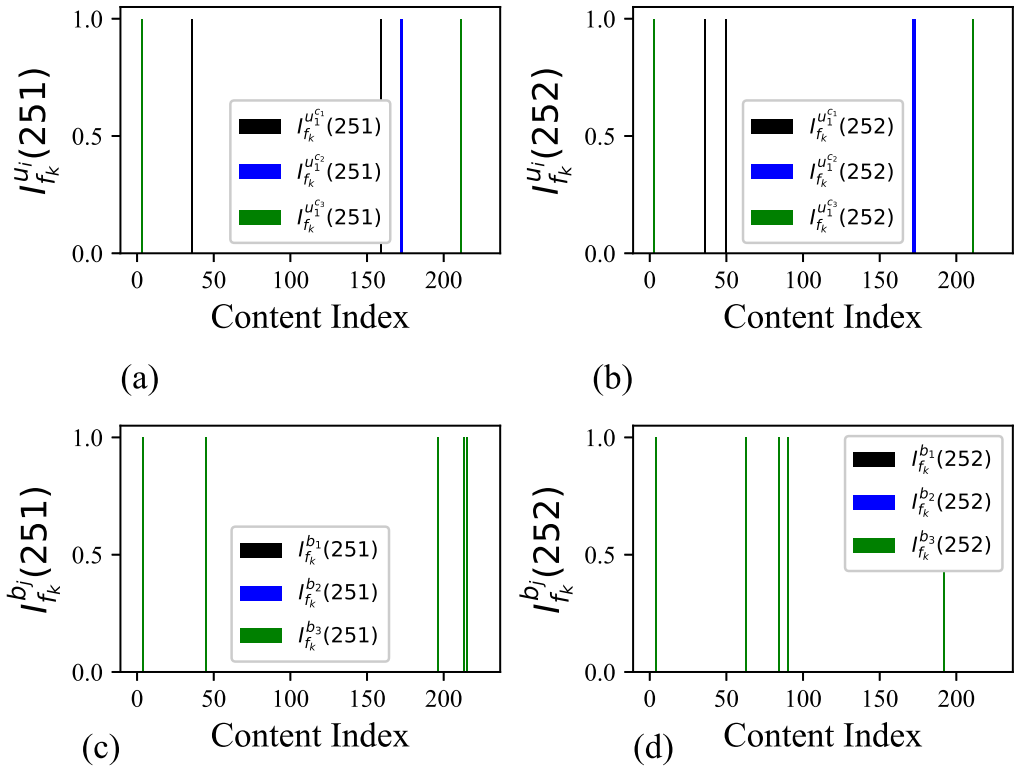


Fig. 3.17: Cache placement indicator functions in collaborative homogeneous caching case for $\mathcal{C}_d = 2$ and $C_b = 5$: (a) $\mathbb{I}^{u_i}_{f_k}(251)$, (b) $\mathbb{I}^{u_i}_{f_k}(252)$, (c) $\mathbb{I}^{b_j}_{f_k}(251)$ and (d) $\mathbb{I}^{b_j}_{f_k}(252)$

stored at the respective user node first. Therefore, more contents are being stored closer to the user nodes. Whereas in collaborative greedy caching - user first (non-overlapping) case, the common contents are being stored at the user node first and then the BS cache storage is being filled out. As the user cache size is fixed in this figure, each user is storing a fixed number of contents while the user may not store its own preferred contents in its cache storage as the common contents that are being requested by all users in the respective cell have to be stored first. However, the proposed collaborative greedy overlapping caching algorithm outperforms all the others in this case as the user cache storage size is at a moderate level and the preferred contents of a user are being stored at its self cache storage first.

Next, the cost function is observed by keeping the BS cache size fixed while the UE

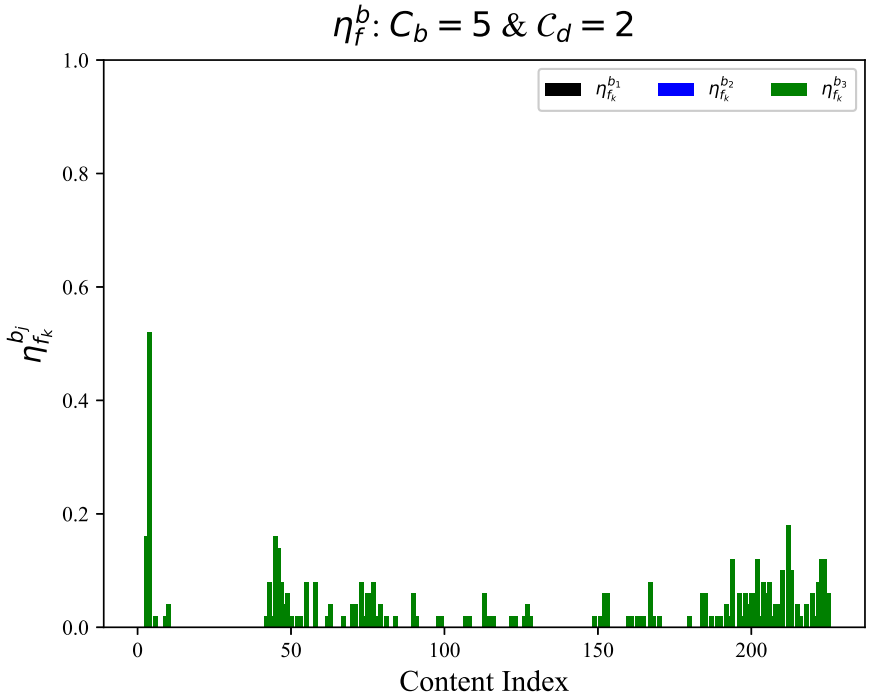


Fig. 3.18: Probability of storing content at the base stations in collaborative homogeneous caching for $C_b = 5$

cache size is varied in between 1 to 5. Note that, usually, the user cache size is much lower than that of the BS. However, here, this range is analyzed to show the impact of the cache size of the BS. Since the cost of getting the content from the neighboring D2D nodes is much lower than getting it from the BS or cloud, it is expected that the performance of the proposed collaborative greedy overlapping algorithm will be better and distinguishable when the UE cache size is relatively moderate to larger. While the proposed algorithms should perform excellently in the heterogeneous case, on the contrary, homogeneous caching is expected to perform worse than any of the three proposed algorithms.

In Fig. 3.23, a fixed $C_b = 8$ is used while the users cache size, C_ds are varied in between 1 to 5. When C_ds are lower, the performance of the greedy - (a) user first and (b) overlapping caching performs nearly similar. This is due to the fact that, in both cases, a UE's most preferred contents are first stored into its cache storage. However, as the cache size of the user increases, as expected, the proposed greedy overlapping caching policy performs better

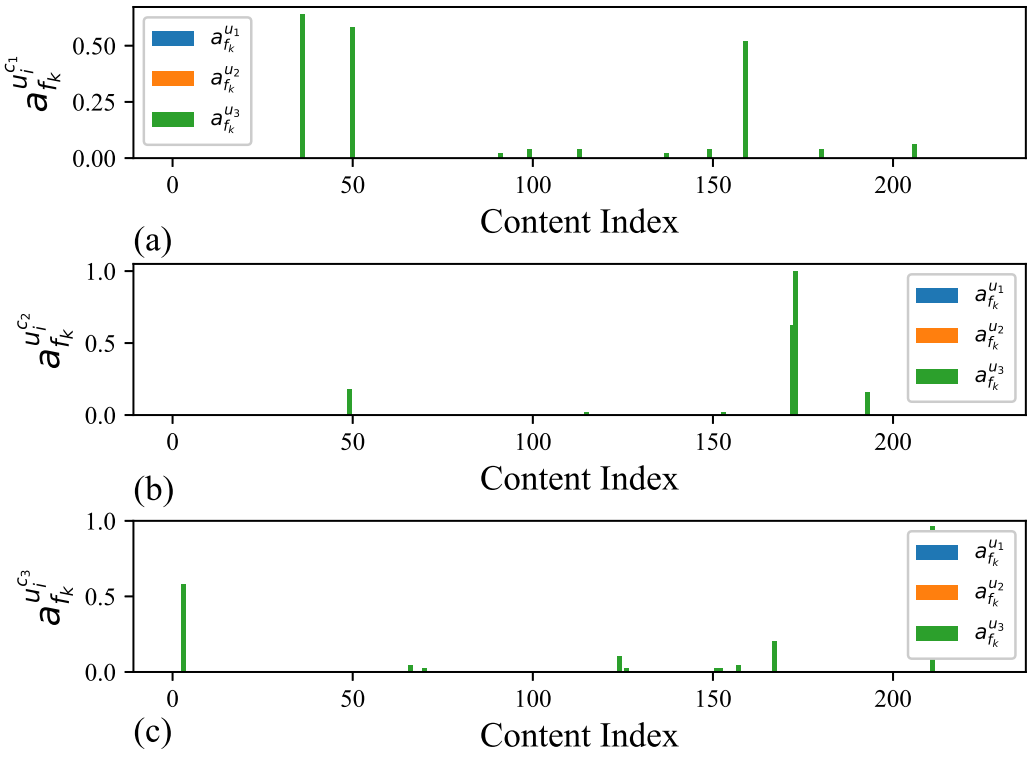


Fig. 3.19: Probability of storing content at the user level in homogeneous caching for $\mathcal{C}_d = 2$: (a) $a_{f_k}^{u_i^{c_1}}$ for user $u_1^{c_1}$, $u_2^{c_1}$ and $u_3^{c_1}$, (b) $a_{f_k}^{u_i^{c_2}}$ for user $u_1^{c_2}$, $u_2^{c_2}$ and $u_3^{c_2}$, and (c) $a_{f_k}^{u_i^{c_3}}$ for user $u_1^{c_3}$, $u_2^{c_3}$ and $u_3^{c_3}$

than all of the other cache placement strategies. Fig. 3.24 also validates the claim with a fixed BS cache size, $C_b = 12$. Notice a similar critical observation, as described in Fig. 3.22, in Fig. 3.24 when the cache size of the user is approximately 4.5 for the collaborative greedy - (1) BS first (non-overlapping) and (2) user first (non-overlapping) cases. Whereas, as deemed, the proposed greedy overlapping caching algorithm outperforms the others.

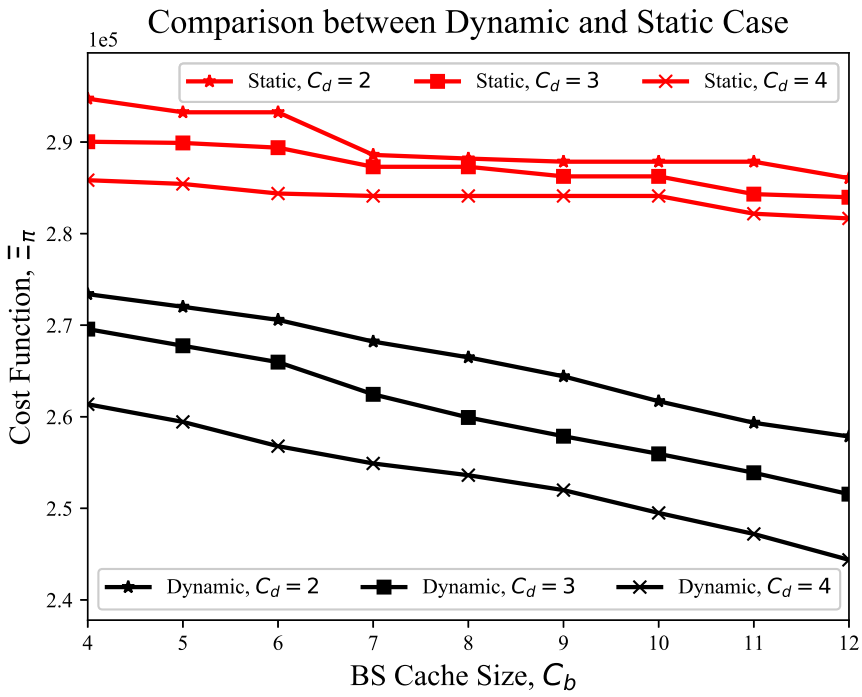


Fig. 3.20: Comparison between static and dynamic cases for different C_bs size with fixed UE cache size $\mathcal{C}_d$

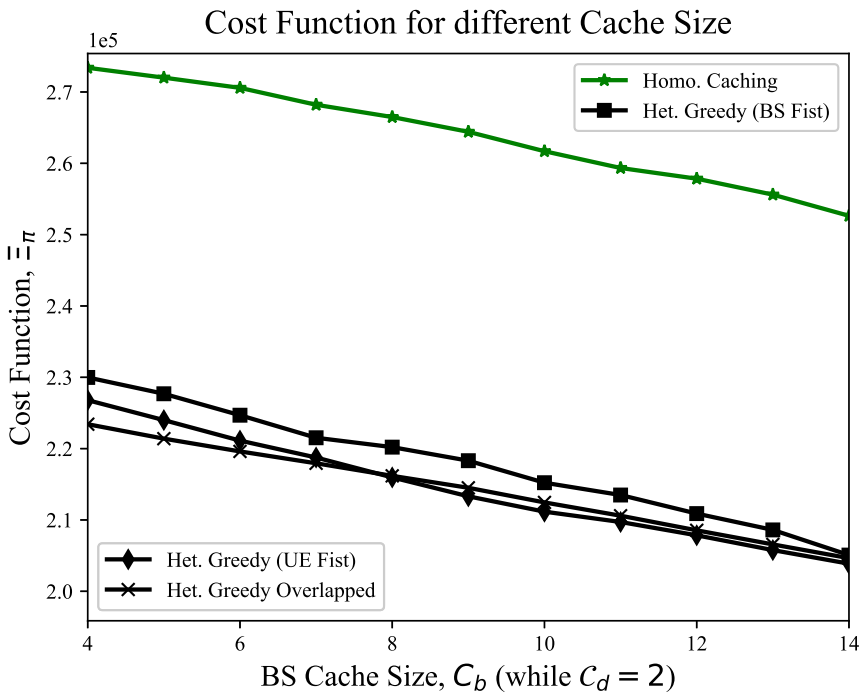


Fig. 3.21: Cost functions for different C_b size with fixed UE cache size of $C_d = 2$

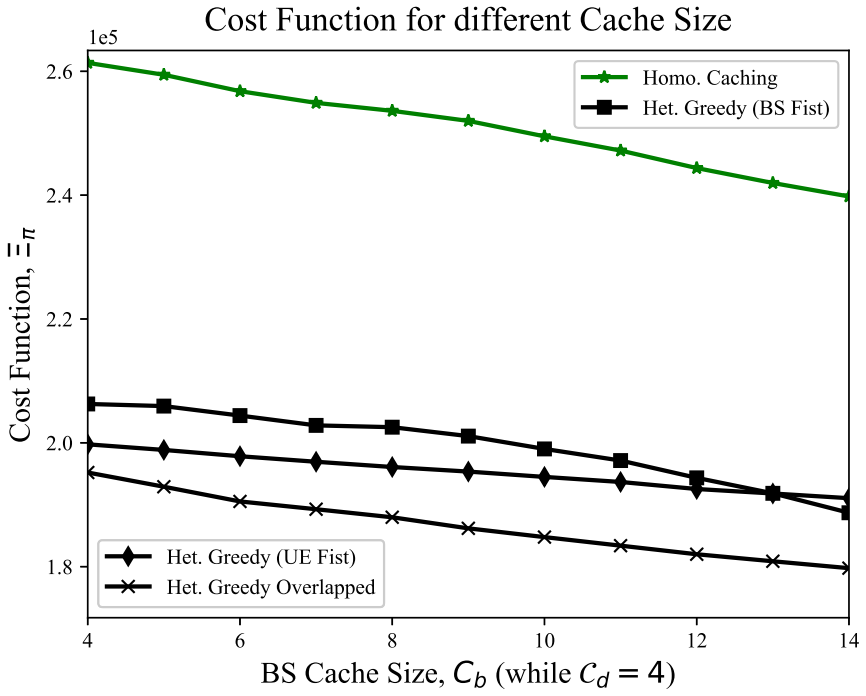


Fig. 3.22: Cost functions for different C_b size with fixed UE cache size of $C_d = 4$

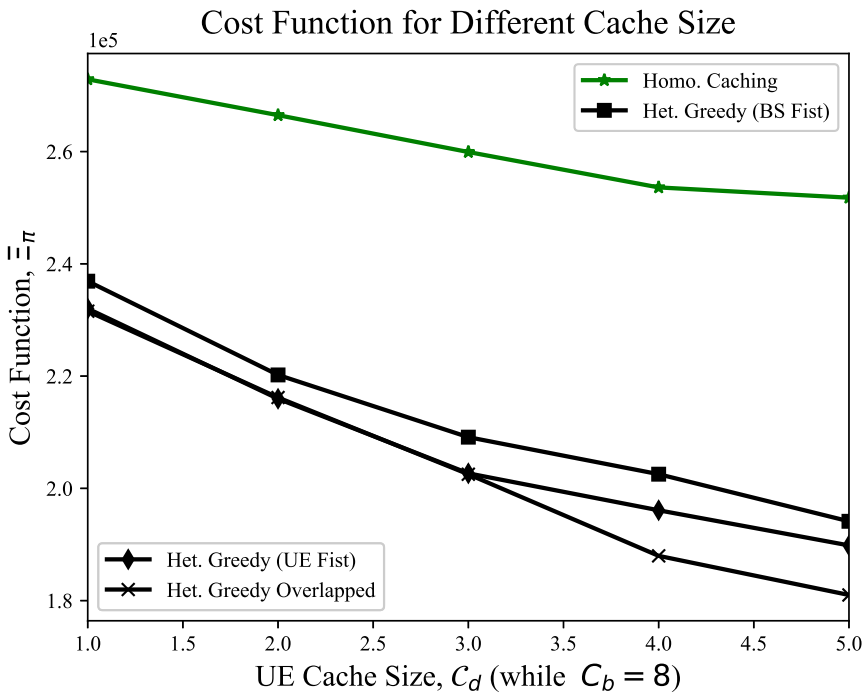


Fig. 3.23: Cost functions for different C_d size with fixed BS cache size of $C_b = 8$

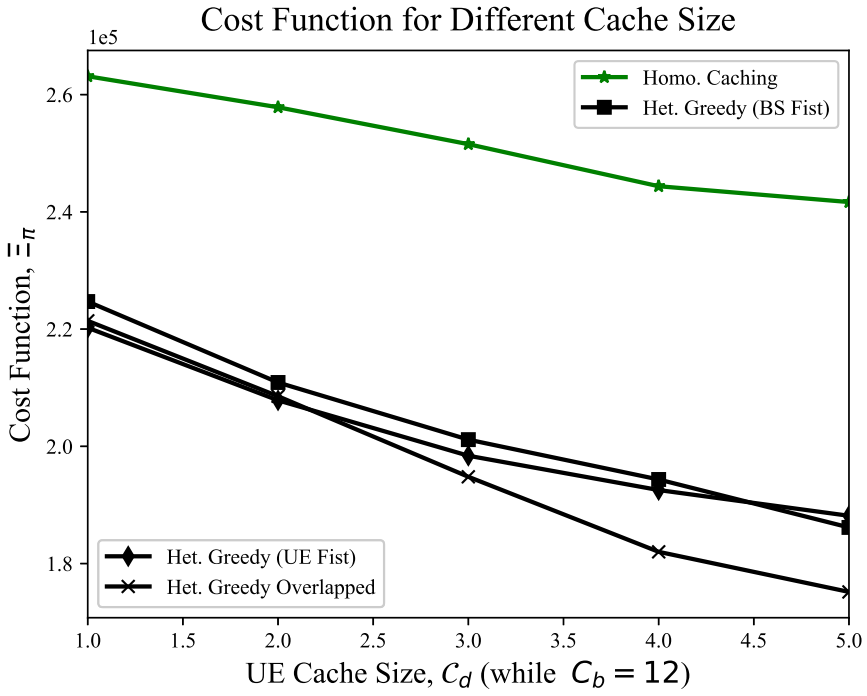


Fig. 3.24: Cost functions for different C_d size with fixed BS cache size of $C_b = 12$

CHAPTER 4

Collaborative Edge Caching at Spatially Distributed Full-Duplex Edge Nodes

In this chapter, a more practical system model is incorporated. All nodes are assumed to be spatially distributed. Following the actual communication scenarios, a typical complex heterogeneous network platform has been adopted for evaluating the caching performances by placing the contents at the edge nodes.

4.1 System Model and Problem Definition

This section presents the nodes distributions followed by the caching properties and concise problem description.

4.1.1 Nodes Distributions

A practical two-tier heterogeneous network (HetNet) is considered in this chapter. The system model consists of macro base stations (MBS) and low-power small base stations(sBSs) or relays, associated with underlay D2D communications. All of these nodes are drawn following independent homogeneous Poisson point processes (HPPP). The densities of the nodes are denoted by λ_u, λ_b and λ_m, respectively for D2D users, sBS and MBS. To fully exploit the advantages of edge caching, it is assumed that both sBS and MBS operate in in-band full-duplex (IBFD) modes. In other words, they use the same frequency to transmit and receive [49]. However, the users are assumed to be operated in half-duplex (HD) mode. An open access case for all base stations, i.e., all users are allowed to get connected to all sBSs and MBSs, if the criteria of user associations are fulfilled. The communication ranges of the BSs are denoted by R_b and R_m for sBS and MBS, respectively, while R_u denotes the communication range of the D2D users. The set of users, sBS and MBS are denoted by $u \in \{\mathcal{U}\}$, $b \in \{\mathcal{B}\}$ and $m \in \{\mathcal{M}\}$. A snapshot of such a realization is shown in Fig. 4.1.

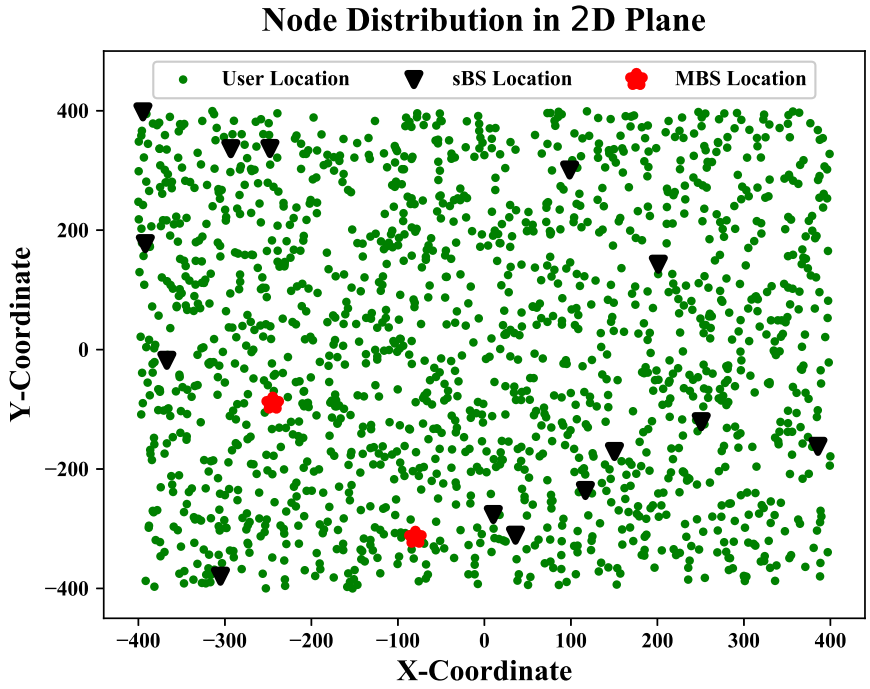


Fig. 4.1: Node distribution in $2D$ plane with $\lambda_u = 2.5^{-3}$ (per m^2), $\lambda_b = 2.5^{-5}$ (per m^2) and $\lambda_m = 1.56^{-7}$ (per m^2)

The requester user node is denoted as tagged user node. Apart from always being associated with the serving MBS, a user can also get connected with the low powered sBS (or relays), if the association rules are satisfied. The associated sBS is denoted as the tagged sBS for that user. If such a tagged sBS exist for a user, it will maintain its communication with the serving MBS via the tagged sBS. In that case, the sBS can also use its FD mode to deliver requested content from the other sBSs or the cloud via the MBS. If such a tagged sBS does not exist for a user, the user will have to rely on the neighbor nodes and the serving MBS for extracting a content that she wants but not stored in her cache store. Besides, a α portion of the users are assumed to serve as tagged users, i.e., can request for contents - serve as the receiver, and the remaining $(1 - \alpha)$ portion of the users act as the transmitters.

Furthermore, for a successful communication among different nodes (such as - between two D2D users), the channel quality between the nodes has to satisfy a certain predefined

threshold. However, the link level optimization is out of the scope of this thesis. The underlying assumption made in this chapter is that the nodes can communicate with other nodes if they are in each others communication range.

4.1.2 Cache Storage and Caching Policy

The cache storage of all nodes i.e. users, sBS, and MBS are considered to have an equal cache size denoted by $\mathcal{C}_d, \mathcal{C}_b$, and $\mathcal{C}_m$, respectively. Considering equal sized contents, denoted by S_f, in the proposed model, the users can make request for content from a content directory of $\mathcal{F} = \{f_k\}$, where $k \in \{1, 2, \ldots, F\}$. The users' requests are modeled using Zipf type distribution. Note that a heterogeneous content preference of all users is considered. For the caching model, a probabilistic strategy is considered assuming heterogeneous caching placement strategy. Let $\eta_{f_k}^{u_i}$, $\eta_{f_k}^{b_j}$ and $\eta_{f_k}^{m_l}$ be the probabilities of storing a content $f_k \in \{\mathcal{F}\}$ at the cache store of the user node u_i, sBS b_j, and MBS m_l, respectively.

4.1.3 Proposed Content Access Protocol

For accessing the contents, several practical cases have been considered. The considered instances are listed in the following.

Case 1 - Local/self cache hit

If a requested content is already stored into the requester's self cache storage, the requester can access that content from there. This scenario is noted as a self/local cache hit.

Case 2 - D2D cache hit

If a requested content is not stored in the tagged node's storage, the content is searched in the neighbor D2D nodes first. If one of the neighbors - that are in the communication range of the requester node, the requested content is delivered from that neighbor node. This case is denoted as the D2D cache hit.

Case 3 - sBS cache hit

If the tagged user is under the communication range of a sBS, it will maintain its communication via the tagged sBS. In this case, we have the following sub-cases:

- *Case 3.1*: If the requested content is in the tagged sBS cache store, it can access the content directly from there. We denote this case as a direct cache hit from the tagged sBS.

- *Case 3.2*: If the content is not stored in the tagged sBS cache store but available in one of the neighbor sBSs, the tagged sBS first extract the content from the neighbor sBS via its FD capability. Then it deliver the content from to the tagged user. We denote this term as soft-sBS (SsBS) cache hit.

- *Case 3.3*: If the requested content is not available in any of the sBSs, the tagged sBS forward the request to the serving MBS. If the content is in the serving MBS, it is delivered to the tagged sBS and then to the user. This case is denoted as the sBS-MBS cache hit.

- *Case 3.4*: If all of the above sub-cases fail, then the content is extracted from the cloud. First, the MBS extract the content from the cloud using its FD capability. Then, the sBS extract the content from the MBS using its own FD capability. Finally, the tagged sBS deliver the content to the tagged user. This term is denoted as sBS cache miss.

Note that if a tagged user is under the communication range of multiple sBSs, it gets connected to the one that provides best received power to it. However, since the link level optimization is not performed, it is assumed that the user follows the closest distance rule for the association.

If *Case 3* is true for a tagged user, it is represented by an indicator function $\mathbb{I}_s \in \{0, 1\}$. If $\mathbb{I}_s = 1$, then the tagged user is under the communication range of at least one sBS. Else it will fall into *Case 4*.

Case 4 - MBS cache hit

If the tagged user is not in the communication range of any of the sBS, it has to rely on the neighbors and the serving MBS for its communication. In this case, we consider the following sub-cases:

- *Case 4.1*: If the requested content is available in the MBS cache store, the content is directly delivered to the tagged user. This case is denoted as a MBS cache hit.
- *Case 4.2*: If the content is not available in the MBS cache store and the above case fail, the MBS extract the content from the cloud using its FD capability. Then the content is directly delivered to the user. This case is referred as a MBS cache miss.

Without loss of generality, this case - *(Case 4)* is denoted by the indicator function $\mathbb{I}_m = \{0, 1\}$, where $\mathbb{I}_m = 1$ if $\mathbb{I}_s = 0$.

4.1.4 Problem Definition

Like the preceding chapter, the main goal of this chapter is also delivering the as many requested contents as possible from the edge nodes. However, the spatial node distribution has been incorporated in this chapter. Considering practical node distribution, the content sharing cost is analyzed. The main intention of incorporating spatial edge node is to observe the actual performance of edge caching in a real platform.

In the next section, the probability of cache hit and cost calculation are analyzed. Fig. 4.1 and Table 4.1 represent the proposed system model and symbols used in this paper, respectively.

4.2 Full Duplex Edge Caching: Cost Analysis

Before diving into the cost analysis, different necessary terms are introduced in the following.

Table 4.1: List of Symbols

Symbol	Description
$\mathcal{F}$	Content catalog / library
$\mathcal{U}$	Set of user nodes (D2D Nodes)
$\mathcal{B}$	Set of small base stations
$\mathcal{M}$	Set of macro base stations
α	portion of users act as a receiver
$\mathcal{C}_d$, $\mathcal{C}_b$, $\mathcal{C}_m$	Cache storage of D2D node, sBS and MBS, respectively
$\rho_{f_k}^{u_i}$	Probability that user u_i requests content f_k
$\eta_{f_k}^{u_i}$, $\eta_{f_k}^{b_j}$, $\eta_{f_k}^{m_l}$	Probability that user u_i, sBS b_j and MBS m_l store the content f_k, respectively
$\mathbb{I}_s$, $\mathbb{I}_m$	Indicator function that a user is associated with a sBS and a MBS, respectively
ϕ_l	Content sharing cost from node l

4.2.1 Content Popularity

Content popularity, in this chapter, is modeled following the Zipf distribution as shown in equation (3.1). Recall that skewness, γ, governs this distribution. In this chapter, it is assumed that each user will have a different content preference. Therefore, each users content preference order and the parameter are randomly chosen. While the content order is chosen using random permutation, the parameter, γ, is chosen following Uniform random distribution within a maximum and a minimum range. Without any loss of generality, the probability that user u_i will request for content f_k is denoted by $\rho_{f_k}^{u_i}$. This is modeled based on the Zipf distribution.

4.2.2 Cache hit probabilities

Firstly, the cache hit probabilities at different nodes for the above cases are analyzed. Note that a cache hit occurs at a node if a requested content is available in that node.

Case 1 - Local/self cache hit

The local cache hit probability is denoted as $\mathrm{P_o^u} = \eta_{f_k}^{u_0}$ i.e. the probability of storing the content f at the self cache storage of the tagged user.

Case 2 - D2D cache hit

The cache hit probability by the D2D nodes can be calculated as follows:

$$\mathrm{P}_d^u = \left(1 - \eta_{f_k}^{u_0}\right) \left[1 - \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right)\right], \tag{4.1}$$

where $\prod_{u_i \in \Phi_u} \left(1 - \eta_{f_k}^{u_i}\right)$ means that none of the Φ_u active neighbors (D2D nodes) in its communication range have the content. Thus, the complement of that is the probability that at least one of the user has stored the content.

Case 3 - sBS cache hit

In this case, all of the cache hit probabilities that can be achieved via the tagged sBS have to be calculated. In the following, each of these sub-cases under this case is calculated.

At first, the probability of getting a requested content from the tagged sBS is calculated. This sub-case is termed as *Case 3.1.*

$$\mathrm{P}_{b_o}^u = \left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \eta_{f_k}^{b_0}. \tag{4.2}$$

Then, the probability of getting a content from one of the neighbor sBSs given that the tagged sBS does not have the content is considered. This is essentially saying a cache miss has occurred at the tagged sBS i.e. $\mathbb{I}_{b_0} = 1$. The tagged sBS now needs to initiate its FD capability to extract the content from the neighbors. Firstly, it will look for the content in the neighbors sBSs in its communication range and if found, the neighbor will deliver the content to the requester. This case is named as soft sBS cache hit and termed as *case 3.2.* Mathematically, this can be expressed as follows:

$$\mathrm{P}_B^u = \left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \left(1 - \eta_{f_k}^{b_0}\right) \left(1 - \prod_{b_j \in \Phi_b \setminus b_0} \left(1 - \eta_{f_k}^{b_j}\right)\right), \tag{4.3}$$

where Φ_b is the set of active neighbor sBSs that are in the communication range of the tagged sBS.

If the requested content is not stored any of the neighbors' nodes, the request is forwarded to the serving MBS via the tagged sBS. Therefore, this case is calculated as follows:

$$\mathrm{P}^{u}_{M_{\mathbb{I}_s}} = \left(1-\eta^{u_0}_{f_k}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1-\eta^{u_i}_{f_k}\right) \left(1-\eta^{b_0}_{f_k}\right) \prod_{b_j \in \Phi_b \setminus b_o} \left(1-\eta^{b_j}_{f_k}\right) \eta^{m_0}_{f_k}. \tag{4.4}$$

When $\mathbb{I}_s = 1$, from the above cases, the total cache hit probability can be calculated as

$$\begin{aligned} \mathrm{P}^{\mathbb{I}_s}_{l} &= \eta^{u_0}_{f_l} + \mathrm{P}^{u}_{d} + \mathrm{P}^{u}_{b_0} + \mathrm{P}^{u}_{B} + \mathrm{P}^{u}_{M_{\mathbb{I}_s}} \\ &= 1 - \left[\left(1-\eta^{u_0}_{f_k}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1-\eta^{u_i}_{f_k}\right) \left(1-\eta^{b_0}_{f_k}\right) \prod_{b_j \in \Phi_b \setminus b_0} \left(1-\eta^{b_j}_{f_k}\right) \right] \left(1-\eta^{m_0}_{f_k}\right). \end{aligned} \tag{4.5}$$

Now, if the content is not even stored in the MBS cache store, it has to be downloaded from the cloud. This case is termed as a cache miss via both sBS and MBS. In this case, the MBS initiates its FD mode and download the content from the cloud. Therefore, the cache miss probability is calculated from equation (4.5) as follows:

$$\begin{aligned} \mathrm{P}^{u}_{C_{\mathbb{I}_s}} &= 1 - \mathrm{P}^{\mathbb{I}_s}_{l} \\ &= \left[\left(1-\eta^{u_0}_{f_k}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1-\eta^{u_i}_{f_k}\right) \left(1-\eta^{b_0}_{f_k}\right) \prod_{b_j \in \Phi_b \setminus b_0} \left(1-\eta^{b_j}_{f_k}\right) \right] \left(1-\eta^{m_0}_{f_k}\right) \end{aligned} \tag{4.6}$$

Case 4 - MBS cache hit

Recall that *case 4* is only considered if the tagged user is not under the coverage region of any of the sBSs, i.e. $\mathbb{I}_s = 0$. Therefore, for a tagged user, the option of getting a content locally is limited to the availability of the content at (1) the local/self-store, (2) the neighbor D2D node's cache store and (3) the serving MBS's cache store.

Firstly, the cache hit probability at the MBS when $\mathbb{I}_m = 1$ (i.e. $\mathbb{I}_s = 0$) is calculated in the following.

$$\mathrm{P}^{u}_{M_{\mathbb{I}_M}} = \left(1-\eta^{u_0}_{f_k}\right) \prod_{u_i \in \Phi_u \setminus u_i} \left(1-\eta^{u_i}_{f_k}\right) \eta^{m_0}_{f_k}. \tag{4.7}$$

Then, in this case, when a tagged user is only connected to the serving MBS, the total local cache hit probability is calculated as follows:

$$\mathrm{P}_{l}^{\mathbb{I}_m} = \mathrm{P}_{\mathrm{o}}^{\mathrm{u}} + \mathrm{P}_{d}^{u} + \mathrm{P}_{M_{\mathbb{I}_m}}^{u} = 1 - \left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \left(1 - \eta_{f_k}^{m_0}\right). \tag{4.8}$$

Hence, the cache miss probability in this case is calculated similarly as follows:

$$\mathrm{P}_{C_{\mathbb{I}_m}}^{u} = \left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \left(1 - \eta_{f_k}^{m_0}\right). \tag{4.9}$$

4.2.3 Cost Analysis

Like the preceding chapter, both transmission and storage cost has been considered in this chapter. Storage cost is deemed to be equal for all nodes, and it is denoted by Λ^{stor}. The cost of getting a requested content from node j is denoted by $\phi_j = \Lambda_j^{\mathrm{stor}} + \Lambda_j^{\mathrm{com}}$, where Λ_j^{com} is the communication cost. Recall that user's preference for a content is denoted by $\rho_{f_k}^{u_i}$. The cost model considered in this chapter is based on the notion that a user can request all the contents available in the content catalog. Contents can be delivered to the tagged user from different nodes. The goal is to design a heterogeneous caching model to minimize the content sharing cost. Considering the above-mentioned content access protocols, the cost of accessing the contents for a user is calculated in (4.11).

Now, the objective function is presented, aiming to minimize this content sharing cost.

$$\mathbf{P_1}: \quad \underset{\eta_{f_k}^{u_i}, \eta_{f_k}^{b_j}, \eta_{f_k}^{m_l}}{\text{minimize}} \quad \Xi_{\pi} \tag{4.10a}$$

$$\text{s. t.} \sum_{k=1}^{F} \eta_{f_k}^{u_i} \leq \mathcal{C}_u, \quad \forall\ u_i \in \{\mathcal{U}\}, f_k \in \{\mathcal{F}\} \tag{4.10b}$$

$$\sum_{k=1}^{F} \eta_{f_k}^{b_j} \leq \mathcal{C}_b, \quad \forall\ b_j \in \{\mathcal{B}\}, f_k \in \{\mathcal{F}\} \tag{4.10c}$$

$$\sum_{k=1}^{F} \eta_{f_k}^{m_l} \leq \mathcal{C}_m, \quad \forall\ m_l \in \{\mathcal{M}\}, f_k \in \{\mathcal{F}\} \tag{4.10d}$$

$$0 \leq \eta_{f_k}^{u_i} \leq 1,\ 0 \leq \eta_{f_k}^{b_j} \leq 1,\ 0 \leq \eta_{f_k}^{m_l} \leq 1, \tag{4.10e}$$

where the constraints in (4.10b-4.10d) ensure the physical storage size limitations of user node, sBS and MBS, respectively, while the constraints in (4.10e) are due to the probability range in $[0, 1]$.

Due to a large number of system parameters, problem P_1 is extremely hard to solve. Therefore, in the next section, an artificial intelligence (AI) based solution is proposed.

$$\begin{aligned}
\Xi_\pi &= \sum_{k=1}^{F} \rho_{f_k}^{u_0} \Bigg[\Lambda^{\mathrm{stor}} \eta_{f_k}^{u_0} + \phi_d \mathrm{P}_d^u + \left(\phi_{b_0} \mathrm{P}_{b_0}^u + \phi_B \mathrm{P}_B^u + \phi_m^{\mathbb{I}_s} \mathrm{P}_{M_{\mathbb{I}_s}}^u + \phi_C^{\mathbb{I}_s} \mathrm{P}_{C_{\mathbb{I}_s}}^u \right) \mathbb{I}_s + \\
&\qquad \left(\phi_m^{\mathbb{I}_m} \mathrm{P}_{M_{\mathbb{I}_M}}^u + \phi_C^{\mathbb{I}_m} \mathrm{P}_{C_{\mathbb{I}_m}}^u \right) \mathbb{I}_m \Bigg] \\
&= \sum_{k=1}^{F} \rho_{f_k}^{u_0} \Bigg\{ \Lambda^{\mathrm{stor}} \eta_{f_k}^{u_0} + \phi_d \left(1 - \eta_{f_k}^{u_0}\right) \left[1 - \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \right] + \Bigg(\phi_{b_0} \left(1 - \eta_{f_k}^{u_0}\right) \\
&\quad \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \eta_{f_k}^{b_0} + \phi_B \left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \left(1 - \eta_{f_k}^{b_0}\right) \\
&\quad \left(1 - \prod_{b_j \in \Phi_b \setminus b_0} \left(1 - \eta_{f_k}^{b_j}\right) \right) + \phi_m^{\mathbb{I}_s} \left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \left(1 - \eta_{f_k}^{b_0}\right) \\
&\quad \prod_{b_j \in \Phi_b \setminus b_o} \left(1 - \eta_{f_k}^{b_j}\right) \eta_{f_k}^{m_0} + \phi_C^{\mathbb{I}_s} \Bigg[\left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \left(1 - \eta_{f_k}^{b_0}\right) \\
&\quad \prod_{b_j \in \Phi_b \setminus b_0} \left(1 - \eta_{f_k}^{b_j}\right) \Bigg] \left(1 - \eta_{f_k}^{m_0}\right) \Bigg) \mathbb{I}_s + \Bigg(\phi_m^{\mathbb{I}_m} \left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_i} \left(1 - \eta_{f_k}^{u_i}\right) \eta_{f_k}^{m_0} + \\
&\quad \phi_C^{\mathbb{I}_m} \left(1 - \eta_{f_k}^{u_0}\right) \prod_{u_i \in \Phi_u \setminus u_0} \left(1 - \eta_{f_k}^{u_i}\right) \left(1 - \eta_{f_k}^{m_0}\right) \Bigg) \mathbb{I}_m \Bigg\}
\end{aligned} \tag{4.11}$$

4.3 Cost Minimization using Artificial Intelligence

The goal is to minimize the cost function. However, the optimization problem $\mathbf{P_1}$ is highly challenging to solve. By its nature, in general, $\mathbf{P_1}$ in equation (4.10) is non-convex [39]. Furthermore, due to the nonlinear and combinatorial content placement problem, this decision-making problem is, in general, NP complete. The option of exhaustive search is also out of consideration since it has exponential complexity. Instead, an artificial intelligence

based algorithm is applied to obtain a sub-optimal, yet, efficient solution. Recall that the minimization problem requires to find the optimal solutions for the caching placement probabilities at all individual nodes for all contents at the catalog. In the following, a modified particle swarm optimization (PSO) algorithm is used to obtain the best set of parameters. Before diving into the proposed modified algorithm for solving problem $\mathbf{P}_1$, a concise description of the PSO algorithm is presented in the following.

PSO was introduced inspired by the collective behavior of bees or birds. It is a swarm intelligence approach which is guaranteed to converge [50]. Here, all possible candidate solutions sets are named as the particles. Each particle, denoted by i, has a position, denoted by x_i. Furthermore, there is a personal best position for a particle, usually indicated by p_i^{best}, while there also exists a global best position for the entire swarm, typically denoted by g^{best}. The PSO algorithm is then formulated with an intention that each particle updates its position and reach to an optimal point with an exploration and exploitation manner. To do that, a velocity term, usually denoted by v_i^t, where i and t represents the particle index number and current iteration number, respectively; is added to the particle's previous position to get an updated position. The following two simple equations, thus, govern the PSO algorithm.

$$v_i^{t+1} = a v_i^t + \psi_1 \epsilon_1 \left(p_i^{best} - x_i \right) + \psi_2 \epsilon_2 \left(g^{best} - x_i \right), \tag{4.12}$$

$$x_i^{t+1} = x_i^t + v_i^t, \tag{4.13}$$

where a, ψ_1 and ψ_2 are the parameters that need to be selected properly. ϵ_1 and ϵ_2 are two Unifrom random variables. Note that ψ_1 and ψ_2 are positive acceleration coefficients, which are also known as the cognitive and social learning factors, respectively [39].

While this is a general framework for the PSO algorithm, direct use of this algorithm may not suffice the constraint of the optimization problem [51,52]. The problem arises when a new velocity is added to the previous position of a particle. Remember that the constraints of $\mathbf{P_1}$ needs to be satisfied while obtaining a possible solution set. Therefore, each particle must have a position matrix which must not violate the restriction. Note that the constraints

are - each caching placement probabilities must be in $[0, 1]$ and summation of these terms cannot exceed the total cache storage capacity of the respective node. Therefore, in the following, the PSO algorithm is modified to solve the optimization problem $\mathbf{P}_1$ efficiently satisfying all the constraints.

Let there be P numbers of particles. Let $\boldsymbol{\eta}_f^{u_i}$ denote the caching probabilities of user u_i for all contents f_k in the content catalog. Therefore, this has a size of $F \times 1$. Similarly, for all sBS and MBS, let $\boldsymbol{\eta}_f^{b_j}$ and $\boldsymbol{\eta}_f^{m_l}$ denote their caching placement probabilities for all contents, respectively. Then all of these parameters can be stacked into a matrix of dimension, $(|\mathcal{U}| + |\mathcal{B}| + |\mathcal{M}|) \times |\mathcal{F}|$ which is the exact shape of each particle. Let the current position of each of these particles be denoted by $\mathbf{X}_i^t$. Notice that in this case, each particle's position $\mathbf{X}_i^t$ has a shape of $(|\mathcal{U}| + |\mathcal{B}| + |\mathcal{M}|) \times |\mathcal{F}|$. Similar to the original PSO algorithm, the goal is to update each of these positions by adding a velocity term to the particle's previous position. Let $\mathbf{V}_i^t \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{B}|+|\mathcal{M}|)\times|\mathcal{F}|}$ denote the velocity. Furthermore, let $\mathbf{P}_i^{\text{best}}$ denote the personal best position of particle i, while the global best for the entire swarm is denoted by $\mathbf{G}^{\text{best}}$. Hence, each particle updates its velocity with social and individual cognition. The following equation is used to govern these updates.

$$\mathbf{V}_i^{t+1} = a\mathbf{V}_i^t + \psi_1 \left[\boldsymbol{\mathcal{E}}_1 \odot \left(\mathbf{P}_i^{\text{best}} - \mathbf{X}_i^t\right)\right] + \psi_2 \left[\boldsymbol{\mathcal{E}}_2 \odot \left(\mathbf{G}^{\text{best}} - \mathbf{X}_i^t\right)\right], \tag{4.14}$$

where a, ψ_1 and ψ_2 are the parameters as described in equation (4.12). $\boldsymbol{\mathcal{E}}_1$ and $\boldsymbol{\mathcal{E}}_2$ are two matrices of shapes $\mathbb{R}^{(|\mathcal{U}|+|\mathcal{B}|+|\mathcal{M}|)\times|\mathcal{F}|}$. Each elements of these two matrices are drawn from Uniform random distribution. Moreover, $\odot$ represents Hadamard product.

The position of each particle is then get updated by the velocity similar to equation (4.12). However, as the constraint of equations (4.10b-4.10d) and (4.10e) need to be satisfied, necessary modifications are also required for this equation accordingly. Let $\mathbf{X}_{i_{int}}^{t+1}$ denote an intermediate updated position of particle i as shown in the following equation.

$$\mathbf{X}_{i_{int}}^{t+1} = \mathbf{X}_i^t + \mathbf{V}_i^t. \tag{4.15}$$

The reason for considering this intermediate position is to keep each of the elements of the particle positions in the feasible search space. Then necessary normalization and scaling are performed to keep the obtained result in the feasible space. Note that from this intermediate particle position, a normalized particle position is obtained which is then used as the current particle position $\mathbf{X}_i^t$. Moreover, the ultimate goal for each particle is to reach to an optimal position $\mathbf{X}_i^*$ which is the global best $\mathbf{G}^{\text{best}}$ and results in the minimum cost. The detailed process of the proposed modified algorithm is presented in Alg. 6.

Some observations on the algorithm: The algorithm is modeled in a way that is capable of dealing with a normalized particle - position and velocity. The constraints suggest that the particle position needs to be restricted in a probability range while the summation cannot exceed the cache storage capacity of the respective node. Therefore, the initial values are bounded in between $[0,\ 1/\mathcal{C}^j]$ so that when necessary scaling is performed the obtained value does not violate the probability range. Therefore, the particle position in step 4 and velocity in step 6 are initialized following this notion. Furthermore, as the caching probabilities of the nodes in dimension j is limited to $\mathcal{C}^j$, in steps 24 and 26, a random number of contents, randint($\mathcal{C}^{\text{j}}$), are chosen to be stored with higher probability values.

4.4 Results and Discussion

For the simulation, user are considered to be distributed in a 2D plane following a HPPP of intensity, $\lambda_u \in [2.5^{-3}, 2.5^{-4}]$ (per m^2). The low powered sBS are drawn following another HPPP of intensity, $\lambda_b = 2.5^{-5}$, (per m^2). For the MBS, $\lambda_m = 1.5^{-6}$ (per m^2) is considered. The coverage radii of the user, sBS and MBS are taken as $R_u = 15\ m$, $R_b = 150\ m$, $R_m = 500\ m$, respectively. $\alpha = 0.5$ and the skewness, γ of the Zipf distribution is considered to be selected uniformly in between $\{0.1, 2.5\}$. For the PSO algorithm $a = 0.9$ and $\psi_1 = \psi_2 = 0.4$ has been considered. The costs of content storing Λ^{sror} is chosen to be 2000. The communication costs are chosen as $\Lambda_d^{com} = 100$, $\Lambda_{b_0}^{com} = 500$, $\Lambda_{b_{\mathbb{I}_s}}^{com} = \Lambda_{b_0}^{com} + 600$, $\Lambda_{m_{0_{\mathbb{I}_s}}}^{com} = \Lambda_{b_0}^{com} + 800$, $\Lambda_{C_{\mathbb{I}_s}}^{com} = \Lambda_{b_0}^{com} + 5000$, $\Lambda_{m_{0_{\mathbb{I}_m}}}^{com} = 2500$ and $\Lambda_{C_{\mathbb{I}_m}}^{com} = \Lambda_{m_{0_{\mathbb{I}_m}}}^{com} + 5000$ for D2D, tagged sBS, soft sBS, MBS (when $\mathbb{I}_s = 1$), cloud (when $\mathbb{I}_s = 1$), serving MBS MBS

Algorithm 6 Collaborative Edge Caching Algorithm using Modified PSO

for each particle, $i = 1, 2, \ldots, P$ **do**
  $\mathbf{X}_i = [\,]$, $\mathbf{V}_i = [\,]$
  for each dimension $j = 1, 2, \ldots, D$ **do** ▷ $D = |\mathcal{U}| + |\mathcal{B}| + |\mathcal{M}|$
    initialize the particles positions, $\mathbf{x}_{ji}$ with uniform random vector of size $\mathbb{R}^{|\mathcal{F}|}$ by making sure $\sum_{k=1}^{F} x_{ji}[k] = 1$ and $0 \leq x_{ji}[k] \leq \frac{1}{\mathcal{C}^j}$, $\forall\ k \in \mathcal{F}$. ▷ $\mathcal{C}^j$ is the cache storage of the node in j^{th} dimension
    set $\mathbf{X}_i[j, :] \leftarrow \mathbf{x}_{ji}$
    initialize particles velocity, $\mathbf{v}_{ji}$ with uniform random vector of size $\mathbb{R}^{|\mathcal{F}|}$ by making sure $\sum_{k=1}^{F} v_{ji}[k] = 1$ and $0 \leq v_{ji}[k] \leq \frac{1}{\mathcal{C}^j}$, $\forall\ k \in \mathcal{F}$.
    set $\mathbf{V}_i[j, :] \leftarrow \mathbf{v}_{ji}$
  end for
  set particle best position, $\mathbf{P}_i^{best}$ as the initial position
  if $\Xi_\pi\left(\mathbf{P}_i^{\text{best}}\right) < \Xi_\pi\left(\mathbf{G}^{\text{best}}\right)$ **then**
    $\mathbf{G}^{\text{best}} \leftarrow \mathbf{P}_i^{\text{best}}$
  end if
end for
while termination criteria has not met **do**
  for each particle, i **do**
    for each dimension, $j = 1, 2, \ldots, D$ **do**
      draw uniform random vectors, $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_1$ of size $\mathbb{R}^{|\mathcal{F}|}$
      set $\mathbf{v}_{ji} \leftarrow a\mathbf{v}_{ji} + \psi_1\left[\boldsymbol{\epsilon}_1 \odot \left(\mathbf{p}_{ji}^{\text{best}} - \mathbf{x}_{ji}\right)\right] + \psi_2\left[\boldsymbol{\epsilon}_2 \odot \left(\mathbf{g}_j^{\text{best}} - \mathbf{x}_{ji}\right)\right]$
      set $\mathbf{V}_i[j, :] \leftarrow \mathbf{v}_{ji}$
    end for
    update particles intermediate position, $\mathbf{X}_{i_{int}}$ using equation (4.15)
    $\mathbf{X}_i^{\text{scl}} = [\,]$, $\mathbf{P}_i^{\text{scl\_best}} = [\,]$, $\mathbf{G}^{\text{scl\_best}} = [\,]$
    for each dimension $j = 1, 2, \ldots, D$ **do**
      random_hike $\leftarrow$ randint$(\mathcal{C}^j)$
      for i in len(random_hike) **do**
        $\mathbf{X}_{i_{int}}[j, \text{randint}(F)] \leftarrow \frac{\sum_{k=1}^{F} \mathbf{X}_{i_{int}}[j,:]}{\mathcal{C}^j}$
      end for
      $\mathbf{X}_i[j, :] \leftarrow \frac{\mathbf{X}_{i_{int}}[j,:]}{\sum_{k=1}^{F} \mathbf{X}_{i_{int}}[j,:]}$ ▷ Normalized particle position
      $\mathbf{X}_i^{\text{scl}}[j, :] \leftarrow \mathcal{C}^j \mathbf{X}_i[j, :]$
      $\mathbf{P}_i^{\text{scl\_best}}[j, :] \leftarrow \mathcal{C}^j \mathbf{P}_i^{\text{best}}[j, :]$
      $\mathbf{G}_i^{\text{scl\_best}}[j, :] \leftarrow \mathcal{C}^j \mathbf{G}_i^{\text{best}}[j, :]$
    end for
    if $\Xi_\pi\left(\mathbf{X}_i^{\text{scl}}\right) < \Xi_\pi\left(\mathbf{P}_i^{\text{scl\_best}}\right)$ **then**
      $\mathbf{P}_i^{\text{best}} \leftarrow \mathbf{X}_i$
      do necessary scaling following step 30
      if $\Xi_\pi\left(\mathbf{P}_i^{\text{scl\_best}}\right) < \Xi_\pi\left(\mathbf{G}^{\text{scl\_best}}\right)$ **then**
        $\mathbf{G}^{\text{best}} \leftarrow \mathbf{P}_i^{\text{best}}$
      end if
    end if
  end for
end while
return $\mathbf{G}^{\text{best}}$ and do necessary scaling following step 31 and **return** $\mathbf{G}^{\text{scl\_best}}$

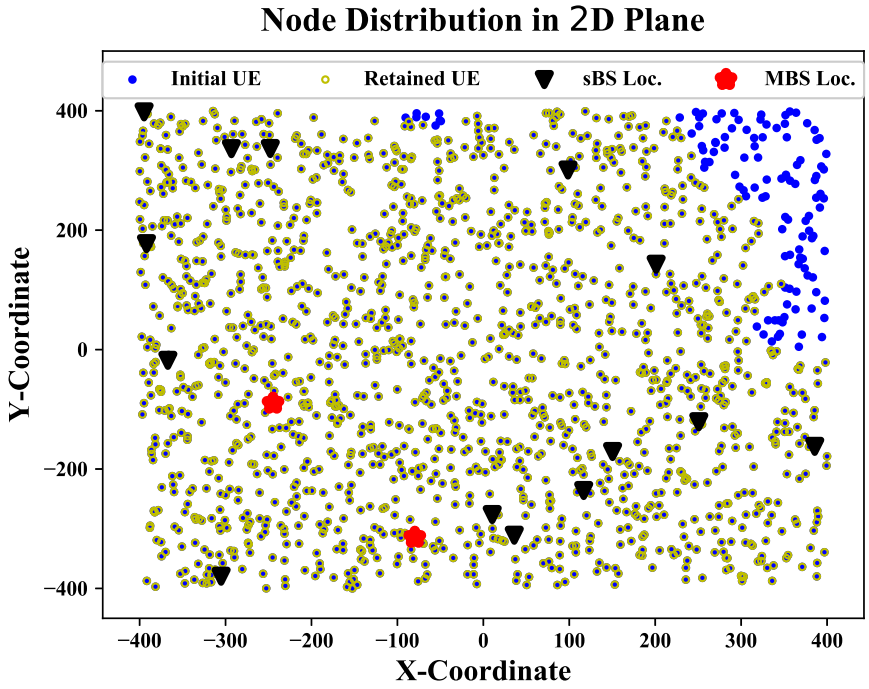


Fig. 4.2: Retained nodes in $2D$ plane with $\lambda_u = 2.5^{-3}$ (per m^2), $\lambda_b = 2.5^{-5}$ (per m^2) and $\lambda_m = 1.56^{-7}$ (per m^2)

(when $\mathbb{I}_m = 1$) and cloud (when $\mathbb{I}_m = 1$), respectively. Thus, the total costs of accessing content are $\phi_d = 2200$, $\phi_{b_0} = 2500$, $\phi_B = 3100$, $\phi_m^{\mathbb{I}_s} = 3300$, $\phi_C^{\mathbb{I}_s} = 7500$, $\phi_m^{\mathbb{I}_m} = 4500$ and $\phi_C^{\mathbb{I}_m} = 9500$.

Firstly, the nodes are drawn following their respective intensities as mentioned above. Then the users that are not in the coverage region of any of the MBS or sBS are discarded for the convenience. Such a realization is presented in Figs. 4.2 and 4.3. From this realization, α portion of the users is randomly chosen to act as the receiver. The selected users then can make content requests while the other $(1 - \alpha)$ portion serves as transmitters. From a random realization, this set up is shown in Fig. 4.3. Note that the goal of adopting spatial node distribution, in this chapter, is to see how caching performs in a real heterogeneous network. Thus, next, the nodes are associated with their respective sBS and MBS. As the link level optimization is out of the scope of this thesis, users are associated with the closest base stations. After these initial steps, next, the obtained results from the modified PSO

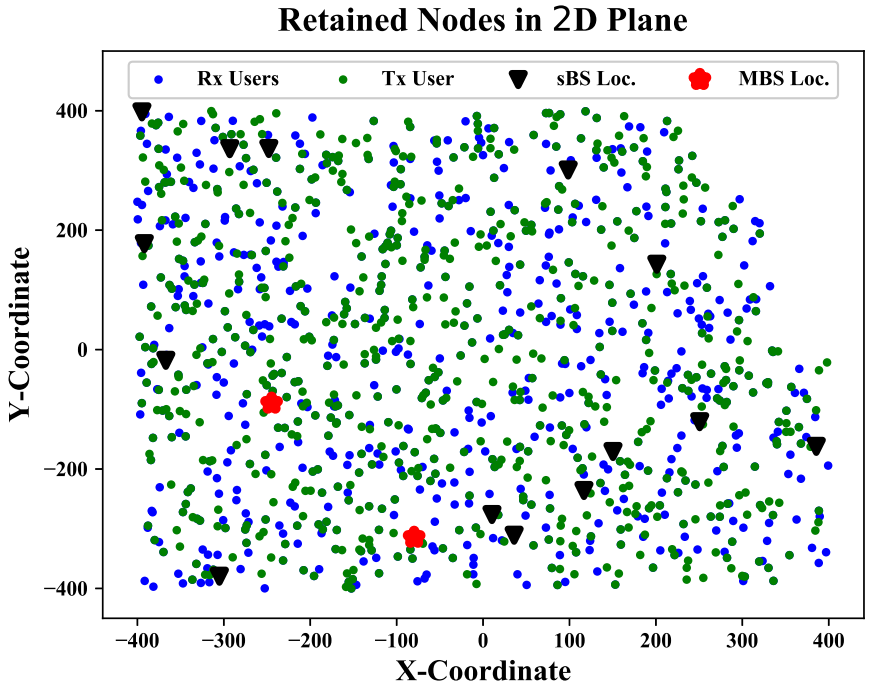


Fig. 4.3: Transmitter and receiver nodes in $2D$ plane with $\lambda_u = 2.5^{-3}$ (per m^2), $\lambda_b = 2.5^{-5}$ (per m^2) and $\lambda_m = 1.56^{-7}$ (per m^2)

algorithm is observed in the following subsection.

4.4.1 Caching Placement

In the following, the modified PSO algorithm is used to attain an optimal cache placement solution, and then its performance is studied. To show the effectiveness of the proposed algorithm, first, the result of the convergence of this algorithm is presented. The results and observations of this suggest that similar to the general PSO algorithm, the proposed modified algorithm also converges after a minimal number of iterations. However, it is worth mentioning here that since PSO is a meta-heuristic algorithm, it may get stuck to a local minimum until the particles find a $\mathbf{P}^{\text{best}}$ position that is better than the previous global best position. A convergence plot of the proposed algorithm is shown in Fig. 4.4 which also validates this point. The obtained global best $\mathbf{G}^{\text{scl\_best}}$ using this algorithm is now scrutinized in the following to confirm that it does not break any constraints.

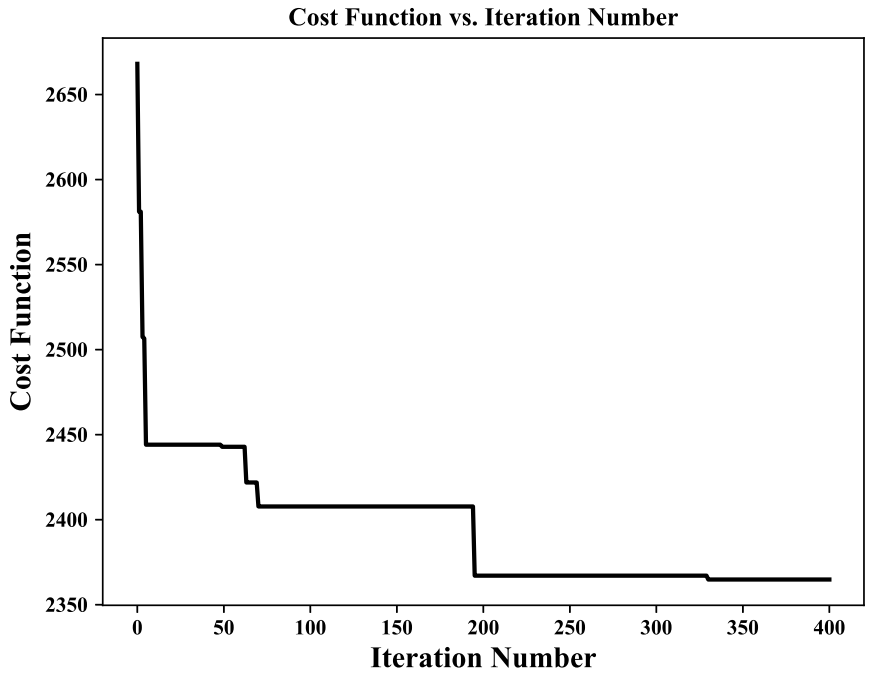


Fig. 4.4: Convergence of the proposed modified PSO algorithm

At first, the obtained results are validated for the user nodes. Recall that a heterogeneous caching placement case is being considered in this chapter. Therefore, it is expected that the algorithm should satisfy the constraints for each of the user nodes while ensuring different contents are stored. The caching placement probabilities for user 1, 10 and 20 are presented in Fig. 4.5 for judging the obtained result when $\mathcal{C}_d = 2$. From this figure, it is quite evident that different users are storing different contents. The cache storing probabilities are also bounded in between 0 and 1. Furthermore, the summations of the caching placement probabilities for each user results in 2.

Similarly, the caching placement probabilities for sBS 1, 2 and 3 are shown in Fig. 4.6 when $\mathcal{C}_b = 6$, while the caching probabilities at the MBS are shown in Fig. 4.7 when $\mathcal{C}_m = 6$. These figures also validate that the obtained results for the sBS and MBS are also not violating any constraints. The above discussion and presented results suggest that the proposed modified PSO algorithm is performing quite well. Therefore, the solution set obtained using this algorithm is next used to evaluate the performance in the following

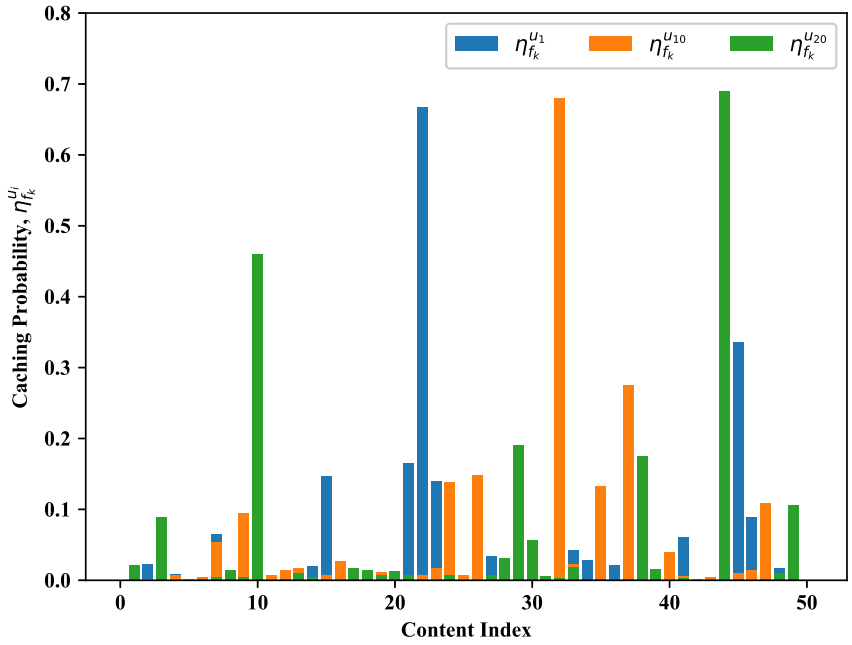


Fig. 4.5: Obtained caching probabilities at the user nodes

subsection.

4.4.2 Performance Analysis

The performance of the proposed PSO algorithm is studied for different cache sizes. Note that if the cache size increases the cost should decrease because more contents can then be placed into the cache storage. Thus, when the cache sizes of the cache-enabled nodes increase the cost of content sharing should decrease. Alg. 6 should also validate this general notion. In the following, this will be tested. Moreover, the performance of the algorithm is also compared to that of (1) random caching scheme, (2) equal caching scheme and (3) K-popular caching scheme. Note that in the random caching scheme, contents are stored randomly while satisfying the constraints. In the equal caching scheme, contents are placed with equal probability, while in K-popular scheme, only the first K popular contents are stored. For the simulation, K is chosen to be 15.

First, the performance of the PSO algorithm is presented for different user cache sizes

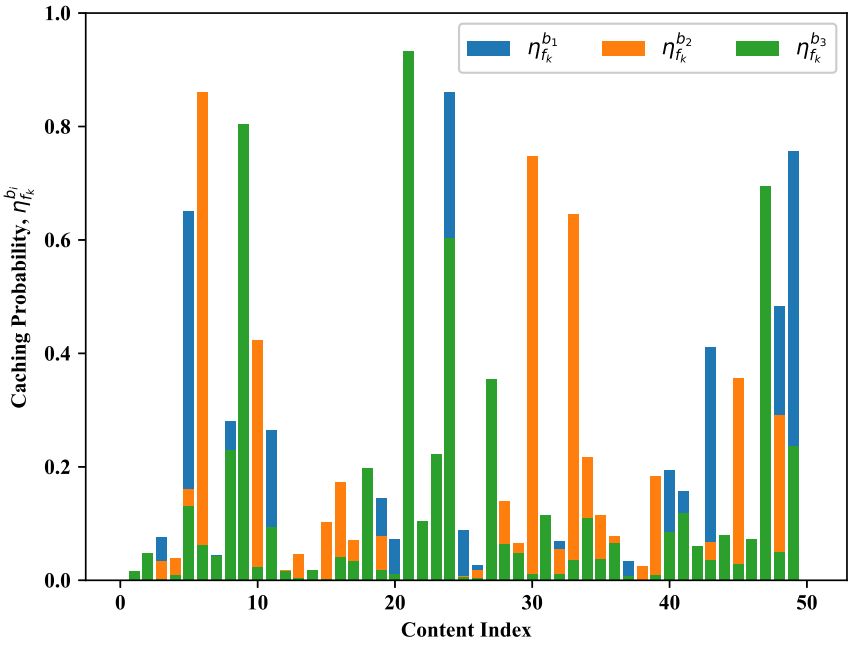


Fig. 4.6: Obtained caching probabilities at the small base stations

of the users. The cache sizes of the D2D users are increased from 1 to 5 while the sBS and MBS cache sizes have remained fixed. As expected, the obtained results from all the caching schemes, except random caching, show that as cache size increases, the cost is decreasing as more contents are being stored in the proximity. Note that the random caching scheme does not have any control over this. In each time, some contents are being stored randomly at the cache enabled nodes. However, this does not necessarily increase system performances. The performance only increases when the most likely to be requested contents are being stored at the caching nodes. Also, in K-popular first case, only a few popular contents are being stored discarding storing any other contents that might be requested by other users in proximity. Therefore, it is expected that the performance of the proposed algorithms outperforms all of these baseline caching schemes.

The obtained results also reflect this. A decreasing trend is observed while the cache sizes of the users keep increasing. Fig. 4.8 shows the performance graph in this case. As claimed, the proposed modified algorithm always outperforms the baseline caching schemes.

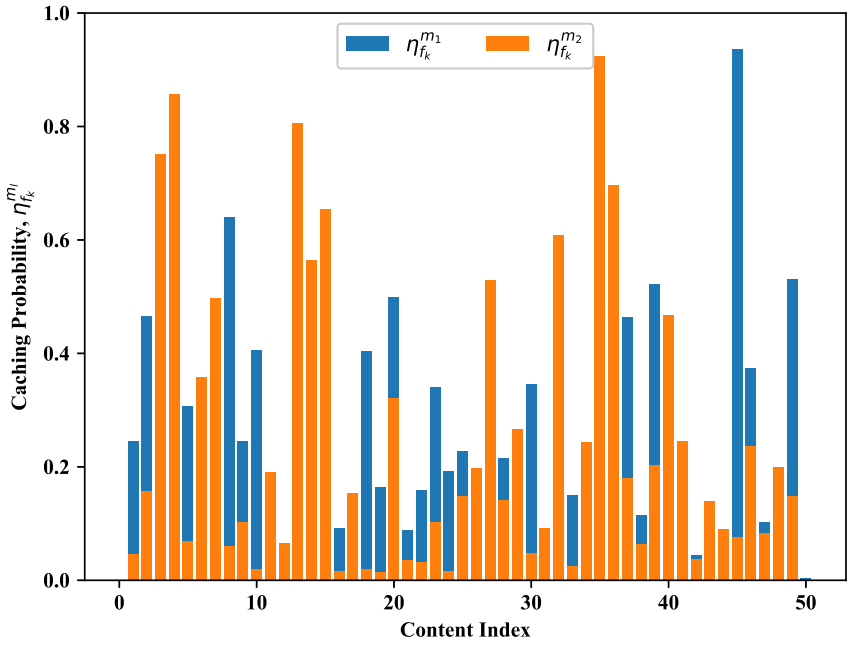


Fig. 4.7: Obtained caching probabilities at the macro base stations

However, the performance slowly increases while the cache sizes of the sBS or MBS keep growing. The reason for this is getting the content from the base stations are more costly than that of the neighbor D2D nodes. The impact of the sBS and MBSs cache sizes are shown in Figs. 4.9 and 4.10.

This has to be mentioned that as PSO is a meta-heuristic algorithm, a globally optimal solution is not guaranteed while the algorithm does converge. Often it may get stuck into local minima. Besides, the iteration number of the while loop can have a significant impact. Therefore, while the proposed modified PSO gives a good trusted solution, similar to that of the general PSO algorithm, a globally minimum solution may not be attainable. Depending on the computation resources, however, one can run the algorithm for a larger number of iterations and be able to obtain better results.

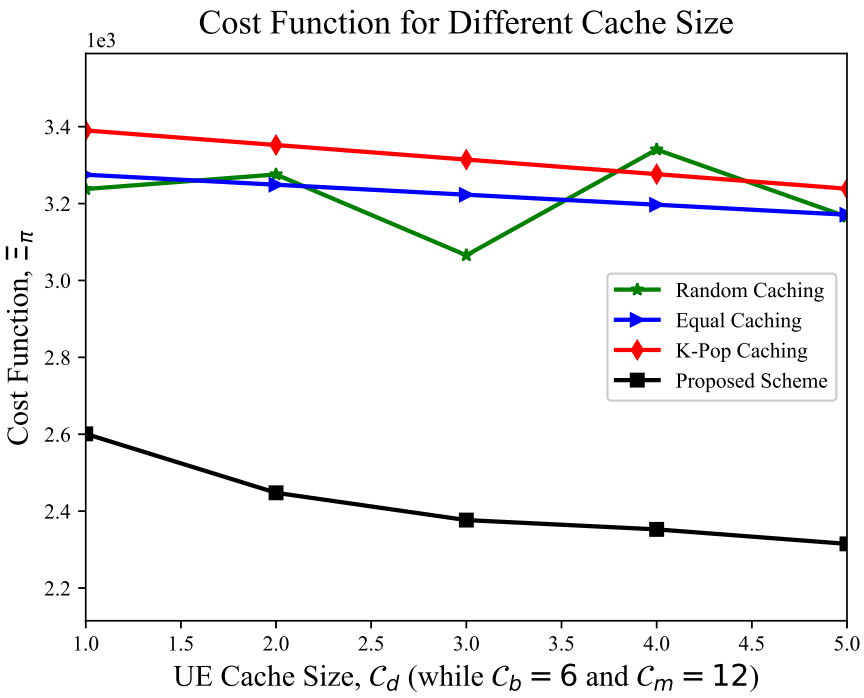


Fig. 4.8: Impact of the user cache storage size

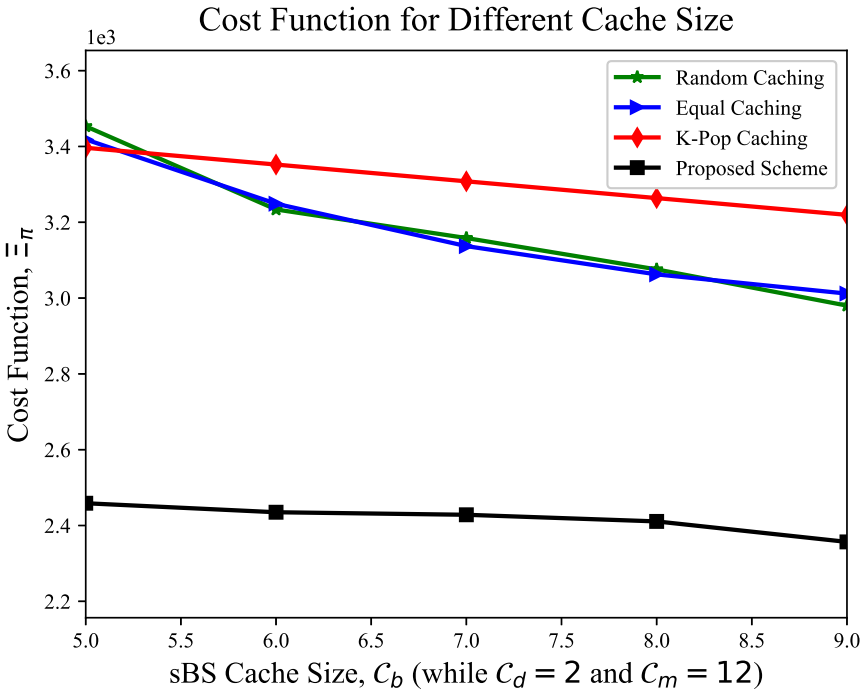


Fig. 4.9: Impact of the sBS cache storage size

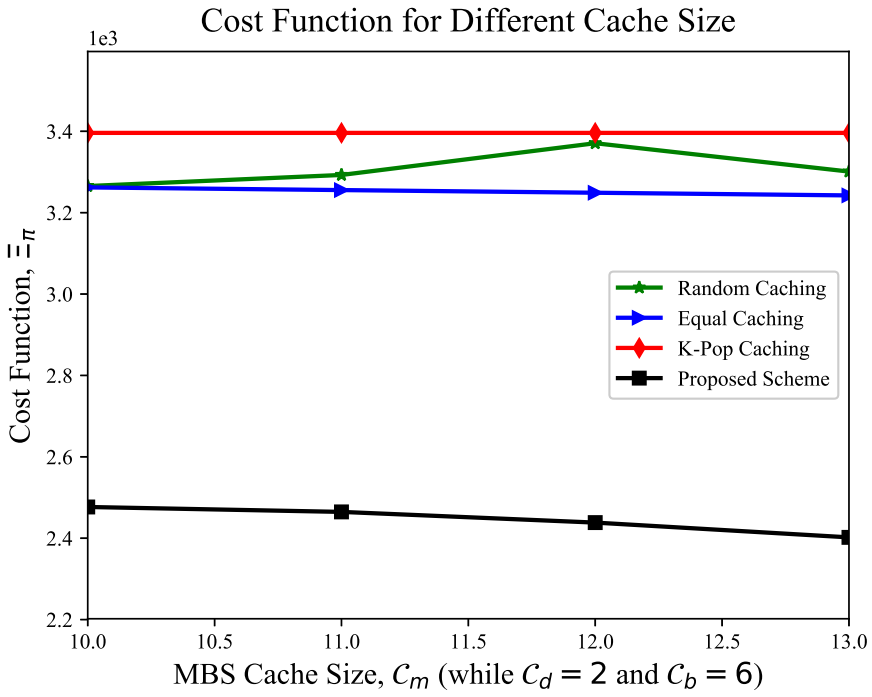


Fig. 4.10: Impact of the MBS cache storage size

CHAPTER 5

Conclusion and Future Works

5.1 Conclusion

In a CDN, obtaining accurate content popularity prediction is a hard task. Therefore, a well-studied prediction model is required for predicting individual user's preferences as well as the user's activity level. Following the presented LSTM model, both the temporal dynamics of the user preferences and activity levels have been successfully captured. With presented analysis and simulation, it can be concluded that the performance of a CDN heavily depends on the prediction part. Furthermore, as far as content placement is a concern, heterogeneous caching placement outperforms the homogeneous placement strategy. Hence, from a system administrator perspective, heterogeneous content placement strategy is beneficial with a fair trade-off with the complexity of measuring each users preferences. Finally, after making the fair comparisons among different cache placement strategies, it can be concluded that the proposed greedy overlapped caching mechanism outperforms the others content placement algorithms when a fair amount of cache storage capacity is available at the edge nodes. Furthermore, in a real-world HetNet deployment, when the numbers of nodes are considered to be random, AI-based algorithms can be implemented to solve the hard combinatorial heterogeneous caching placement decision problems efficiently and timely.

5.2 Future Works

While this thesis presents some key ideas for content caching at the edges, there are several future scopes of this work. Some of these future directions can be listed as follows:

1. If spatial node distribution is considered, a major challenge is to capture the dynamic behaviors of the user preference and activity levels. While this thesis, in Chapter

4, considered a real-world HetNet deployment, this thesis limits to model the short temporal dynamics in this case. A more evolved ML-based approach is required since, in each realization, a random number of nodes are obtained in the 2D plane. It would be interesting to come up with an approach to predict and to forecast user preferences in that case.

2. Genre-based prediction modeling can also be adopted and tested following the outline of the proposed LSTM based prediction model.

3. User-generated content (UGC) can also be considered to be predicted in the prediction model.

4. Link level optimization can be performed. However, in that case, the mathematical tractability will be much challenging.

5. Several performance metrics, such as cache hit ratio (CHR), throughput, energy efficiency, area spectral density, etc. can also be exploited.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] A. Fehske, G. Fettweis, J. Malmodin, and G. Biczok, "The global footprint of mobile communications: The ecological and economic perspective," *IEEE Commun. Mag.*, vol. 49, no. 8, pp. 55–62, August 2011.

[3] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wireless Personal Commun.*, vol. 58, no. 1, pp. 49–69, 2011.

[4] R. Wang, X. Peng, J. Zhang, and K. B. Letaief, "Mobility-aware caching for content-centric wireless networks: modeling and methodology," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 77–83, August 2016.

[5] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1791–1805, Aug 2018.

[6] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless powered mobile-edge computing systems," *arXiv preprint arXiv:1806.04589*, 2018.

[7] T. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, pp. 1–1, 2018.

[8] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, pp. 1–1, 2018.

[9] 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 Commun. Mag.*, vol. 52, no. 2, pp. 131–139, 2014.

[10] 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 Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3665–3676, July 2014.

[11] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femto-caching: Wireless video content delivery through distributed caching helpers," in *2012 Proc. IEEE INFOCOM*, March 2012, pp. 1107–1115.

[12] J. Liu, N. Kato, J. Ma, and N. Kadowaki, "Device-to-device communication in lte-advanced networks: A survey," *IEEE Commun. Surveys Tutorials*, vol. 17, no. 4, pp. 1923–1940, Fourthquarter 2015.

[13] 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*. Springer, 2018, pp. 169–177.

[14] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, 2012.

[15] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 44–53, December 2012.

[16] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Commun. Surv. Tutor.*, vol. 16, no. 2, pp. 1024–1049, Second 2014.

[17] C. Zhan and Z. Wen, "Content cache placement for scalable video in heterogeneous wireless network," *IEEE Commun. Lett.*, vol. 21, no. 12, pp. 2714–2717, Dec 2017.

[18] Z. Li and G. Simon, "In a telco-cdn, pushing content makes sense," *IEEE Trans. Netw. Service Manage.*, vol. 10, no. 3, pp. 300–311, 2013.

[19] M. A. Salahuddin, J. Sahoo, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on content placement algorithms for cloud-based content delivery networks," *IEEE Access*, vol. 6, pp. 91–114, 2018.

[20] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Commun. Surveys Tutorials*, vol. 17, no. 3, pp. 1473–1499, thirdquarter 2015.

[21] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. on Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec 2013.

[22] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *2014 IEEE Int. Conf. on Commun. (ICC)*, June 2014, pp. 1897–1903.

[23] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *2014 11th Int. Symp. on Wireless Commun. Syst. (ISWCS)*, Aug 2014, pp. 917–921.

[24] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang, "Learning-based content caching and sharing for wireless networks," *IEEE Trans. on Commun.*, vol. 65, no. 10, pp. 4309–4324, Oct 2017.

[25] B. N. Bharath, K. G. Nagananda, D. Gündüz, and H. V. Poor, "Caching with time-varying popularity profiles: A learning-theoretic perspective," *IEEE Trans. on Commun.*, vol. 66, no. 9, pp. 3837–3847, Sept 2018.

[26] B. N. Bharath, K. G. Nagananda, D. Guenduez, and H. V. Poor, "Learning-based content caching with time-varying popularity profiles," in *GLOBECOM 2017 - 2017 IEEE Global Commun. Conf.*, Dec 2017, pp. 1–6.

[27] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. on Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[28] M. M. Amiri and D. Gündüz, "Fundamental limits of coded caching: Improved delivery rate-cache capacity tradeoff," *IEEE Trans. on Commun.*, vol. 65, no. 2, pp. 806–815, Feb 2017.

[29] Z. Chen, P. Fan, and K. B. Letaief, "Fundamental limits of caching: improved bounds for users with small buffers," *IET Communications*, vol. 10, no. 17, pp. 2315–2318, 2016.

[30] A. Younis, T. X. Tran, and D. Pompili, "Bandwidth and energy-aware resource allocation for cloud radio access networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6487–6500, Oct 2018.

[31] B. Chen and C. Yang, "Caching policy for cache-enabled d2d communications by learning user preference," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6586–6601, Dec 2018.

[32] M.-C. Lee and A. F. Molisch, "On the caching policy and cooperation distance design in base station assisted wireless d2d networks," in *proc. of ICC*, 2018, pp. 1–7.

[33] M. Lee and A. F. Molisch, "Caching policy and cooperation distance design for base station-assisted wireless d2d caching networks: Throughput and energy efficiency optimization and tradeoff," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7500–7514, Nov 2018.

[34] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Trans. on Netw.*, vol. 16, no. 6, pp. 1447–1460, Dec 2008.

[35] R. Amer, M. M. Butt, M. Bennis, and N. Marchetti, "Delay analysis for wireless d2d caching with inter-cluster cooperation," in *proc. GLOBECOM*, 2017, pp. 1–7.

[36] R. Amer, M. M. Butt, M. Bennis, and N. Marchetti, "Inter-cluster cooperation for wireless d2d caching networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 6108–6121, Sep. 2018.

[37] Y. Sun, Z. Chen, and H. Liu, "Delay analysis and optimization in cache-enabled multi-cell cooperative networks," in *proc. GLOBECOM*, 2016, pp. 1–7.

[38] L. T. Tan, R. Q. Hu, and Y. Qian, "D2d communications in heterogeneous networks with full-duplex relays and edge caching," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4557–4567, Oct 2018.

[39] L. T. Tan, R. Q. Hu, and L. Hanzo, "Heterogeneous networks relying on full-duplex relays and mobility-aware probabilistic caching," *IEEE Trans. Commun.*, pp. 1–1, 2019.

[40] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb 2017.

[41] S. Li, J. Xu, M. van der Schaar, and W. Li, "Popularity-driven content caching," in *In Proc. IEEE INFOCOM 2016*, 2016.

[42] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proc. 7th ACM SIGCOMM*, 2007.

[43] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[44] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[45] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Commun. Surveys & Tuts.*, vol. 17, no. 3, pp. 1473–1499, 2015.

[46] K. Xu, X. Li, S. K. Bose, and G. Shen, "Joint replica server placement, content caching, and request load assignment in content delivery networks," *IEEE Access*, vol. 6, pp. 17 968–17 981, 2018.

[47] W. Bao, D. Yuan, K. Shi, W. Ju, and A. Y. Zomaya, "Ins and outs: Optimal caching and re-caching policies in mobile networks," in *Proc. of the Eighteenth ACM Int. Symposium on Mobile Ad Hoc Netw. and Comput.* ACM, 2018, pp. 41–50.

[48] J. G. Andrews, F. Baccelli, and R. K. Ganti, "A tractable approach to coverage and rate in cellular networks," *IEEE Trans. on Commun.*, vol. 59, no. 11, pp. 3122–3134, November 2011.

[49] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan, and R. Wichman, "In-band full-duplex wireless: Challenges and opportunities," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 9, pp. 1637–1652, Sep. 2014.

[50] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*, pp. 760–766, 2010.

[51] X. Hu and R. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proc. 6th World Multi-conf. Syst., Cybern. Informat.*, vol. 5, 2002, pp. 203–206.

[52] X. Hu, R. C. Eberhart, and Y. Shi, "Engineering optimization with particle swarm," in *Proc. IEEE Swarm Intell. Symp.* IEEE, 2003, pp. 53–57.

[53] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[54] T. K. Moon and W. C. Stirling, *Mathematical methods and algorithms for signal processing*. Prentice hall New York, 2000, vol. 1.

APPENDICES

APPENDIX A

Detail Derivations and Proofs

A.1 Detail Derivation of the Cost Function

$$\Xi_{\pi}^{\text{het}} = \frac{1}{U}\sum_{j=1}^{B}\sum_{i=1}^{U_{b_j}}\sum_{k=1}^{F}\rho_{f_k}^{u_i}\left\{\Lambda^{stor}\mathrm{P}_{o}^{u_i} + \phi_d\mathrm{P}_{d}^{u_i} + \phi_b\mathrm{P}_{b_j}^{u_i} + \phi_{BS}\mathrm{P}_{\mathrm{B}}^{u_i} + \phi_C\mathrm{P}_{c}^{u_i}\right\} \tag{A.1a}$$

$$= \frac{1}{U}\sum_{j=1}^{B}\sum_{i=1}^{U_{b_j}}\sum_{k=1}^{F}\rho_{f_k}^{u_i}\left\{\Lambda^{stor}a_{f_k}^{u_i} + \phi_d\left[\left(1-a_{f_k}^{u_i}\right)\left\{1-\prod_{i'\in\mathcal{U}_{b_j}\setminus i}\left(1-a_{f_k}^{u_{i'}}\right)\right\}\right] + \right.$$

$$\phi_b\left[\left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}_{b_j}\setminus i}\left(1-a_{f_k}^{u_{i'}}\right)\right]\eta_{f_k}^{b_j} + \phi_{BS}\left(\left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}_{b_j}\setminus i}\left(1-a_{f_k}^{u_{i'}}\right)\right.$$

$$\left.\left\{\left(1-\eta_{f_k}^{b_j}\right)\left[1-\prod_{j'\in\mathcal{B}\setminus j}\left(1-\eta_{f_k}^{b_{j'}}\right)\right]\right\}\right) + \phi_C\left[\left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}\setminus i}\left(1-a_{f_k}^{u_{i'}}\right)\right.$$

$$\left.\left.\left\{\left(1-\eta_{f_k}^{b_j}\right)\prod_{j'\in\mathcal{B}\setminus j}\left(1-\eta_{f_k}^{b_{j'}}\right)\right\}\right]\right\} \tag{A.1b}$$

$$= \frac{1}{U}\sum_{j=1}^{B}\sum_{i=1}^{U_{b_j}}\sum_{k=1}^{F}\rho_{f_k}^{u_i}\left\{\Lambda^{stor}a_{f_k}^{u_i} + \phi_d\left(1-a_{f_k}^{u_i}\right) - \phi_d\left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}_{b_j}\setminus i}\left(1-a_{f_k}^{u_{i'}}\right) + \right.$$

$$\left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}_{b_j}\setminus i}\left(1-a_{f_k}^{u_{i'}}\right)\left[\phi_b\eta_{f_k}^{b_j} + \phi_{BS}\left(1-\eta_{f_k}^{b_j}\right)\left\{1-\prod_{j'\in\mathcal{B}\setminus j}\left(1-\eta_{f_k}^{b_{j'}}\right)\right\} + \right.$$

$$\left.\left.\phi_C\left(1-\eta_{f_k}^{b_j}\right)\prod_{j'\in\mathcal{B}\setminus j}\left(1-\eta_{f_k}^{b_{j'}}\right)\right]\right\} \tag{A.1c}$$

$$= \frac{1}{U}\sum_{j=1}^{B}\sum_{i=1}^{U_{b_j}}\sum_{k=1}^{F}\rho_{f_k}^{u_i}\left\{\Lambda^{stor}a_{f_k}^{u_i} + \phi_d\left(1-a_{f_k}^{u_i}\right) - \left(1-a_{f_k}^{u_i}\right)\prod_{i'\in\mathcal{U}_{b_j}\setminus i}\left(1-a_{f_k}^{u_{i'}}\right)\left[\phi_d - \right.\right.$$

$$\left.\phi_b\eta_{f_k}^{b_j} - \phi_{BS}\left(1-\eta_{f_k}^{b_j}\right) + \phi_{BS}\left(1-\eta_{f_k}^{b_j}\right)\prod_{j'\in\mathcal{B}\setminus j}\left(1-\eta_{f_k}^{b_{j'}}\right) - \phi_C\left(1-\eta_{f_k}^{b_j}\right)\right.$$

$$\prod_{j' \in \mathcal{B} \setminus j} \left(1 - \eta_{f_k}^{b_{j'}}\right) \Bigg] \Bigg\} \tag{A.1d}$$

$$= \frac{1}{U} \sum_{j=1}^{B} \sum_{i=1}^{U_{b_j}} \sum_{k=1}^{F} \rho_{f_k}^{u_i} \Bigg\{ \Lambda^{stor} a_{f_k}^{u_i} + \phi_d \left(1 - a_{f_k}^{u_i}\right) - \left(1 - a_{f_k}^{u_i}\right) \prod_{i' \in \mathcal{U}_{b_j} \setminus i} \left(1 - a_{f_k}^{u_{i'}}\right) \Bigg[\phi_d -$$

$$\phi_b \eta_{f_k}^{b_j} - \phi_{BS} \left(1 - \eta_{f_k}^{b_j}\right) + (\phi_{BS} - \phi_C) \left(1 - \eta_{f_k}^{b_j}\right) \prod_{j' \in \mathcal{B} \setminus j} \left(1 - \eta_{f_k}^{b_{j'}}\right) \Bigg] \Bigg\} \tag{A.1e}$$

A.2 Proof of Monotonicity of $\eta_{f_k}^{b_j}$

To show the monotonicity of the Ξ_{π}^{het} with respect to the optimizing parameter $\eta_{f_k}^{b_j}$, let us take the first order partial derivatives with respective to all $\eta_{f_k}^{b_j}$s as follows:

$$\frac{\delta \Xi_{\pi}^{\text{het}}}{\delta \eta_{f_k}^{b_j}} = \tau_1 \Bigg[\phi_b - \phi_{BS} + (\phi_{BS} - \phi_C) \prod_{j' \in \mathcal{B} \setminus j} \left(1 - \eta_{f_k}^{b_{j'}}\right) \Bigg], \tag{A.2}$$

where τ_1 is a constant and non-negative term as shown in equation (A.3). As $\phi_C >> \phi_{BS} >> \phi_b$, $\frac{\delta \Xi_{\pi}^{\text{het}}}{\delta \eta_{f_k}^{b_j}}$ is thus ≤ 0.

$$\tau_1 = \frac{1}{U} \rho_{f_k}^{u_i} \left(1 - a_{f_k}^{u_i}\right) \prod_{i' \in \mathcal{U}_{b_j} \setminus i} \left(1 - a_{f_k}^{u_{i'}}\right). \tag{A.3}$$

$$\frac{\delta \Xi_{\pi}^{\text{het}}}{\delta \eta_{f_k}^{b_{j'}}} = \tau_1 \Bigg[(\phi_{BS} - \phi_C) \left(1 - \eta_{f_k}^{b_j}\right) \prod_{l \in \mathcal{B} \setminus j} \left(1 - \eta_{f_k}^{l}\right) \Bigg]. \tag{A.4}$$

By the same reasoning, it can be said that $\frac{\delta \Xi_{\pi}^{\text{het}}}{\delta \eta_{f_k}^{b_{j'}}} \leq 0$. As the first order partial derivative is non-positive $\forall \eta_{f_k}^{b_j}$s, thus, it can be concluded that Ξ_{π}^{het} is non-increasing in terms of $\eta_{f_k}^{b_j}$ which concludes the proof of Proposition 3.4.1.

A.3 Proof of Monotonicity of $a_{f_k}^{u_i}$

In order to analyze the monotonicity of Ξ_{π}^{het}, let us take first order partial derivative of it $\forall\ a_{f_k}^{u_i}$s in the following.

$$\frac{\delta \Xi_{\pi}^{\text{het}}}{\delta a_{f_k}^{u_i}} = \frac{\rho_{f_k}^{u_i}}{U}\left\{\Lambda^{stor} - \phi_d + \prod_{i' \in \mathcal{U}_{b_j} \setminus i}\left(1 - a_{f_k}^{u_{i'}}\right)\tau_2\right\}, \tag{A.5}$$

where τ_2 is shown in equation (A.6)

$$\tau_2 = \phi_d - \phi_b \eta_{f_k}^{b_j} - \phi_{BS}\left(1 - \eta_{f_k}^{b_j}\right) + \left(\phi_{BS} - \phi_C\right)\left(1 - \eta_{f_k}^{b_j}\right)\prod_{j' \in \mathcal{B} \setminus j}\left(1 - \eta_{f_k}^{b_{j'}}\right). \tag{A.6}$$

Notice that in both equations (A.5) - (A.6) the insider terms contains probabilistic values. The term $\prod_{i' \in \mathcal{U}_{b_j} \setminus i}\left(1 - a_{f_k}^{u_{i'}}\right)$ itself does not draw any conclusion. Similarly, τ_2 also does not reflect a clear notion. Depending on the values of caching placement probabilities, $\frac{\delta \Xi_{\pi}^{\text{het}}}{\delta a_{f_k}^{u_i}}$ may either be greater or less than equal to 0.

$$\frac{\delta \Xi_{\pi}^{\text{het}}}{\delta a_{f_k}^{u_{i'}}} = \frac{\rho_{f_k}^{u_i}}{U}\left\{\left(1 - a_{f_k}^{u_i}\right)\prod_{l \in \mathcal{U}_{b_j} \setminus i}\left(1 - a_{f_k}^{u_l}\right)\tau_2\right\}. \tag{A.7}$$

In a similar reasoning as above, it is claimed that $\frac{\delta \Xi_{\pi}^{\text{het}}}{\delta a_{f_k}^{u_{i'}}}$ may either be greater or less than equal to 0. Hence, it is concluded that there is no clear conclusion about the monotonicity of Ξ_{π}^{het} in terms of $a_{f_k}^{u_i}$s. This therefore, completes the proof of Proposition 3.4.2.

A.4 Proof of Convexity

Recall that if the second derivatives of a function $\mathcal{S}$ exist at each point in its $\mathbf{dom}(\mathcal{S})$, then $\mathcal{S}$ is convex if and only if $\mathbf{dom}(\mathcal{S})$ is convex and its Hessian is positive semidefinite. Also, $\mathcal{S}$ is concave if and only if its $\mathbf{dom}(\mathcal{S})$ is convex and Hessian is negative semidefinite [53]. In order to show that Proposition 3.4.4 is true, first, let us derive the elements of the Hessian of equation (3.45).

$$H_{11} = \frac{\delta^2 \Xi_{\pi_c}}{\delta \left(a_{f_k}\right)^2} = \frac{-\rho_{f_k}^{u_i}}{U}\left[U_c\left(U_c - 1\right)\left(1 - a_{f_k}\right)^{U_c - 2}\right]\tau_3, \tag{A.8}$$

where τ_2 is shown in equation (A.9).

$$\tau_3 = \phi_d - \phi_b \eta_{f_k} - \phi_{BS}\left(1 - \eta_{f_k}\right) + \left(\phi_{BS} - \phi_C\right)\left(1 - \eta_{f_k}\right)^B \tag{A.9}$$

Here, notice that by the same properties used for Proposition 3.4.1 and 3.4.2, it is not reasonable to conclude whether $H_{11} \geq 0$, or $H_{11} \leq 0$ in equation (A.8).

$$H_{12} = \frac{\rho_{f_k}^{u_i}}{U} U_c \left(1 - a_{f_k}\right)^{U_c - 1} \tau_3 = H_{21}. \tag{A.10}$$

Equation (A.10) states that the Hessian is symmetric. Although the term τ_3 contains the probabilistic values, i.e. it may become greater or less than 0 as per the analysis in Proposition 3.4.2, for now, let us take $H_{12} = H_{21} \geq 0$.

$$H_{22} = \frac{-\rho_{f_k}^{u_i}}{U} B_1 \left[\left(\phi_{BS} - \phi_C\right) B \left(B - 1\right)\left(1 - \eta_{f_k}\right)^{B-2}\right], \tag{A.11}$$

where B_1 is shown in equation (3.46). Finally, since $\phi_C >> \phi_{BS}$, right side of equation (A.11) is greater or equal to zero i.e. $H_{22} \geq 0$.

Now, let us aim to calculate whether the Hessian is positive semidefinite or negative semidefinite. Remind that if the Hessian is positive semidefinite (negative semidefinite), then all of its eigenvalues must be non-negative (non-positive) [54]. So, to find the answer, the eigenvalues of the Hessian matrix is calculated in the following. In order to do that, the following Corollary is introduced.

Corollary 1. *The determinant of a $m \times m$ matrix is the product of its eigenvalues.*

$$\det(\mathrm{H}) = \prod_{i=1}^{m} \lambda_i, \tag{A.12}$$

where, λ_i denotes the eigenvalues.

Proof. If the matrix, H has distinct eigenvalues λ_i, then it can be written as $H = S\Lambda S^{-1}$, where S is unitary matrix and Λ is a diagonal matrix. The diagonal elements of Λ are the

eigenvalues [54]. If $i = 2$ (as in this case), it can be written as $\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$. Therefore, the determinant of H is expressed as follows:

$$\det(H) = \det(S)\det(\Lambda)\det(S^{-1}) = \prod_{i=1}^{2} \lambda_i, \tag{A.13}$$

where $\det(S) = \det(S^{-1}) = 1$. ■

Next, using Corollary 1 the determinant of the Hessian matrix is calculated in the following.

$$\det\left(H_{\pi}^{\text{hom}}\right) = H_{11}H_{22} - H_{12}H_{21}. \tag{A.14}$$

Remind that it is inconclusive to conclude whether $H_{11} \geq 0$ or $H_{11} \leq 0$. Furthermore, $H_{22} \geq 0$ and $H_{12} = H_{21} \geq 0$. Therefore, if $H_{11} \leq 0$, then $\det(H_{\pi}^{\text{hom}}) \leq 0$. In other words, the eigenvalues do not have the same sign. This therefore, ruled out the chance of having all non-negative (or non-positive) eigenvalues i.e., the Hessian has both positive and negative eigenvalues. Hence, the optimization problem in P_2 is not convex. Therefore, this concludes the proof of Proposition 3.4.4.