Design of Efficient Caching Algorithms for Mobile Edge Networks

Submitted in partial fulfillment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

Submitted by

Somesula Manoj Kumar

(Roll No. 716041)

Under the guidance of

Dr. Rashmi Ranjan Rout

and

Prof. D. V. L. N. Somayajulu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

TELANGANA - 506004, INDIA

OCTOBER 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

THESIS APPROVAL FOR Ph.D.

This is to certify that the thesis entitled, Design of Efficient Caching Algorithms for Mobile Edge Networks, submitted by Mr. Somesula Manoj Kumar [Roll No. 716041] is approved for the degree of DOCTOR OF PHILOSOPHY at National Institute of Technology Warangal.

Examiner

Research Supervisor

Dr. Rashmi Ranjan Rout
Dept. of Computer Science and Engg.
NIT Warangal, India

Research Supervisor

Prof. D.V.L.N. Somayajulu
Dept. of Computer Science and Engg.
NIT Warangal, India

Chairman

Prof. P. Radha Krishna
Head, Dept. of Computer Science and Engg.
NIT Warangal, India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

CERTIFICATE

This is to certify that the thesis entitled, Design of Efficient Caching Algorithms for Mobile Edge Networks, submitted in partial fulfillment of requirement for the award of degree of DOCTOR OF PHILOSOPHY to National Institute of Technology Warangal, is a bonafide research work done by Mr. Somesula Manoj Kumar [Roll No. 716041] under my supervision. The contents of the thesis have not been submitted elsewhere for the award of any degree.

Research Supervisors

Dr. Rashmi Ranjan Rout
Associate Professor
Dept. of CSE, NIT Warangal, India

Prof. D.V.L.N. Somayajulu
Professor
Dept. of CSE, NIT Warangal, India

Place: NIT Warangal

Date:

DECLARATION

This is to certify that the work presented in the thesis entitled "*Design of Efficient Caching Algorithms for Mobile Edge Networks*" is a bonafide work done by me under the supervision of Dr. Rashmi Ranjan Rout and Prof. D.V.L.N. Somayajulu and was not submitted elsewhere for the award of any degree.

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

S Manoj Kumar
(Roll No. 716041)

ACKNOWLEDGMENTS

It is with great pleasure that I acknowledge my sincere thanks and deep sense of gratitude to my supervisor, Dr. Rashmi Ranjan Rout sir, from whom I have learnt so much about how to conduct oneself being a teacher. His values as a teacher and as a human being, are something I would like to take forward both in my professional and personal life. He has been very supportive to me throughout my research period which has been full of 1's and 0's, and constantly encouraged me to do good work as a researcher. He has always given ample time to me for research discussions, technical help, and the revision of the work. I feel lucky to be associated with such a great human being in my life.

I also would like thank my co-supervisor, Prof. D.V.L.N. Somayajulu sir, who have inspired me immensely directly or indirectly throughout my association with NIT, Warangal. His integrity and the core values with which he works as a teacher, are something I aspire to imbibe into myself. He has been very supportive to me through out the Ph.D. period and his insightful comments have immensely improved the quality of the research work.

I extend my gratitude to all my Doctoral Scrutiny Committee members Prof. B. B. Amberker, Prof. S. G. Sanjeevi, and Dr. Ch. Venkaiah for their insightful comments and suggestions during oral presentations.

I am immensely thankful to Dr. Ch. Sudhakar, Prof. R.B.V. Subramaanyam, and Prof. P. Radha Krishna Heads of Dept. of CSE during my stay in the department, for providing adequate facilities. I wish to express my thanks to faculty members of Computer Science and Engineering department.

I thank my colleagues and friends at NITW, Sai Krishna, Mallikarjun, Sudarshan, Sandipan Maiti, Ramesh, Abhilash, Govind, Sumalatha, Satyanarayanan, Pavan Kumar, Preethi, Spoorthy, Suresh, Sanjib, Hem Kumar, Vinay Raj, Uma Maheswara Sharma, Greeshma and Satish who has made my life at NITW easy and helped in their own ways to better my research work.

I thank my close friends outside of NITW, Shekar, Gautam Reddy, Babu, Ganesh, Gurappa and Subramanyam just for being there for me always.

I thank my brothers Ganesh Kumar and Sravan Kumar who always motivated and encouraged me to achieve good things in life. I am sure if not for their support, I wouldn't have the courage to take this career route. They have given me all kinds of support throughout this Ph.D. period, be it emotional or financial. I will be forever thankful for the way they back me in every endeavor of mine.

Finally, I thank my parents who brought me on to this earth and who have given their blood and sweat to make me what I am today. They have made many sacrifices to provide quality education to me and my brothers. From my childhood, my father consistently reiterated the importance of education in life. I am always in awe of the way my father planned my education with all the financial constraints our family had. The dreams of my parents and sheer determination with which they have worked to provide for us, has made me to attain this level of education. I love you amma and nanna!

S Manoj Kumar

Dedicated to

My Parents Lakshmi Devi, Subramanyam, my brothers Ganesh and Sravan & My little Nephew Havish

ABSTRACT

The proliferation of mobile devices and processing capabilities escalates the demands for multimedia content like remote education, video conferencing, augmented reality, virtual reality, and video on demand. Multimedia applications consume more bandwidth on the Internet, causing huge network traffic. The overwhelming network traffic in the coming years affects the overall network efficiency with respect to quality of experience, energy consumption, and delivery latency. To meet the user demands, deploying the base stations densely (network densification) is one of the important techniques. The tremendous growth in mobile data traffic adds a burden on backhaul links, and this may cause congestion leading to long delays in delivering content to users. Mobile edge caching is one of the prominent techniques providing computation and communication capabilities along with network caching capability where the MEC nodes are co-located with the base stations, and the contents are brought closer to users. With the increase of Internet content and considering limited storage at edge nodes, the overall performance gain of the caching will be reduced in mobile edge networks. Therefore, caching popular content at appropriate nodes (to improve the overall performance) is an important research issue (in a MEC-based system) that has been addressed in this thesis.

The thesis focuses on efficient caching algorithms for mobile edge networks. In this thesis, the proposed approaches achieve better cache hit ratio and acceleration ratio. Firstly, an echo state network-assisted fuzzy logic-based cooperative caching mechanism has been designed to maximize the saved delay in mobile edge networks. An approximation algorithm has been presented to find the optimal caching strategy, and a fuzzy caching heuristic is proposed to find a sub-optimal caching strategy by predicting the content popularity distribution using the Echo stat network. Secondly, a cooperative caching strategy has been presented by considering heterogeneous user preferences, activity level, and uneven distribution of users in large-scale mobile edge networks. The user preference learning is modeled using long short term memory to capture the dynamic user behaviour, and a greedy cooperative caching algorithm is presented by considering the submodular optimization to optimize the caching strategy. Thirdly, a caching strategy has been presented by consid-

ering the user mobility across the MECs and randomness of contact duration in mobile edge networks. The user moving path and the sojourn time of a user are modeled using the Markov renewal process. A greedy caching algorithm is proposed to optimize the caching strategy, and a genetic algorithm is presented to solve large-scale problems. Finally, a cooperative cache updating strategy has been presented. Considering the dynamic nature of the content popularity, high dimensional parameters, and for an intelligent caching decision, the problem has been modeled as a partially observable Markov decision process and presents an efficient deep reinforcement learning algorithm. Performances of proposed approaches have been evaluated through simulation.

Keywords: Proactive caching, Popularity prediction, Fuzzy logic, Cooperative caching, User mobility, Submodular optimization, Mobile edge networks, Genetic algorithm, Machine Learning, User preference learning, LSTM, Multi-agent deep reinforcement learning, partially observable Markov decision process.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

5G	Fifth Generation Wireless
A2C	Advanced Actor Critic
A3C	Asynchronous Advantage Actor Critic
ADMM	Alternating Direction Method of Multipliers
AI	Artificial Intelligence
AP	Access Point
BS	Base Station
BTPP	Back propagation through time
CDN	Content Delivery Network
CPP	Content Placement Problem
C-RAN	Cloud Radio Access Network
D2D	Device to Device
DC	Data Center
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DQL	Deep Q-Learning
DQN	Deep Q-Network
EM	Expectation Maximization

FC	Femto Caching Algorithm
ESN	Echo State Network
FBS	Femto Base Station
FCA	Fuzzy logic-based Caching Algorithm
FIFO	First In First Out
FIS	Fuzzy Inference System
GA	Genetic Algorithm
ILP	Integer Linear Programming
IoT	Internet of Things
ISP	Internet Service Provider
LFU	Least Frequently Used
LP	Linear Programming
LSTM	Long Short Term Memory
LRU	Least Recently Used
LTE	Long-Term Evolution
MADRL	Multi-Agent Depp Reinforcement Learning
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MARDDPG	Multi-Agent Recurrent Deep Deterministic Policy Gradient
MBS	Macro Base Station
MCC	Mobile Cloud Computing
MDS	Maximum Distance Separable Codes
MDP	Markov Decision Process
MEC	Mobile Edge Computing
MEN	Mobile Edge Network
MF	Membership Function
MINLP	Mixed Integer Non-Linear Programming

MPC	Most Popular Content
NEF	Network Exposure Function
NFV	Network Function Virtualization
PBS	Pico Base Station
PDA	Personal Digital Assistant
POMDP	Partially Observable Markov Decision Process
PPP	Poisson Point Process
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RAR	Relaxation and Rounding Algorithm
RDDPG	Recurrent Deep Deterministic Policy Gradient
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RSU	Road Side Unit
SBS	Small Base Station
SCN	Small Cell Network
SDN	Software Defined Networks
TD- error	Temporal Difference error
VoD	Video on Demand
WiFi	Wireless Fidelity
WTD	Wireless Topology Discovery

Chapter 1

Introduction

The proliferation of mobile devices, processing capabilities and accelerated advancements in multimedia applications escalate the demands for multimedia content like video conferencing, virtual reality, augmented reality, video on demand and remote education. This multimedia content utilizes extra resources for transmission on the Internet, which leads to the exceptional growth of network traffic, imposes a massive load on the backhaul [1]. According to the Cisco survey overall mobile data traffic anticipated to rise 7-fold from 2017 to 2022 [2]. To meet user demands and deal with the overwhelming traffic, network densification (deploying the base stations densely) is a fundamental technique in mobile edge networks (MEN) [3]. A significant part of backhaul traffic is the duplicate downloads of some popular content [4]. Thus, mobile edge caching is a prominent technique that utilizes the edge nodes as caching nodes to bring the contents closer to users, enhancing the user quality of experience (QoE) and alleviating the burden on backhaul and core network [3]. The mobile edge networks have merits over conventional network architecture regarding latency, bandwidth, energy, etc. *Latency*: Bringing the computation and caching abilities near users will reduces communication delay, particularly for the video content delivery and computation offloading. In [5], authors have proven that offloading latency-sensitive applications like health applications and highly interactive and computationally intensive applications like AR attained significant improvement.*Bandwidth*: In [6], it has been shown that bringing servers near users preserves the operation cost for computation-intensive and bandwidth-hungry applications by 67%. Studies show that by employing proactive caching

mechanisms, the burden on backhaul reduces up to 22%. The authors [7] claimed that more significant benefits could be achieved with expanded cache storage. *Energy Efficiency*: In [8], authors demonstrated that the energy consumption of nano data centers (DCs) in fog preserves the energy influenced by the type of application, ratio of active and idle time of DCs, and type of access network. The energy consumption of various applications in LTE and WiFi networks is studied and shown that significant energy saving achieved with edge computing [5]. *Context Information Utilization*: The detailed context information comprises device level and network information obtained with the help of edge nodes placed at various locations in the radio access network [9]. In MEN, the edge servers associated with the BSs can accommodate the location-based applications that use the context information to enhance the user experience and allocate the resources effectively. So, the edge nodes can serve a massive amount of duplicate content requests, and hence there will be a reduction in the service delay and the content delivery distance. Therefore, this can support latency-critical mobile applications in a mobile edge computing framework.

With the increase of Internet content and considering limited storage at MECs, the overall performance gain of the caching will be affected in mobile edge network. Due to limited cache capacity at base stations (BS), caching popular content proactively reduces the load on content servers during off-peak time. It is observed that the content popularity follows Zipf distribution and this states that most of the content requests come for a small number of frequently accessed top ranked content [10]. Therefore, caching popular content at appropriate nodes (to improve the overall performance) is an important research issue which has been addressed in this thesis. The above mentioned observations motivate the present work for designing efficient caching strategies for data delivery in MEN. Further, based on the cooperation among the BSs, caching can be classified as cooperative and non-cooperative caching [11]. The non-cooperative edge caching may experience long delays when a large number of contents are required to be fetched from content servers. In cooperative caching, different BSs share their content, and this forms larger cache storage. Therefore, the cooperative caching mechanism has a better cache hit ratio compared to non-cooperative caching and improves the quality of service (QoS) [11]. Hence, the crucial research problem is to take caching decisions including the content to be cached and place-

ment of the cache content. The problems related to content caching has been addressed in MEN. The contributions in this thesis are as follows:

- **Deadline-aware content caching using echo state network integrated fuzzy logic for mobile edge networks**: This work presents an echo state network-assisted fuzzy logic-based cooperative caching mechanism in mobile edge networks. An approximation algorithm has been presented to find the optimal caching strategy. Further, this work considers a proactive caching mechanism by predicting the content popularity distribution using the Echo state network, where the popularity is predicted by leveraging the user context information. A fuzzy caching heuristic is proposed to find a sub-optimal caching strategy. The proposed mechanisms improve the cache hit ratio, acceleration ratio and the number of contents delivered.

- **User Preference based Cooperative Cache Placement for Mobile Edge Networks**: This work presents a cooperative caching strategy by considering heterogeneous user preferences, activity level and uneven distribution of users in large scale mobile edge networks. In this work, user preference learning is modelled using long short term memory to capture the dynamic user behaviour, and further content clustering is presented to identify the relation between users. A greedy cooperative caching algorithm is presented by considering the submodular optimization to optimize the caching strategy. The proposed mechanism improves the cache hit ratio, acceleration ratio and cache utilization.

- **Contact duration-aware cooperative cache placement with user mobility across MECs (i.e., BS) using genetic algorithm for mobile edge networks**: This work presents a caching strategy by considering the user mobility across the MECs and randomness of contact duration in mobile edge networks. In this work, the user moving path and the sojourn time of a user is predicted using the Markov renewal process. A greedy caching algorithm is proposed by considering the submodular optimization to optimize the caching strategy. Further, a genetic algorithm is presented to solve large-scale problems. The proposed mechanisms improve the cache hit ratio and acceleration ratio.

- **Cooperative Cache Replacement using Recurrent Multi-Agent Deep Reinforcement Learning for Mobile Edge Networks**: This work presents a cooperative cache updating strategy. The content popularity is time-varying and unknown in reality, so the assumption on content popularity known in advance makes it less practical. Therefore, considering the dynamic nature of the content popularity, high dimensional parameters, and for an intelligent caching decision, the problem is modelled as a partially observable Markov decision process. Further, an efficient deep reinforcement learning algorithm has been presented by embedding the long short-term memory network into a multi-agent deep deterministic policy gradient formalism. The proposed algorithm improves the cache hit ratio, acceleration ratio and cache reward.

The rest of this chapter is organized as follows. Motivation behind the work has been presented in Section 1.1. In Section 1.2.1, a deadline-aware caching using echo state network integrated fuzzy logic for mobile edge networks has been highlighted. Content popularity distribution prediction and fuzzy caching algorithm is presented in this section. In Section 1.2.2, a cooperative cache placement in mobile edge networks with user preference based learning has been presented. A greedy algorithm is also presented in this section. In Section 1.2.3, contact duration-aware cooperative cache placement for mobile edge network has been highlighted and a greedy algorithm is presented. A genetic algorithm is also presented in this section. Section 1.2.4 describes the cooperative cache updating strategy in mobile edge networks. Section 1.3 discusses experimental setup details. The organization of the thesis has been presented in Section 1.4.

1.1 Motivation and objectives

The recent advancements in mobile and multimedia applications accelerate the demand for more resources on the Internet. To accommodate the user demands and manage the network traffic generated with these improvements, deploying base stations densely is a solution. The dense deployment of edge nodes causes more burden on backhaul links by accessing the same content. Therefore, to alleviate the burden on backhaul links, the most accessed

content needs to be cached at the edge of the network, and this technique is known as edge caching. Edge caching brings the most frequently accessed content stored near users by reducing the traffic generated by these requests. Hence, the edge nodes can serve the user-requested content with less delay and minimal congestion at the core network and this improves the quality of the service in the network. However, the effective cache utilization is reduced when the individual edge nodes with limited cache capacity make their caching decisions independently. A practical solution is to facilitate cooperation among edge nodes by sharing the content. Different edge nodes share their content in cooperative caching, which forms more extensive cache storage and enables cache diversity [11]. One of the significant constraints in mobile edge networks is the limited cache capacity. Therefore, the primary concern for mobile edge networks is the issue of effectively utilizing the limited cache capacity by placing the appropriate content at each edge node. Recently, proactive caching mechanisms have been designed for efficient data delivery in mobile edge networks by proactively caching the most popular content at each edge node.

With the rapid growth in time-critical and delay-sensitive applications like video streaming, Internet of Things (IoT), and financial applications need a response within a deadline (i.e., a specific time limit) [12]. The deadline determines the maximum allowable time for the response received for the request [13]. Some applications like health care and financial transactions demand the guarantee of timeliness strictly (hard deadline), whereas some IoT applications may tolerate the delay (soft deadline). If a request is not served within the deadline, the quality of service would be affected, and this in turn affects user QoE. Hence, to improve the user QoE, the request deadlines must be satisfied. Therefore, MECs should cache content cooperatively by considering the deadline, and limited caching capacity. Most of the existing proactive caching mechanisms assume that all users have homogeneous preferences like content popularity, homogeneous activity level, and uniform user distribution. However, the assumption made by the previous works is not realistic and not valid based on the recent study. Hence, designing the cooperative cache placement problem in a realistic scenario where unevenly distributed users, heterogeneity of the user preferences, and activity level in large-scale mobile edge networks is a significant challenge. Most of the existing literature assumes static network models where all

the users remain static throughout the data transfer time, and the user can download the requested content from the associated base station. Caching content by considering user mobility and randomness of contact duration is an important research issue. In the proactive caching mechanism, the caching decisions are based on the popularity of the content. In the literature, earlier works consider the content popularity is either known in advance [14] or content popularity be predicted [15, 16]. Practically, the content popularity may be time-varying, so the above assumption (known in advance) makes it less practical. In contrast, popularity prediction requires user association, and further user preferences may vary in different contexts, such as personal information, topology, location, etc [16]. For taking the caching decision, futuristic content popularity information may not be available. Therefore, designing a cooperative cache replacement problem in a realistic scenario where the edge nodes are unaware of the content's popularity is a significant challenge. Thus, this thesis also focuses on the above mentioned observations to increase the hit ratio while minimizing the latency and congestion in the mobile edge networks. The above mentioned challenges motivate the present work towards efficient caching mechanisms for data delivery in mobile edge networks. The major objectives of this thesis are as follows.

1. Design of a deadline-aware cache placement scheme for the mobile edge network to maximize the saved delay with capacity and deadline constraints.

2. Design of a user preference based cooperative caching scheme for mobile edge networks to maximize saved delay by considering the uneven distribution of users, heterogeneous user preferences and limited cache capacity.

3. Design of a contact duration-aware cooperative cache placement scheme with user mobility across BSs for mobile edge network to maximize the saved delay.

4. Design of a cooperative content replacement mechanism in the obscene of content popularity information for mobile edge networks to maximize saved delay.

1.2 Overview of the Contributions of this Thesis

In this section, an overview of chapter-wise contributions of this thesis has been presented. Each subsection presents summary of contributions of the corresponding chapter.

1.2.1 Deadline-aware Content Caching using Echo State Network Integrated Fuzzy Logic for Mobile Edge Networks

In this work, deadline-aware content placement mechanism has been proposed using the fuzzy logic to maximize the saved delay in wireless networks. The novelty of the approach lies in designing a caching mechanism for wireless networks (MEN) by considering limited storage at base stations, the deadline of content request and popularity prediction. Initially, the cache placement problem is formulated as an integer linear programming (ILP) problem. The solution is designed as relaxation-and-rounding based on the rounding technique. Further, a fuzzy logic based caching algorithm has been proposed by considering deadline, the benefit of caching content and content request distribution prediction for content placement decisions. Moreover, an Echo State Network (ESN) based prediction mechanism has been designed to predict the content request distribution for mobile edge network. The major contributions of this work are as follows:

- Formulate a content placement problem (CPP) as an integer linear programming problem in mobile edge networks with an objective to maximize the saved delay subject to cache capacity and request deadline.
- Design an approximation algorithm based on the relaxation and rounding technique to solve the integer linear programming version of content placement problem in MENs.
- Propose a fuzzy logic-based caching algorithm (FCA) to find the near-optimal solution by considering content request distribution, deadline of the content and benefit (distance) of caching content. The content request distribution prediction mechanism is designed using echo state networks.

Formulation of a Content Placement Problem

In this section, first the delay is modelled then the problem is formulated as Integer linear programming problem aiming to maximize the saved download delay with capacity and deadline constraints. Then the proposed problem is shown as NP-hard.

Approximation algorithm

In this section, the proposed ILP problem is converted into linear programming problem by relaxing the integer decision variables. Hence, the optimal solution can be found in polynomial time then round the fraction solution. The deterministic rounding algorithm [17] has been adopted by constructing the weighted bipartite graph for each content f of different content types c.

Fuzzy Caching Algorithm based on Content Prediction

The content placement problem is proposed to maximize the saved delay, since the popularity is determined by content request prediction (i.e., the appropriate content to be cached at each base station cooperatively requires the content popularity prediction). To address this issue, a fuzzy logic based cooperative content placement algorithm has been presented using content popularity prediction. The content request distribution has been predicted using the ESN (echo state network) [18]) model by considering the state of user content requests observed by network evaluation function. ESN predicts the content request distribution by establishing the relationship between the requested content and user context (user information). ESN trains the neurons using simple linear regression and it has fast convergence speed. The idea of the fuzzy caching algorithm is to cooperatively cache more popular content with minimal delay by considering content benefit, deadline and request prediction (popularity) to improve the performance in terms of hit ratio, acceleration ratio and the number of requests satisfying deadline. Further, a replacement strategy is presented for dynamic network. The proposed caching algorithms is implemented with python by utilizing the real-world movilens dataset for demand model. The performance metrics such as cache hit ratio, acceleration ratio, number of requests satisfying deadline and cache utiliza-

tion are taken to compare the proposed caching mechanisms with most popular, random, least recently used, cooperative and non-cooperative caching mechanisms. Simulation results show that the cache hit ratio and acceleration ratios are improved significantly using the proposed mechanisms compared to reference algorithms. Further, it is observed that there is an improvement of up to 20% on acceleration ratio, up to 18% on hit ratio and up to 24% on number of deadline satisfied.

1.2.2 User Preference based Cooperative Cache Placement for Mobile Edge Networks

Content popularity indicates the average interest of multiple users but not exhibits the individual user preferences [19]. Most of the existing literature considers that all users have the same content distribution (homogeneous popularity). However, various users have diverse preferences. The assumption made on homogeneous popularity ignores the users' preferences results in losing valuable information. Less than 20% of users generate 80% of traffic, which shows the users' activity level is heterogeneous [20]. The user activity level and user preferences, and unevenly distributed users introduce new challenges into mobile edge networks. In the literature, most proactive caching approaches ignored user behaviour, such as heterogeneous user activity levels and user preferences. Therefore, employing the individual user preferences and activity levels improves the cooperative caching strategy design.

This work aims to maximize the saved delay by considering the capacity and deadline constraints for accessing a large volume of data while reducing redundancy, congestion and delay. The content request deadline has been considered for generality and practicality, which is reasonable in latency-sensitive mobile and IoT applications but not sufficiently investigated. The work's novelty lies in designing a clustered cooperative cache placement mechanism for mobile edge networks with uneven user distribution, heterogeneous user preferences and activity levels into account. User preferences are predicted using the recurrent neural network mechanism LSTM using the historical user behaviour, and the users are clustered based on the predicted user preferences. Further, the clustered cooper-

ative content placement is designed by formulating the maximum saved delay problem. A solution is obtained to maximize saved download delay using a submodular function with matroid constraints for the cooperative content placement problem.

The contributions of our work are as follows:

- Design a user preference prediction mechanism by adopting the long short-term memory network.
- Design a user preference-based clustering mechanism and formulate a clustered cooperative caching problem as an integer linear programming problem in mobile edge networks to maximize the saved download delay subject to the deadline of the content and cache capacity.
- Propose a submodular optimization based cooperative content caching algorithm by utilizing the clustering and prediction mechanisms to solve the proposed problem.

User Preference Prediction

In this section, the user preferences are predicted by adopting long short term memory model. In the training phase, the data of each user is supplied into the LSTM network. Once the network is trained, the values are predicted and this indicate the number of times a user requests content in each time slot. Based on the prediction result, the user activity level and user preferences are computed.

User Clustering and Problem Formulation

In this section, the users are divided into logical groups based on the predicted user preferences. Each cluster has similar type of user and associated base stations. The content placement problem is formulated as Integer linear programming problem by maximizing the saved download delay with capacity and deadline constrains. Then the proposed problem is shown as NP-hard.

Content Placement algorithm using Sub-modular Optimization

In this section, a submodular optimization based greedy algorithm is presented to solve the proposed content placement problem. The submodular property with matroid constraint of the given problem has been proved and the given problem is converted into submodular optimization problem. Since the proposed problem satisfies the submodular property a greedy approximation algorithm has been presented. The proposed mechanism is compared with the global popular, local popular, femtocaching, cooperative and clustered cooperative caching mechanisms in terms of hit ratio, acceleration ratio and cache utilization. From the simulation results, it has been observed that the cache hit ratio and acceleration ratio are improved significantly using the proposed caching approach.

1.2.3 Contact Duration-Aware Cooperative Cache Placement with User Mobility Across MECs (i.e., BS) using Genetic Algorithm for Mobile Edge Networks

In this work, the cache placement problem in a realistic scenario has been considered where the users with different speeds intermittently connect to the BSs at irregular intervals. The users will frequently move between BSs and can download only parts of the requested content from different encountered BSs along the moving path. If the user fails to download the complete content from encountered BSs, then the requested content is downloaded from a macro base station (MBS), this in turn increases the overall delay and affects the QoS. Hence, the caching mechanism should consider the user mobility pattern. Although [21, 22] assumes the user mobility, the randomness of contact duration is not considered. According to [23], data transmission is associated with contact duration (sojourn time). If the contact duration is short, the user is moving at high speed and if the contact duration is long, it means the user moves at low speed. Thus, contact duration randomness caused by user mobility affects the transmission of data and this in turn affects the content placement. Therefore, the aim of this work is to design caching methods by considering limited resources, content popularity, deadline, the randomness of contact duration (speed of the user) and user mobility.

The novelty of this work lies in designing a content placement mechanism for dynamic networks where the moving users intermittently connect to the BSs at irregular intervals of time. User mobility is modeled as Markov renewal process to predict contact duration and user moving paths. Then the contact duration aware content placement is designed by formulating the maximum saved delay problem. For the contact duration aware content placement problem, a solution is obtained to maximize saved download delay using a submodular function with matroid constraints. Further, a heuristic search mechanism based on a genetic algorithm is designed to efficiently obtain content placement solution for large scale problems (the scenarios that scale to large video library sizes).

Major contributions of this work are as follows:

- Formulation of a mixed integer non linear programming problem for contact duration aware content placement problem: maximization of saved download delay subject to constraints, namely cache capacity and popularity of the content in mobile edge networks.
- Modeling user mobility as a Markov renewal process to predict the user moving paths and contact duration.
- Design of a greedy algorithm by adopting submodular optimization to solve the problem and development of a heuristic search mechanism based on a genetic algorithm to solve the content placement problem for large scale problems efficiently.

Greedy algorithm for contact duration aware cooperative content placement

In this section, a greedy algorithm is presented by predicting the user mobility and contact duration with the base station. The proposed greedy algorithm achieves a polynomial time complexity; the complexity grows with an increase in the number of contents. For real scenarios as the scale continues to increase (large scale problems where hundreds of users, tens of BSs), the complexity can be very high, making it impossible for implementation [24, 25]. Therefore, low complexity sub-optimal algorithms are required due to cheapness and delay sensitive implementations [25]. To address the system with a large number of nodes, contents and mobility paths, and to simplify the computational complexity, a

heuristic algorithm has been designed based on the genetic algorithm (GA). GA gives a near optimal and robust solution (video segment placement in content delivery networks [26] and base station placement in heterogeneous network [27]) for NP-hard problems. The proposed greedy and genetic algorithms provide improvement of up to 13 and 16 per cent on hit ratio compared with MPC, FC and MCFD, respectively.

1.2.4 Cooperative Cache Replacement using Recurrent Multi-Agent Deep Reinforcement Learning for Mobile Edge Networks

In the real world, the limited cache size restricts the mobile edge caching performance [28]. A simple solution is to devise efficient content placement mechanisms by considering user preferences and content popularity [22]. The effective cache utilization reduced when the individual nodes with limited storage make their independent decisions since they may redundantly cache popular content. Different edge nodes share their content in cooperative caching, which forms more extensive cache storage and enables cache diversity [11]. Generally, the caching decisions of various nodes depend on each other, but each edge node is aware of its own caching decision and unaware of the other nodes decisions. Make use of cooperative edge caching presents various technical problems. (1) To achieve cooperation, edge nodes should be aware of the neighbouring nodes caching state by sharing content, which causes considerable burdens [29]. (2) Efficient cooperation control needs an adaptive and dynamic framework. (3) Further, the designed caching mechanism has to tackle large-scale information induced by enormous data and information interaction. The conventional optimization mechanisms, such as dynamic programming and integer linear programming, can handle the first two problems [30]. Considering the dynamic nature of the content popularity, high dimensional parameters, and for an intelligent caching decision, the conventional optimization methods will not be suitable [30]. The recent success in reinforcement learning (RL) [31] and deep reinforcement learning (DRL) [16, 32] has encouraged this research work to use these learning mechanisms to tackle dynamic and complex systems.

This work aims to maximize the saved delay by considering the capacity and dead-

line constraints for accessing a large volume of data. Content request deadline has been considered for generality and practicality reasonable in latency-sensitive mobile and IoT applications. The novelty of this work lies in designing a cache mechanism for a dynamic environment where the time-varying nature of content popularity is unknown in advance for latency-sensitive applications by considering the limited storage at each edge node. Since each edge node observes its local state, the cooperative cache replacement problem is modelled as a partially observable markov decision process (POMDP) [33]. Further, a multi-agent actor-critic framework has been designed to manage nodes to coordinate the caching decisions. The contributions of this work are as follows:

- Formulate the cooperative cache replacement as an integer linear programming problem to maximize the saved download delay and further the designed problem is modeled as a POMDP based multi-agent decision problem to maximize the cumulative reward by ensuring the MEC servers' coordination.
- Design a recurrent multi-agent deep reinforcement learning-based cooperative cache replacement algorithm for mobile edge networks by devising a multi-agent actor-critic framework.

Multi-Agent Deep Reinforcement Learning Model for Cooperative Caching

In the real world, the environment has challenging conditions for a multi-agent system that demands cooperation among agents, such as partial observable agents and non-stationary nature. Therefore, the multi-agent decision problem has been modeled as a partially observable Markov decision problem and presented a multi-agent DRL framework for cooperative content replacement in MEN. In the proposed algorithm, LSTM is included in the actor-network and critic network. The inclusion of LSTM enables a way to remember the last communication (the effect of the actions on reward) received from other agents.

1.3 Experimental Setup

To evaluate the performance of the proposed caching algorithms, this thesis presents the system setting in this section. The proposed mechanisms has been experimented using Python with the TensorFlow platform. The proposed algorithms has been executed on a desktop with a dual-core Intel i5-5200U 3.20 GHz and 8 GB of installed RAM. A square region has been considered for simulation with an area of $500m \times 500m$. In the given simulation area, a cellular network is considered with 15 BSs associated with MEC servers and 90 mobile users, MECs are randomly deployed and connected and the mobile users' initial locations are uniformly distributed over MECs at the beginning of the simulations [24, 34, 35]. In the simulation, 500 contents have been considered with size determined uniformly at random from a range of [10MB to 100MB]. Content request probability follows Zipf distribution with $\gamma = 0.6$ [24]. Each content has a deadline picked randomly from [5 to 30s]. Each MEC can cache 10% of the total files. The data transmission capacity of MEC is 8 Mbps. The latency to fetch content from the base station to the user is specified using uniform distribution ranges from [10 to 30s]. Latency to fetch content from the content server to BS is taken as 80s. The proposed algorithms are compared with the existing algorithms based on publicly available real-world datasets available at WTD Project [36], MovieLens [37] and Lastfm [38]. The presented simulations results are an average of 100 runs.

1.4 Organization of the Thesis

The main focus of this dissertation is to design caching algorithms in mobile edge networks by considering dynamic environment conditions to improve cache hit ratio, acceleration ratio and cache utilization. The thesis has been organized in seven chapters.

Chapter 1: In this chapter, a brief introduction to mobile edge networks and objectives of the thesis have been presented. It also describes an overview of the major contributions and outline of the thesis.

Chapter 2: In this chapter, proactive caching, cooperative caching, mobility based caching,

coded caching, prediction based policies, user preference based policies, reactive caching and learning based policies are discussed. The challenges in content caching in mobile edge network have also been presented.

Chapter 3: In this chapter, a cooperative content placement mechanism has been proposed for mobile edge networks to maximize the saved download delay. The content placement problem has been formulated as Integer linear programming problem with an objective to maximize the saved download delay subject to cache capacity, request deadline and popularity of the content. An approximation algorithm is presented to solve the proposed ILP problem. Further, a fuzzy logic based caching algorithm is proposed to find near-optimal solution for large scale problems.

Chapter 4: In this chapter, a user preference based cooperative caching mechanism has been proposed for mobile edge networks. LSTM is used to model the user preference prediction and further, a greedy algorithm is presented to cache the content.

Chapter 5: In this chapter, contact duration aware cooperative caching mechanism has been presented. The mobility and contact duration is modeled with Markov renewal process and further, a greedy algorithm is proposed to cache the content at MEC nodes.

Chapter 6: A recurrent multi-agent deep reinforcement learning algorithm has been presented to dynamically update the content in mobile edge networks. An LSTM model is adopted to reduce the instability produced by partial observable environment. Inclusion of the LSTM enables a way to remember the last communication (the effect of the actions on reward) received from other agents.

Chapter 7: This chapter summarizes the work, outcomes of the contributions and future scopes for expansion of the work.

Chapter 2

Literature Survey

The evolution of telecommunication and information technology has witnessed the five generations of mobile cellular networks over the past two decades. At the same time, the growth in mobile devices and multimedia applications enables the user demands for mobile networks for lower latency and higher data rates. The conventional cellular network with base station-centric architecture no longer satisfies the demands. Therefore, in the 5th and sixth generations, mobile cellular networks emerge from conventional BS-centric [39] to content-centric [40], device-centric networks. Furthermore, mobile devices are becoming smarter in their computing. Modern machine-type devices like sensors, wearable devices, and human-type devices lead to huge machine-to-machine communications. These machine-type communications introduce additional challenges like low latency, power consumption, and limited processing abilities for cellular systems [41]. Different approaches have been proposed in the literature like mobile cloud computing, cloudlets [42] and edge computing to address the challenges raised by machine-type communications. Mobile cloud computing (MCC) consists of a pool of servers called cloud and clients to address the challenges provided by machine-type communications. MCC offloads the computational tasks to the cloud to support multiple platforms and provide adequate resources. Even though MCC addresses some issues, it suffers specific problems like long latencies and a burden on backhaul due to bandwidth limitation. Like MCC, Cloudlet is a small-scale data center near deployable users, energy-efficient, and self-managed [43]. Widespread deployed cloudlet management is challenging. Mobile edge networks are a solution to ad-

dress the demerits of the MCC and cloudlet.

Bringing the network resources (communication, caching, and computing), contents, and functions near users by employing the NFV and SDN technologies is the primary objective of mobile edge networks. Each network resource invites distinct challenges [29]. The evolution of various smart devices and applications like the Internet of things (IoT), augmented reality, virtual reality, haptic communications, and tactile Internet generate a huge volume of content that requires additional resources on the Internet, and this leads to the exceptional growth of network traffic and imposes a massive load on the backhaul [1]. Thus, mobile edge caching (MEC) is a prominent technique that utilizes the edge nodes as caching nodes to bring the contents near users, alleviating the backhaul and core network burden that enhances user quality of experience (QoE) [3]. So, the edge nodes can serve a massive amount of duplicate content requests, which reduces the service delay and reduces the content delivery distance. Therefore, this can support latency-critical mobile applications in a mobile edge computing framework. Content caching has been investigated extensively in the literature like web caching [10, 44] and information cnetric networks [45, 7, 46]. In [44], Ali *et al.* have presented caching approaches and studied prefetching mechanisms to improve web performance. Podlipnig *et al.* [47] have discussed the merits and demerits of cache updating mechanisms. A survey on caching strategies in an information-centric network is presented by [7]. However, the caching mechanism mentioned above may not be applied directly because of the wireless networks characteristics, and further the caching mechanisms in cellular networks need to be studied in detail.

2.1 Mobile Edge Networks Architecture

In this section, first, an overview of mobile edge computing (MEC) and its architecture has been provided. MEC is a prominent solution to minimize the delay between users and cloud servers. As shown in Fig. 2.1, multiple scenarios exist with different edge paradigms. In [48], authors studied different edge paradigms and showed each paradigm's merits and demerits in detail.

MEC affords caching ability, mobility, high computational capability, local and context-

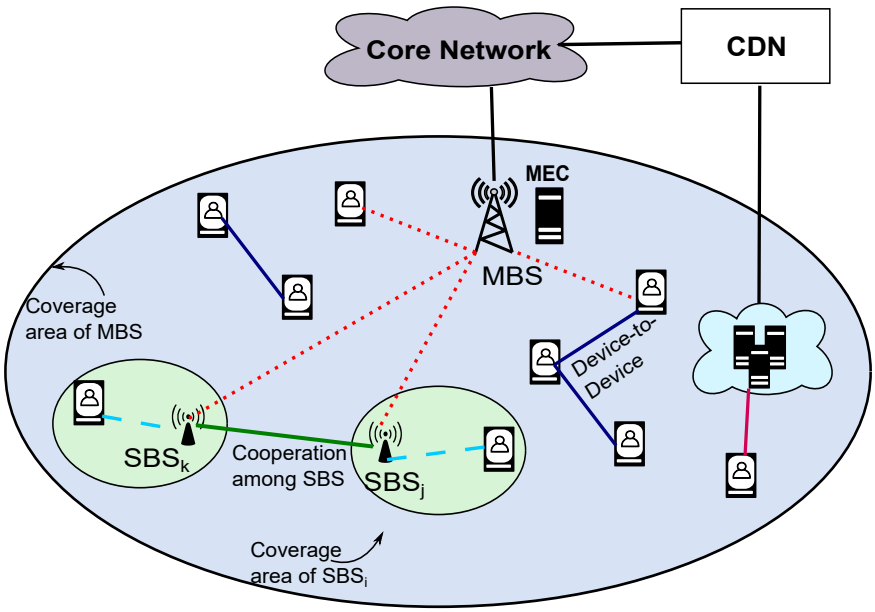


Figure 2.1: Mobile Edge Network Architecture

aware support to the end-users by bringing proximity to users. Furthermore, MEC offers high bandwidth, ultra-low latency, and energy-efficient environments to delay sensitive applications like IoT for better responsiveness than other edge paradigms [48]. It can be seen from Fig. 2.1 that the MEC servers are deployed along with the base stations to support the task computation at the edge and forward the tasks to the remote cloud servers for the applications to fulfil high computational requirements.

2.2 Mobile Edge Caching

Caching content at the edge nodes in MEN has advantages. Because of the densely deployed base stations, the MEN will be hybrid and heterogeneous. In conventional cellular networks, the requested content is obtained from distant Internet CDN. Therefore, in MEN, the cache can be placed at several positions to bring content near users. A mobile edge caching architecture is shown in Fig. 2.2. With the evolution of technology, BS and the cheaper storage cost lead to placement of storage at heterogeneous BSs is reasonable. However, the densely deployed BS needs high-cost backhaul links among the core networks and BSs [49]. The backhaul links witness more duplicate content transmission

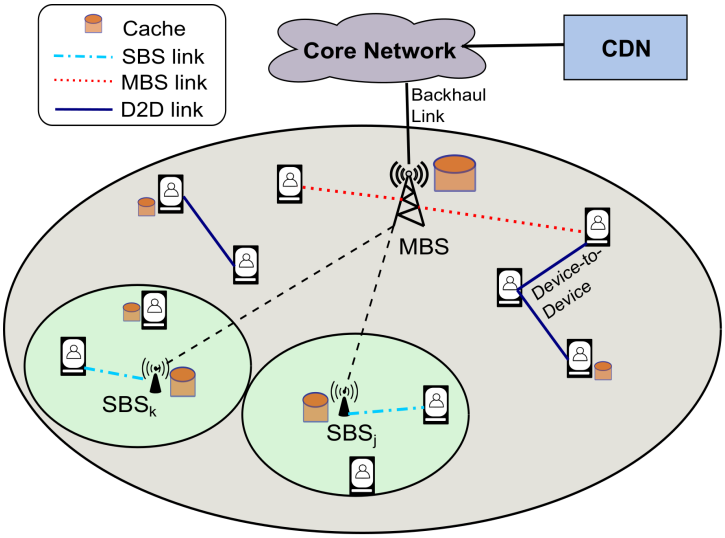


Figure 2.2: Edge Caching Architecture

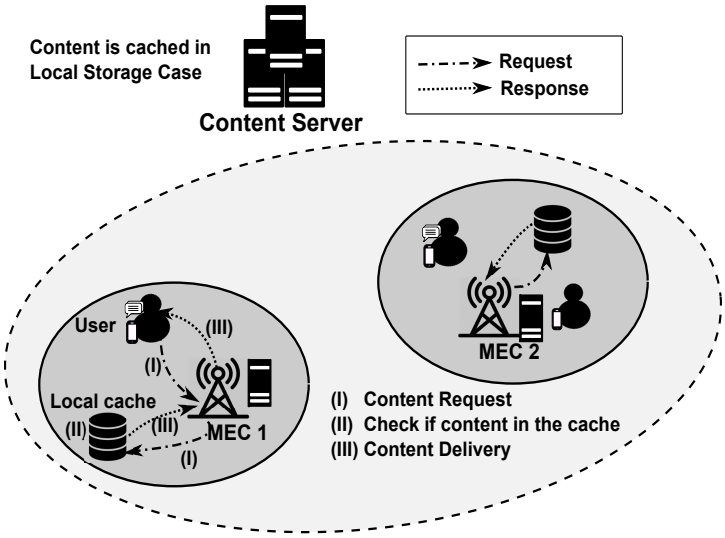


Figure 2.3: Content cached at Local Cache

because of the popular content [49, 50]. When multiple users from several locations request a large number of contents repeatedly at different times, the need to serve the user demands by fetching content from the central CDN leads to a vast amount of duplicate content producing the backhaul traffic. If the edge node caches most of the users desired content at its local storage, it can be served immediately without fetching from the central node every time, reducing duplicate content [51, 52]. Therefore, the basic problem in caching for mobile edge networks is to decide *what* content to cache, *where* to cache, and *how* to cache.

The cache storage units can be placed at several places in MEN. In the conventional

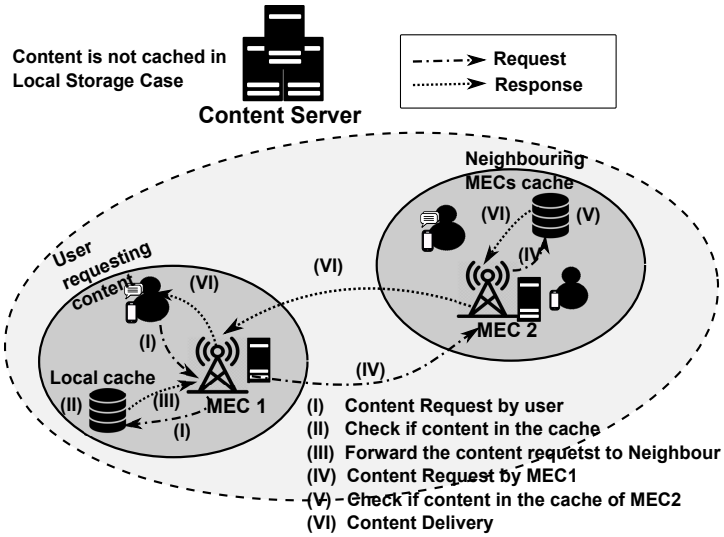


Figure 2.4: Content cached at Neighbour Cache

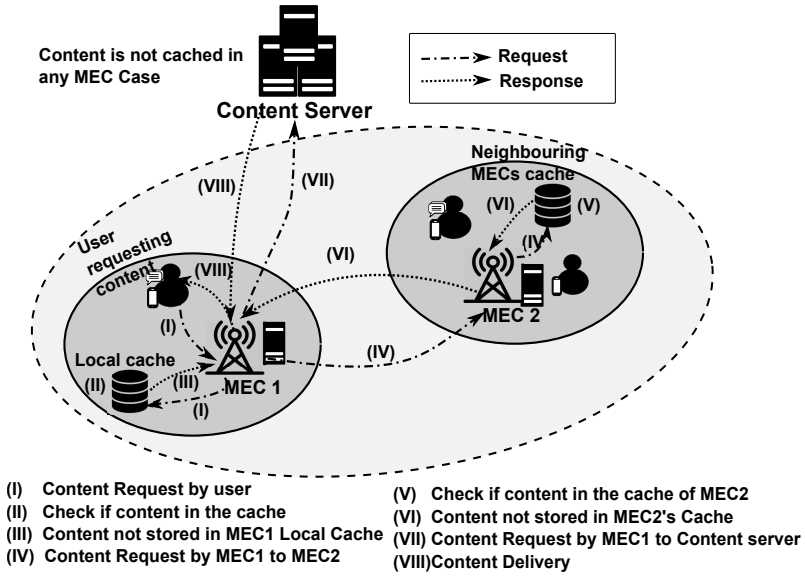


Figure 2.5: Content not cached at any edge node

centralized cellular network, the cache is deployed only at the core CDN. In contrast, in the MEN, the cache can be deployed at multiple places like radio access networks (RAN) (cloud RAN, fog RAN) [53], user devices (device to device networks) [54], HetNet (Heterogeneous networks) [24], macro (MBS) [55, 56], small BS (SBS), femto (FBS) [57], and pico (PBS) cellular networks. As shown in Fig. 2.3, the content can be served to the requested user based on availability of the content at local cache deployed at BS. If the requested content is not available at local BS then corresponding BS serves the requested

user by obtaining the content from nearby (neighbouring node) BS as shown in Fig. 2.4 otherwise the content will be fetched from the central server as shown in Fig. 2.5. This thesis focuses on the design of cache placement (mainly *what* and *where* to cache the content).

- *Caching at BS*: Caching the content near users at the base stations is a good solution that provides lower latency and better throughput. Caching content at BS has challenges like inter-cell inference, connection uncertainty, and limited coverage. In HetNets overlapped BSs imposes additional complications in addition to inter-cell inference, connection uncertainty, and limited coverage [58]. HetNets expand the MBS by placing the SBSs (Femto base station (FBS) and Pico base stations (PBS)) to increase the throughput and network coverage [59].

    1. *Caching at MBS*: MBS coverage is higher than SBS coverage and can accommodate more users lead to better performance in terms of hit rate. The performance in MBS caching can be improved by proactively caching the more popular content from the central node. The proactive and reactive caching at MBS has been studied in [60] for scheduling the backhaul and wireless channel and shown that the stalling probability has been reduced and the capacity of the video capacity has been increased. In [61], the resource allocation in MBS was solved using a heuristic approach by proving the problem as NP-hard. In [62], a HetNet with two levels which consist of MBS level and SBS level is introduced. This work aims to maximize the hit ratio. Users can be served in four different ways like serving immediately by corresponding SBS, serving immediately by corresponding MBS, serving from MBS through corresponding SBS, and serving from nearby SBS through MBS. In [63], a matching algorithm has been presented to map the content to the appropriate BSs (SBS and MBS) in HetNets. The limitations of this work are that they consider only one content in the entire network and this leads to intra-network overhead and restricts the cache hit rate.

    2. *Caching at SBS*: Small base stations have a lower coverage area than MBS, so

they are placed densely under an MBS for better coverage. With limited cache capacity, SBS brings the content near users, which boosts the higher data rates and low transmission power and relieves the backhaul burden. For placement, SBS needs different caching schemes than MBS because of its dense nature. Cooperation among the SBS needs to be considered to avoid redundant data storage at the entire network and this provides higher data rates to improve cache storage utilization [51, 64, 57, 65]. In [57], the authors proposed a content placement problem (CPP) in cache-enabled SBS to minimize the expected download delay. Authors in [64] consider the cache update problem for dynamic content popularity and presented a learning-based caching mechanism to maximize the reward. Nevertheless, these works consider the caching and delivery phases individually. In [66], authors consider the D2D and SBS caching where the SBS serves various users, and user devices share content among other devices using the D2D communication. In [66], content caching and delivery are considered jointly to optimize network performance.

3. *Caching at FBS*: The fundamental idea of Femto caching is to substitute the backhaul capacity with storage capacity at small cell access points [57]. Femto caching is a cost-effective and flexible technique to effectively handle the extremely anticipated massive traffic (VoD) by placing content at the edge nodes. The femto caching architecture for video content is proposed in [57] and shown the two order magnitude improvement in throughput than architecture without helpers. Shanmugam *et al.* [24] have discussed the way to minimize expected download delay by assigning files to the helper (femtocells). They proposed a greedy approach with a factor of 2-approximation for the NP-complete problem. By considering the dynamic nature of topology and user mobility, dynamic Femto caching is presented in [67].

4. *Caching at PBS*: Pico caching is a cost-effective mechanism that assists in diminishing the load on the backhaul. PBS cooperates by sharing the content to reduce the burden on backhaul. Caching in HetNet has been considered in [68], where authors consider multiple PBS are deployed under an MBS. MBS serves

as a central controller in the considered system that takes caching decisions and assigns the appropriate content to the particular PBS. User requested content can be served by the associated PBS or from the nearby PBS or the MBS based on the availability of the content. In [69], the authors optimize the multicasting by placing content randomly at PBS and at the MBS tier to maximize the successful transmission probability in HetNet.

- *Caching at User Devices*: With the advancements in smartphones, computation and storage abilities have been increased. Hence, user devices serve other mobile users by caching and sharing the content. Caching the content at the user devices is known as D2D caching, and it is a critical technology where the mobile users get the content from the nearby users using D2D communications instead of fetching content from a long distant core network by exploiting the device storage [70]. In D2D caching, the BS tracks the status of each user device and is responsible for redirecting the user requests to an appropriate device. If the requested content is not available with any of the devices, BS serves the user request by fetching content through backhaul. In D2D caching, several devices form a cluster based on social relationships or common interests. D2D caching provides the benefits offered by D2D communications, such as throughput, energy efficiency, better spectrum utilization, and location-based services [29]. Moreover, the user devices cache the content based on their favorites, so it offers a higher cache performance and flexibility than caching at different places (MBS, SBS, and core network) [71]. In the literature, these studies explored the opportunistic D2D mechanisms, social relationship and shared interests [72]. In this thesis, caching content at the SBS has been considered to improve the acceleration ratio, cache hit ratio and cache utility in MEN by considering the user mobility, heterogeneity of user preferences and absence of content popularity.

2.3 Cooperative Caching

Based on the cooperation among the BSs, caching can be classified as cooperative and non-cooperative caching. The non-cooperative edge caching (content is not shared among

the BSs) may experience long delays due to the large number of contents required to be fetched from content servers. In cooperative caching, different BSs share their content and this forms larger cache storage. Therefore, the cooperative caching mechanism has a better cache hit ratio compared to non-cooperative caching and improves the user quality of service (QoS) [11].

In cache enabled cellular networks, base stations typically cache a set of contents cooperatively [3]. The user can get the interesting content either from its local base station or from the neighbouring base station set [11]. Khreishah *et al.* [55] have discussed collaborative caching to minimize the operation cost in a multi-cell coordinated system. The authors consider the coded and non-coded caching problems. In non-coded caching, the problem is formulated as integer programming to maximize the profit with capacity constraint and it is shown as NP-Complete. A fully polynomial-time algorithm is presented to solve the non-coded caching problem. The coded caching problem is formulated as linear programming, which is polynomial-time solvable. Peng *et al.* [49] have discussed a content placement problem (CPP) to minimize the average download delay of wireless networks by considering backhaul delay and physical layer processing. The CPP is formulated as mixed-integer non-linear programming and its hardness is shown. Since the proposed problem is complex, the authors relaxed the problem into a distance of concave optimization problem and presented a successive convex approximation algorithm with low computational complexity.

Chen *et al.* [18] have investigated cooperative caching and transmission mechanisms in cluster-centric SCN. In cluster-centric SCN, BSs are grouped into clusters, and each BS reserves the cache space for the most popular content, and the remaining cache space is for less popular content. The coordinated multi-point mechanism is adopted to serve the users either with parallel or joint transmission. The authors consider the hexagonal grid design for clusters. The users are distributed based on the Poisson point process, and the analytical results on successful content delivery are provided to show the proposed transmission mechanisms for the user positioned at the cluster center. Psomas *et al.* [73] have investigated cache placement mechanisms at relays in cooperative communications. To address the placement problem, the authors proposed optimal amplify and forward relay

selection mechanisms, and the analytical results are presented to show the outage performance, coding gain, and diversity order of various caching mechanisms. Josilo *et al.* [74] have considered the bandwidth minimization in hierarchical cache networks through cooperative caching. The CPP is formulated as a binary integer programming problem to minimize the bandwidth in hierarchical networks and this is proved to be NP-hard. To solve the problem, the authors presented an approximation algorithm. Further, two low complexity distributed algorithms were presented.

To improve the user QoE, Tran *et al.* [11] have investigated a cooperative hierarchical caching framework in C-RAN. The cooperative CPP is formulated as an integer programming problem to minimize the download delay and it is proved to be NP-Complete. Octopus presents an efficient cache management scheme, including two low-complexity cache mechanisms. Further, a user request aware cache replacement mechanism is presented. Cui *et al.* [75] have discussed caching in ISP networks with software-defined networks to minimize the download delay, reducing inter-ISP traffic. The CPP is formulated as an integer programming problem with traffic and capacity constraints and shown as NP-Hard. A relaxation-rounding-based approximation algorithm is presented to solve the proposed CPP in ISP networks. Further, a low complexity heuristic solution is presented to solve the content placement problem for large-scale problems. Kumar *et al.* [76] have designed the consolidated cooperative cache placement and replacement schemes to reduce the content access delay in MEN. The authors in [77] devised a test-bed to evaluate the cooperative bit-rate adaptive caching scheme and consider the video caching and request routing using the video trans-coding mechanism.

Ayenew *et al.* [78] have discussed the content placement problem in a heterogeneous cellular network. The CPP is formulated as a 0/1 knapsack problem to maximize the cache hit probability with known popularity and presented a dynamic programming solution to obtain an optimal solution with minimized computational time. Ren *et al.* [79] have proposed a hybrid cooperative caching in MEN to improve the quality of service interns of service latency reduction and energy efficiency. The collaborative caching problem is formulated as integer programming to maximize the service latency and energy savings with popularity and capacity constraints. First, the BSs were logically grouped into clusters us-

ing the fuzzy c-means clustering algorithm. Second, a hybrid collaborative caching scheme has been presented using the Lagrange multiplier scheme. Further, a greedy approximation algorithm has been designed to give a linear complexity solution. Yang *et al.* [80] have discussed cooperative caching based on user access patterns. The interaction between the user, BSs, and contents were investigated using the tensor decomposition technique with a distance constraint.

The works mentioned above have not considered heterogeneous user preferences, user activity, user contextual information, the randomness of contact duration, and the absence of content popularity in cooperative settings. In contrast, this thesis considered designing efficient cooperative caching algorithms for the problems mentioned above in Chapters 3, 4, 5 and 6.

2.4 Mobility based Caching

Most of the existing works [11, 60, 24] focus on caching content cooperatively at BS for static networks. This assumption made by the existing works [11, 60, 24] is unrealistic in a dense network. In a realistic scenario, the users with different speeds intermittently connect to the BSs at irregular intervals. The users will frequently move between BSs and can download only parts of the requested content from different encountered BSs along the moving path. If a user fails to download the complete content from encountered BSs, then the requested content is downloaded from a macro base station (MBS); this, in turn, increases the overall delay and affects the QoS. Consider an example customers move around a shopping mall with three BSs. If a user wishes to download content, then the content should be replicated in all three BSs due to user mobility. Replicating the same content at three BSs is a wastage of resources, so disjoint content parts should be cached at the BSs to improve cache hit ratio and cache utilization. Hence, the caching mechanism should consider the user mobility pattern. Although [21] and [22] assume user mobility, the randomness of contact duration is not considered. According to [23], data transmission is associated with contact duration (sojourn time). If the contact duration is short, the user is moving at high speed, and if the contact duration is long, it means the user moves at low

speed. Thus, contact duration randomness caused by user mobility affects the data transmission, which in turn affects the content placement.

There are a few studies on user mobility. Guan *et al.* [34] have presented a mobility-aware cache placement problem in SCN (small cell network) to minimize the burden on the backhaul network. The mobility-aware CPP is formulated as an optimization problem for maximization of cache utility with capacity constraint and shown as NP-complete. Further, a polynomial-time heuristic algorithm has been proposed to maximize cache utility. In [34], authors have considered that the mobility paths of users are known in advance. In reality, users may not follow the paths predicted from historical user trajectories. Liu *et al.* [81] have presented a mobility-aware coded cache mechanism to improve the throughput in dense wireless networks. In this work, the authors jointly consider the user mobility, channel selection diversity, and content diversity in the optimization problem and proposed a modified mobility model based on discrete jumps. Further, the authors presented two lightweight heuristic solutions to solve MEC's coded probabilistic caching algorithm that enabled small cell networks to maximize throughput. In [81], authors considered only single user mobility, but mobility among different users has not been considered. Liu *et al.* [82] have proposed an optimal file allocation mechanism to minimize the download time in a small cell network and assumed that the user mobility obeys exponential distribution. The authors in [82] assumed that the user does not return once it leaves the base station.

Wang *et al.* [22] have proposed a mobility-aware cache placement scheme to maximize data offloading ratio by considering the advantage of inter contact time between users and proved this to be NP-hard. First, a dynamic programming solution has been presented to achieve lower computational complexity. Second, the proposed problem has been reformulated into a submodular optimization problem, and a greedy approximation algorithm has also been presented, which gives an approximation ratio of at least $\frac{1}{2}$. Chen *et al.* [83] have proposed a mobility-aware caching scheme to cache content at the base station and mobile device using user mobility. The CPP was formulated as integer programming to maximize the offloading ratio with user mobility and randomness of contact duration. The greedy approximation algorithm solves the proposed problem by converting the given problem into a submodular optimization problem. Ye *et al.* [84] have proposed a caching mechanism

for a decentralized multi-task learning problem based on mobility prediction and addressed the problem using hybrid jacobian and Gauss-Seidel proximal multi-block ADMM based mechanism. The sojourn time and mobility path are modeled with the Markov renewal process. In [21, 85], the authors considered the coded cache placement in small cell networks to minimize MBS load and presented a distributed caching paradigm using user mobility predictions. However, the above-mentioned works considered user mobility where randomness of the user contact duration has not been considered. In contrast, in this thesis, user mobility and randomness of contact duration have been considered in Chapter 5 to handle the dynamic scenarios.

2.5 Coded Caching

With the increasing demand for the content, increasing network traffic leads to more burden on the content delivery networks, resulting in under utilization of resources in off-peak time and more congestion at peak times. Therefore, duplicating the preferable content at the idle resources in the network by shifting the underutilized network resources to reduce the network congestion in peak hours. There are two phases in the network that operates like content placement and content delivery. Each phase has its own issues. In the content placement phase, the cache memory is restricted, and the congestion will not increase, whereas, in the delivery phase, the system will be affected by the rate required to serve the content and congestion. Hence, designing a caching mechanism to place the content at each base station in the given network minimizes the user request rate. Based on the comprehension of the nature of caching, there exist two types of caching mechanisms.

- *Local content placement:* In this mechanism, the content is replicated at the nodes near the requesting users to serve the content nearby. If a user requests content, and the requested content is available near the user, then that segment is served to the user locally, and the rest of the segments are fetched from the distant server using the unicast transmission. If multiple users request the same content and the content is served to the user locally, then the rest of the segments are served from the server using multi-cast transmission to users requesting. There exist several approaches

[52, 60, 11] in the literature which uses the content popularity to choose the most popular content and caches at appropriate location to enhance the benefit of caching. The popular content caching mechanisms achieve better results with larger caches to store a significant amount of the popular content locally.

- *Simultaneous multi-casting approach:* In this mechanism, content is cached to permit the server to fulfill the multiple users' demands with different requirements with a single multi-cast stream. A coding mechanism produces these streams [86, 87]. In this mechanism, each user downloads the coded stream and decodes the requested content. Hence, the coding mechanism needs to be designed to satisfy all possible user demands simultaneously.

Coded Caching: For all files F and users U each with cache of size $M \in \{0, \frac{F}{U}, \frac{2F}{U}, \frac{3F}{U}, \ldots, N\}$, $R^*(M) \leq R_c(M) = F(1-\frac{M}{U})\min\{\frac{1}{1+F\frac{M}{U}}, \frac{U}{F}\}$ is achievable. For general $0 \leq M \leq U$, the lower convex enveloped of these points is achievable. Where $R_c(M) = F(1-\frac{M}{U})\frac{1}{1+F\frac{M}{U}}$ which is a global caching gain and M cache size.

Maddah-Ali *et al.* [87] have investigated the coded multi-casting mechanism to enhance the benefit of caching and reduces congestion significantly. In a conventional caching mechanism, the benefit of caching depends on the size of the local cache. In contrast, in the proposed coded multi-casting, the benefit is higher with cumulative cache available at all users in the networks even though there exists no cooperation among the users. The problem is formulated as an information-theoretic formulation and further a coded caching mechanism has been presented to utilize local and global cache benefits. The authors have shown that the overall improvement can be the order of the users in the network. In [88], an effective decentralized coded cache mechanism is presented and proved that the proposed coded multi-casting mechanism attains the rate near optimal to the centralized mechanism.

Pedarsani *et al.* [89] have presented an online coded caching by considering a distributed framework with a single centralized server connected via a shared bottleneck link to multiple users having the limited cache size. The aim is to optimize the number of bits to be sent via shared links to satisfy the user demands by managing the caches of the centralized server and users. The shared link average rate is modelled using the Markov model.

The authors have proven that the proposed online algorithm has a similar performance as the offline caching mechanism. Karamchandani *et al.* [90] have presented a hierarchical coded caching mechanism by combining the coded multi-casting at individual layers and coded multi-casting across the layers in hierarchical content delivery networks. Niesen *et al.* [91] have studied the gain of coded caching for a caching system with non-uniform content popularities. The authors have shown that the optimal caching scheme in the multiple cache scenario needs the coding mechanism. Zhang *et al.* [92] have investigated the coded caching scheme for arbitrary content popularity and obtain an information-theoretic lower bound on expected transmission rate. In Chapter 5, coded caching has been considered to design an efficient contact duration aware caching scheme.

2.6 Learning based Caching

The fundamental problem in caching for mobile edge networks is *what* content to cache, *where* to cache, and *how* to cache. Many caching schemes exist using several optimizations, stochastic, and heuristic techniques to address some of these problems. The increasing dynamics in the mobile and wireless networks, such as diversity in content, number of users, content, and mobility, causes challenges and complexity to implement. Therefore, the inclusion of machine learning into the caching mechanism gives better decision capabilities to the mobile edge networks. Primarily machine learning techniques are classified into three classes: supervise, unsupervised, and reinforcement learning.

2.6.1 Supervised Learning based Caching

Supervised learning is one of the significant practical learning mechanisms where the model learns by example. This learning mechanism uses the idea of learning from the label (mapping from the input to output), similar to learning under a teacher's supervision. Supervised learning aims to design a model that approximates the mapping function. This learning technique is used in many real-world applications like spam filtering, fraud detection, classification of objects, and risk evaluation. Supervised learning is further divided into classification and regression. Many supervised learning algorithms (decision tree, ran-

dom forest, logistic regression, support vector machines and polynomial regression) exist in the literature [93, 28, 94, 95], and the algorithm is chosen based on the requirement.

Bastug *et al.* [93, 28] proposed a learning-based caching mechanism at user devices and BSs to enhance the user experience. In this work, the authors studied a proactive cache mechanism at BS where individual BS learns the popularity of content using supervised learning and collaborative filtering to fill the missing values in the popularity prediction matrix. Initially, the content popularity is computed by minimizing the least square error, and then the content is cached at individual BS greedily till the cache is full. Further, a proactive caching mechanism has been designed based on the D2D communications by caching content at user devices. In device caching, influential users are figured out by evaluating the centrality of the social users and cluster the users into logical groups using the K-means clustering algorithm. In this mechanism, the most popular content is cached to the most influential user in each cluster.

Thar *et al.* [94] proposed a popularity prediction based on supervised and DL in two stages and caches the content proactively at the BSs. First, the supervised learning algorithm collects the user request count to estimate the content popularity in future time. Second, deep learning is used to predict the request counts of content based on the collected content request count in the first step. The problem is formulated as the minimization problem to reduce the delay with capacity constraints. RNN (recurrent neural network) is used to evaluate the popularity prediction, and the caching decision is taken based on DNN (deep neural network) with multiple hidden layers and levels of nonlinear operations. Zhang *et al.* [95] have employed the supervised and unsupervised learning mechanism to make the efficient caching decision in SCN. The caching problem is formulated as integer programming to maximize the cache hit ratio with capacity constraint and shown it is NP-hard. Therefore, the authors use the learn-to-rank technique to predict the content popularity by utilizing content requests and forming logical groups using the K-means algorithm.

2.6.2 Unsupervised Learning based Caching

Unsupervised learning models are not supervised using the training data. These mechanisms explore the hidden patterns or data groupings with the given data. In supervised learning, labeled data is used for training, whereas unsupervised learning learns the patterns without labels. These mechanisms are perfect solutions for image recognition, customer grouping, cross-selling policies, and exploratory data analysis because of the capability to identify the data's differences and similarities. Unsupervised learning is divided into clustering, association and dimensionality reduction.

Clustering: Clustering is a mechanism to group the data into clusters based on more similarity among the data points within the cluster and less similarity among the data points outside the group or among the groups. Clustering is again classified into exclusive, hierarchical, overlapping, and probabilistic.

Association: Association rule is a technique to discover the association among the variables in the given data. Association rule mechanism defines items that appear collectively in the given data set, primarily used in market basket analysis. This technique does an efficient marketing plan by understanding the usages of the consumers empowers the enterprise to produce more valuable recommendations and better selling strategies. It can be observed that this association rule in retail and online marketing where the user who purchases item A (bread) is also direct to purchase item B (butter). A similar approach has been applied to content caching based on user preferences and correlation among the users [96, 97, 98]. In literature, various algorithms exist like Apriori, FP-Growth, and Eclat; among these, Apriori is the extensively used technique.

Dimensionality reduction: The more accurate results are produced when the data is more. However, the machine learning algorithms suffer from a massive amount of data in terms of performance like over-fitting and difficulty in visualization. Therefore, reducing the not useful features (dimensions) by maintaining data integrity from the vast data set eases the execution, known as dimensionality reduction. This technique is more generally used in data preprocessing.

Shen *et al.* [99] have presented unsupervised learning mechanisms to improve the

cache efficiency in an ultra-dense network. The caching problem is formulated as Integer programming to minimize the load on backhaul with capacity constraint and shown it is NP-hard. Machine learning-based caching mechanisms are presented to handle the difficulty of highly random content demands in ultra-dense networks. K-means algorithm was applied to find the hidden Spatio-temporal patterns of user demands at each BS. A K-nearest neighbor classification mechanism was presented to regularly classify the continually raising new contents and store the content at BSs at appropriate clusters with low complexity and high accuracy.

Chen *et al.* [100] have studied the caching gain by learning the user request behavior. The authors determined the relationship between the popularity of content and user preferences and presented a probabilistic model to synthesize the user preferences by utilizing the popularity of content. The content offloading problem is formulated as integer programming to maximize the offloading probability for D2D networks. The user demand behavior is modeled using probabilistic latent semantic analysis based on the expectation-maximization mechanism to solve the user preference learning problem.

Caching the content at the user devices provides lower delay, minimal traffic, energy efficiency, and higher throughput. In [101], the authors present an efficient learning-based caching mechanism working collectively with a non-parametric estimator to minimize the delay in the D2D network. The non-parametric estimator is used to learn the intensity function of the content demands, and it is considered that the popularity information is not known in advance. Further, a caching mechanism is presented to decide the most suitable pairs that improve higher delay with higher throughput and minimal delay. This mechanism can be further extended to the more dynamic system where the parameters fluctuate very frequently.

2.6.3 Reinforcement Learning based Caching

The conventional optimization methods can not be adapted for intelligent caching decisions in cooperative setting, in view of dynamic content popularity and mobility of nodes. Recent success in reinforcement learning (RL) to solve complex control problems attracted

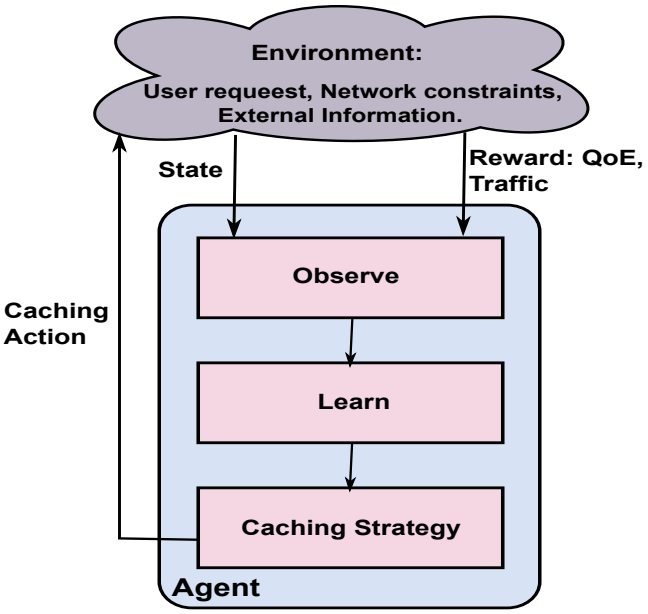


Figure 2.6: Reinforcement Learning Approach

the research community [31]. RL is one of the significant areas of machine learning shown in Fig. 2.6. RL is a learning process where agents observe the dynamic environment and adjust its policy to obtain an optimal strategy. However, getting the best strategy need knowledge of the entire system. Hence, it is not applicable to solve large scale networks. Deep learning (DL) is a well-known technique to address the RL limitation, and by combining the RL with DL emerged as deep reinforcement learning (DRL). Recently, wireless networks (Heterogeneous networks, unmanned aerial vehicle networks and IoT) turns out to be autonomous, decentralized and ad-hoc in nature. These stochastic and uncertain environments complexity grows as the size of the network grows. Therefore, DRL is an alternative solution.

Wang *et al.* [102] have presented a learning-based caching mechanism to maximize the number of users served by the nearby nodes. The authors first present a collaborative filtering mechanism to predict the popularity of the content, and then transfer learning is utilized to increase the accuracy. Further, a distributed iterative algorithm is proposed to maximize the number of users served by the nearby BSs by considering the interactions between users and BSs. Hou *et al.* [103] have presented a caching problem in vehicular networks by considering the mobility of vehicles and storage of the RSU to minimize the

latency. The problem is modeled as a Markov decision process and designed a Q-learning based solution by utilizing the user mobility prediction based on the LSTM network. Further, the authors presented an optimal cache strategy based on a greedy heuristic.

In [104], the authors present an edge caching scheme to maximize the content offloading ratio in hierarchical wireless networks through D2D communications. In proactive caching, the most popular content is cached at the devices to reduce delay and traffic. However, the content popularity is not valid, and it does not exhibit user preferences since the popularity is calculated based on the historical user request information collected within a particular time. In contrast, the user preferences exhibit the probability of individual content demanded by each user. The authors consider the users' social relationship, mobility, and system learning to maximize the offloading ratio in this problem. The caching problem is formulated as an Integer programming to maximize the offloading ratio by analyzing the social relationship and mobility, and it is proved as NP-hard. Further, the proposed problem is modeled as the Markov decision process and presented a Q-learning-based distributed cache replacement scheme.

There are various single-agent learning algorithms like DQN, DDPG, and advantage actor-critic (A2C) in the literature [105]. A Q-learning based cache update mechanism presented to model the local and global content popularities as Markov chains in [106]. In [107], the authors presented the proactive caching mechanism based on policy gradient reinforcement learning schemes to minimize the long-term average energy cost by assuming the Poisson shot noise popularity dynamics. A DQN approach is presented in [108] to address the large continuous state-action space. In [109], the authors formulated the cache replacement problem as MDP to minimize the long term reward of fetching transient data item and presented a caching policy based on the A3C (asynchronous advantage actor-critic) DRL mechanism. A DRL based framework with Wolpertinger architecture is presented in [32] for content caching in a single BS scenario and presented a deep deterministic policy gradient training mechanism for the actor-critic network. However, the works mentioned above consider the DRL mechanism for caching content, which is not a practical solution to the distributed environments where multiple agents are involved in decision making. In contrast, Chapter 6 considers the multi-agent DRL to handle the dis-

tributed nature of the problem.

In reality, the environment is complex and there are several cases where the single-agent can not deal effectively. In this cases multiple agent systems are essential. In the multi agent scenario all the agents learn the policy by interacting within a common environment. Therefore, an agent must either compete or coordinate with other agents in the environment to obtain the good results. The cooperative cache network can improve performance [11]. Conventional single agent reinforcement learning mechanisms such as Q-learning or policy gradient is not applicable in multi agent reinforcement learning because as the training progresses each agent policy changes and environment becomes non stationary. Jiang *et al.* [110] have formulated the content caching in D2D networks as multi-armed bandit problem to minimize the delay and presented two Q-learning based multi agent learning mechanisms. Q-learning based multi agent learning mechanism maintains the Q-value in memory because the massive state-action space storage of individually BS may exhaust. Zhong *et al.* [32] have presented a actor-critic framework with deep deterministic policy gradient learning scheme to reduce the overall delay. The content sharing is not considered by Zhong *et al.* and this leads to cache under utilization. A multi-agent RL mechanism is proposed in [111] to minimize the traffic congestion in the multi-intersection scenario. Song *et al.* [112] have investigated the joint content caching and content sharing in cooperative scenario and addressed the problem in view of multi-armed bandit learning by designing an ADMM. However, the works mentioned above have considered the multi-agent DRL mechanism where they suffer from exchanging information hugely between MECs. As the number of contents and MECs rising leading to huge exchange overhead, distributed cooperative caching has been considered with a recurrent multi-agent DRL mechanism with long short term memory in Chapter 6.

2.7 Proactive and Reactive Caching

Caching can be classified into proactive and reactive based on fetching content from the content server. In proactive caching, the content is cached at the bases stations based on the popularity distribution of the content predicted from the historical user request pat-

tern or known earlier. The caching problem has been studied extensively in the literature. Shanmugam *et al.* [24], have discussed the way to minimize expected download delay by assigning files to the helper (femtocells). They proposed a greedy approach with the factor of 2-approximation for the NP-complete problem. Wu *et al.* [113] have presented the data dissemination issue to guarantee the QoS while caching on the edge. Applegate *et al.* [114] have presented an intelligent content placement algorithm for large-scale library sizes by modeling the content placement problem as a mixed-integer program problem with popularity, link bandwidth, and capacity constraints. The authors presented a Lagrangian-based relaxation and rounding mechanism, and they proposed frequency of placement updates, popularity fluctuation, content updates and content popularity estimation to address the practical issues. Cao *et al.* [115] have briefly introduced mobile edge computing, its applications and also reviewed machine learning-based intelligent data offloading approaches in MEC.

ElBamby *et al.* [14] have presented a cache placement problem to minimize the delay in a small cell network. The authors first decomposed the problem into two parts. First, similar users are grouped using the clustering mechanism. Second, a clustering-based placement strategy was developed by estimating the popularity of content among the users in the group using reinforcement learning. In [93], the authors studied the data offloading problem in wireless networks by utilizing the social and spatial structure of the network. The content is proactively placed at each node based on the correlation among the users and content popularity. Then the authors presented a caching mechanism by predicting the influential users in the network by utilizing the social correlation among the users. Blaszczyszyn *et al.* [116] have proposed an optimal content placement policy to maximize the hitting probability.

Qiao *et al.* [117] have proposed caching based mmWave framework to minimize the retrieval and connection delays in fifth generation cellular networks. The authors designed a cache management scheme and achieved optimal video streaming quality by formulating the problem as MDP. A cell-by-cell decomposition mechanism is designed to practically solve the MDP problem with dynamic programming by reducing the state space. Tadrous *et al.* [118] have investigated a proactive caching problem for delay-sensitive applications to

minimize the service cost. The authors learn the basic bounds for the caching strategy with minimum possible cost and designed caching mechanism for fluctuating and uniform user request pattern. Hou *et al.* [103] have presented a caching problem in vehicular networks by considering the mobility of vehicles and storage of the RSU to minimize the latency. The problem is modeled as a Markov decision process and a Q-learning based solution is designed by utilizing the user mobility prediction based on the LSTM network. Further, the authors presented an optimal cache strategy based on a greedy heuristic.

Shen *et al.* [119] have presented an incentive caching scheme in SCN with one mobile network operator and several content providers. Content providers aim to maximize profits by determining the number of contents cache at SBS to improve user QoS. The authors modeled the problem as a Stackelberg game with the content providers as the followers and mobile network operators as leaders. A non-cooperative game is modeled for followers and proves its Nash equilibrium. Tong *et al.* [120] have investigated the optimal caching mechanism of scalable video coding streaming in SCN by considering the video scalability and channel diversity. The problem is modeled as ILP to maximize the average quality of scalable video coding streaming with cache capacity constraints. Further authors presented a low-complexity caching mechanism based on dynamic programming by simplifying the caching of scalable video coding as a knapsack problem, and it shows that the proposed caching mechanism caches the video based on the video popularity. Li *et al.* [121] have investigated the big data offloading from the cloud server to the mobile users and propose a three-layer edge computing framework in industrial mobile wireless networks. In this study, the authors considered the user mobility path, sojourn time, and the mobile node's capacity to offload the huge data through mobile networks and also presented a Hungarian algorithm to solve the data fetching problem. Elsayed *et al.* [122] have designed a caching mechanism to improve the QoS of the vehicular user with the uniform social pattern. In this work, the authors specifically considered the users who have predictive behavior regarding time and type of access to social media platforms. Kumar *et al.* [123] have presented a radio access network-aware adaptive video caching mechanism (RAVEN) to maximize the hit ratio. The cache placement problem is formulated as Integer linear programming problem and presented the RAVEN caching scheme by utilizing the predicted video request

bit rate and video popularity information for caching decisions. To address the problem of fetching the content from the distant cloud servers, the authors used the parked vehicles as the caching nodes to store the appropriate content, which will be requested by the users crossing the parked vehicles. Further, the authors present the greedy caching scheme to choose the appropriate road segments for content caching.

The works mentioned above have not considered heterogeneous user preferences, user contextual information, user mobility, and the randomness of contact duration in proactive caching. Hence, in this thesis, proactive caching schemes have been designed by considering the popularity prediction, user preference learning, and mobility in Chapters 3, 4 and 5.

Practically, the content popularity is time-varying, so the above assumption (known in advance) makes it less practical. In contrast, popularity prediction requires user association, and further user preferences may vary in different contexts, such as personal information, topology, location, etc [16]. For taking the caching decision, futuristic content popularity information may not be available. In the real world, the limited cache size restricts the mobile edge caching performance [28]. A simple solution is to devise efficient content placement mechanisms by considering user preferences and content popularity [22]. Effective cache utilization is reduced when the individual nodes with limited storage make independent decisions since they may redundantly cache popular content. A practical solution is to facilitate cooperation among edge nodes by sharing the content. Different edge nodes share their content in cooperative caching, forming more extensive cache storage and enabling cache diversity [11]. Generally, the caching decisions of various nodes depend on each other, but each edge node is aware of its own caching decision and unaware of the other nodes decisions.

Li *et al.* [124] presented a survey on content placement and delivery mechanisms in various cellular networks. Content pre-fetching that depends on content popularity has been investigated in the literature. Proactive caching has been studied in [24, 65, 114, 78, 125]. Collaborative cache placement has been investigated in SCN to handle the limitation of cache capacity at each node [11, 126, 30, 127]. The works mentioned above consider the content popularity known in advance. Moreover, popularity prediction based caching strategies were also studied in [14, 128, 129, 15, 130, 131]. However, the works mentioned

above consider the content popularity prediction and the dynamic user requests and environment complexities are not considered.

Yuan *et al.* [132] have discussed proactive and reactive caching in the D2D communication aspect. In proactive caching, the prefetched content is cached at edge nodes, whereas in reactive caching, the intermediate nodes decide whether to cache or drop the content to the neighboring node, which comes into the user's proximity. The authors proposed ProRec, a unified framework to cache content, considering the proactive and reactive caching to maximize the hit rate. A Lagrangian multiplier scheme has been proposed to find the optimal content caching. Further, a greedy approximation algorithm has been presented. Hou *et al.* [133] have discussed the resource allocation scheme for backhaul links to minimize the average downloading delay and proposed the access and backhaul resources allocation algorithms. Zhang *et at.* [134] have studied the collaborative task offloading and content caching models to reduce the overall latency of mobile device. Further, an effective Lyapunov online mechanism is presented to perform joint dynamic data caching and task offloading mechanisms.

Typically, wireless caching has a high time-varying user requests. To address user requests' time-varying nature, BS with finite cache capacity replaces the content very often. Commonly used content replacement mechanisms are least frequently used (LFU), least recently used (LRU) and first-in first-out (FIFO) [47]. The traditional replacement models cannot capture the changing nature of content popularity because of the real environment's complexity. The conventional replacement mechanisms are suitable for single cache replacement. However, multiple cache replacement mechanisms require coordination among the nodes.

Jiang *et al.* [110] have formulated content caching in D2D networks as multi-armed bandit problem and presented two Q-learning based multi-agent learning mechanisms. Q-learning based multi-agent learning mechanism maintains the Q-value in memory because individual BS's massive state-action space storage may exhaust. Zhong *et al.* [32] have presented an actor-critic framework with a deep deterministic policy gradient learning scheme to reduce the overall delay. Content sharing is not considered by Zhong *et al.*, and this leads to cache underutilization. A multi-agent RL mechanism is proposed in [111] to minimize

the traffic congestion in the multi-intersection scenario. Song *et al.* [112] have investigated the joint content caching and content sharing in the cooperative scenario and addressed the problem in view of multi-armed bandit learning by designing an ADMM. However, the works mentioned above have considered the multi-agent DRL mechanism where they suffer from exchanging information hugely between MECs. As the number of contents and MECs rising leading to huge exchange overhead, the distributed cooperative caching has been considered with a recurrent multi-agent DRL mechanism in Chapter 6.

2.8 User Preference and Prediction based Caching

Popularity prediction allows the caching mechanism to make an accurate decision to choose appropriate content in the network. Practically, a few popular contents serve a wide variety of network traffic, whereas other contents are requested rarely (for example, 1% of Facebook videos account for 83% of total watch time [135]). The content popularity information is time-varying and unaware of in advance. To improve the user QoE the proactive caching approaches relied on the popularity of the content. Thus, content popularity distribution prediction is needed, and the prediction algorithm should be accurate, quick, and scalable.

Popularity prediction based caching strategies were studied in [14, 128, 15, 130, 131]. ElBamby *et al.* [14] have presented a cache placement problem to minimize the delay in a small cell network. The authors first decompose the problem into two parts. First, similar users are grouped using the clustering mechanism. Second, a clustering-based placement strategy was developed by estimating the popularity of content among the users in the group using reinforcement learning. Bharath *et al.* [128] have presented a transfer learning mechanism to predict the popularity profile using the user request pattern for distributed heterogeneous cellular networks. Muller *et al.* [129] have presented a context-aware proactive caching mechanism by predicting the content popularity. Chen *et al.* [15] have presented an echo state network to estimate content popularity and mobility of nodes to maximize the user QoE in unmanned areal vehicle placement. In [131], Garg *et al.* have investigated online prediction and online learning mechanisms for content caching in the cellular network.

Li *et al.* [136] have presented a popularity-based content caching mechanism by predicting the future request pattern of a TV program utilizing the demand pattern of the TV program. The content popularity has been computed using neural networks. In [137], the authors have discussed a popularity-based content replacement mechanism. The content popularity has been learned online, which is more responsive to constantly change content popularities because of no training phase. The proposed mechanism learns the popularity of the content with the help of access pattern similarities of different contents. Abdelkrim *et al.* [138] have presented a hybrid regression-based prediction model for user-generated videos. The popularity prediction model dynamically adapts the popularity of content by considering the end-user watch time and the number of shares. Further, a cache replacement model was designed by utilizing the prediction model to decide on content eviction.

Tanzil *et al.* [139] have presented an adaptive caching mechanism to improve the user QoE. The problem is formulated as a mixed-integer linear programming problem to minimize the download delay. The adaptive caching mechanism involves the content popularity prediction to select the appropriate node and size of the cache. The extreme learning machine [140] neural network has been utilized to predict the content with the help of request statistics from users, content features, and user behavior. Hou *et al.* [141] have presented a proactive caching scheme to improve the user QoE by predicting the content popularity. In [141], the solution to the proposed problem is given in two phases. In the first phase, the content popularity has been predicted using the transfer learning technique. In the second phase, a greedy algorithm has been proposed to solve the proposed problem.

In [142], authors have devised a cooperative caching mechanism to minimize the transmission delay by considering limited cache size and bandwidth constraints for mobile networks. In clustering-based caching, the BSs are clustered into different groups. Chen *et al.* [18] have presented a cooperative caching mecahnism to balance the content diversity and transmission in cluster centric cellular network. In [18], first, the cache storage has been divided into two parts, one part stores the most popular content, and the second part caches less popular content cooperatively. In [130], authors have proposed a learning theoretic perspective for content caching heterogeneous networks with time-varying and unknown popularity profiles.

Proactively caching the predicted content at edge nodes reduces the network's latency, congestion, and traffic as the appropriate content is determined. The nodes' next location can be estimated using the user mobility prediction to cache the content. Abani *et al.* [143] have presented a mobility prediction based caching scheme. The user mobility prediction uncertainty has been measured using entropy. Yao *et al.* [144] have presented a mobility prediction based cooperative caching mechanism for vehicular content-centric networks to store the most popular content at mobile users that frequently visit the hot spot areas. The probability of reaching the hot spot by a node has been predicted using partial matching. Further, a cache replacement mechanism was designed based on the predicted content popularity to improve the user QoE. Khelifi *et al.* [145] have presented a mobility prediction based caching mechanism to choose the RSU in the user moving direction to retrieve the content from the RSU. The user mobility has been predicted using the LSTM network. The work mentioned above has not considered the user contextual information to make caching decisions. Therefore, in Chapter 3, content popularity distribution prediction using a machine learning algorithm has been designed to capture user interests efficiently.

Content popularity indicates the average interest of multiple users but not exhibits the individual user preferences [19]. Most of the existing literature considers that all the users have the same content distribution (homogeneous popularity). However, various users have diverse preferences. The assumption made on homogeneous popularity ignores the users' preferences and results in losing valuable information. Less than 20% of users generate 80% of traffic, which shows that the users' activity level is heterogeneous [20]. In the literature, most proactive caching approaches ignored user behaviour, such as heterogeneous user preferences and activity levels, introducing new challenges into mobile edge networks. Therefore, employing the individual user activity levels and preferences improves the cooperative caching strategy design.

User preferences play a crucial role in proactive caching. The user preference prediction is broadly studied in recommender systems [96, 97], where most works consider collaborative filtering. In [98], the authors studied the caching mechanism to enhance the user QoE by combining the caching decisions with recommender systems. Including a recommender system enhances the caching performance by caching the appropriate content

at the base stations by predicting the individual user preferences using the collaborative filtering mechanism. In this scheme, each user preference is figured out using collaborative filtering, then the content that attracts more users is ranked and the content is cached at appropriate BSs. The content caching problem is formulated to maximize the cache hit ratio by considering the user preferences and cache capacity. Further, the authors presented a low-complexity heuristic caching approach to address the proposed problem effectively.

User preferences based caching mechanisms were studied in [146, 147, 148, 100, 19]. Bastug *et al.* [146] have presented a local content popularity based caching mechanism for small cell networks. Liu *et al.* [147] have investigated a CPP in Fog-RANs (radio access network) by considering user preferences and physical layer transmission. The cache placement problem is formulated as an optimization problem to minimize the download delay with capacity constraints and presented distributed and centralized caching policies. In a centralized caching scheme, the CPP has been reformulated into a submodular optimization problem and presented as an approximation algorithm based on a greedy strategy to give at least $\frac{1}{2}$ approximation ratio. In the distributed caching scheme, a belief propagation-based mechanism has been presented to give a sub-optimal solution. In [148], authors have investigated proactive caching schemes in wireless networks by considering spatial locality, activity level, and user preferences. The authors presented a framework to optimize caching schemes in D2D networks with spatial locality, heterogeneous activity level, and user preferences. The caching problem is formulated as maximization of success probability and minimization of average user rate. A synthesized user preference mechanism is presented by utilizing the user activity level and preferences.

Chen *et al.* [100] have studied the benefit by utilizing the association between the popularity of content and user preferences and presented a method to synthesize the user preferences. The cache placement problem has been formulated as an optimization problem to maximize the content offloading ratio from D2D networks with user activity level and preferences. To solve the proposed problem, the authors first learn the user preferences by modelling the user demand pattern by utilizing the probabilistic latent semantic analysis. Next, the model parameters are learned by the EM (expectation-maximization) algorithm. Further, a low-complexity greedy mechanism has been presented to achieve at least $\frac{1}{2}$ ap-

proximation ratio. Zhang *et al.* [149] have investigated the online content caching mechanism in single cache node scenario where the content popularity is not known in advance. The content popularity has been predicted using a novel grouped linear model based on the historical user data. Further, a model-free RL mechanism has been presented to replace the content at the base station to enhance the learning process in dynamic environments.

Jiang *et al.* [150] have studied the caching mechanism to find optimal strategy in fog radio access networks. The caching problem has been formulated as an optimization problem to maximize the cache hit ratio. The authors have proposed two caching architectures and a caching strategy by predicting the popularity of the content and learning the user preferences. The online popularity prediction algorithm uses the user preferences and content features, and the offline user preference learning mechanism uses the online gradient descent technique and follows the regularized leader technique. The caching mechanism presented predicts the content popularity with low computational complexity and tracks the popularity with temporal and spatial popularity without considering delay. Further, two learning-based caching mechanisms have been presented, and the upper bound of the popularity prediction error, lower bound of hit ratio, and regret bound of overall hit ratio of the caching strategy proven theoretically. Lee *et al.* [19] have studied the statistical modeling of individual user preferences to effectively identify the individual user preferences of video content. The authors proposed a novel modeling framework by characterizing the essential features and parameters of genre-based historical data.

The work in this thesis considers the user preference prediction using a machine learning model to perceive the dynamic nature of content popularity which has not been given adequate attention in the existing algorithms. As the number of MEC nodes and contents rise, this leads to huge exchange overhead within the network. Therefore, a heterogeneous user preference-based caching scheme has been proposed utilizing the heterogeneous user activity levels and user preference prediction in Chapter 4.

2.9 Summary

In this chapter, mobile edge network architecture and the advantages of MEN are discussed. Moreover, mobile edge caching and different cache deployment scenarios in edge caching are discussed. A survey on different mobile edge caching schemes has been presented, such as cooperative, mobility and coding based caching schemes. An exhaustive survey on learning-based caching, proactive and reactive caching is performed. Further, user preference and prediction based caching in MEN has been presented. In this thesis, proactive and reactive caching approaches have been designed for mobile edge networks. Further, the work presented in this thesis has been compared with the existing caching mechanism in the literature. The next chapter presents deadline-aware content caching using an echo state network integrated with fuzzy logic to improve the cache hit and acceleration ratios.

Chapter 3

Deadline-aware Content Cache Placement using Echo State Network Integrated Fuzzy Logic for Mobile Edge Networks

The cooperative caching mechanism has a better cache hit ratio than non-cooperative caching and improves the user quality of service [55]. Hence, deciding which content to cache at which MEC cooperatively by utilizing limited cache capacity in MENs is challenging. However, time-critical and delay-sensitive applications like video streaming, Internet of Things (IoT) and financial applications need a response within a *deadline* (i.e., a specific time limit) [12]. The deadline determines the maximum allowable response time [13]. Some applications like healthcare demand the guarantee of timeliness strictly (hard deadline), whereas some IoT applications may tolerate the delay (soft deadline) [13]. If a request is not served within the deadline, the quality of service would be affected, and this in turn affects user QoE. Hence, to improve the user QoE, the request deadlines must be satisfied. Therefore, caching decisions by considering the limited cache capacity of BS and content deadline is a important task that is the focus of this research work.

In this chapter, content placement mechanism has been proposed using the fuzzy logic

to maximize the saved delay in wireless networks. The novelty of the approach lies in designing a caching mechanism for mobile edge networks (MEN) by considering limited storage at base stations, the deadline of content request and popularity prediction. Initially, the cache placement problem is formulated as an integer linear programming (ILP) problem. The solution is designed as relaxation-and-rounding based on the rounding technique. Further, a fuzzy logic based caching algorithm has been proposed by considering deadline, the benefit of caching content and content request distribution prediction for content placement decisions. Moreover, an Echo State Network (ESN) based prediction mechanism has been designed to predict the content request distribution for mobile edge network.

The contributions of this chapter are as follows:

- Formulate a content placement problem (CPP) as an integer linear programming problem in mobile edge networks with an objective to maximize the saved download delay subject to cache capacity, request deadlines and popularity of the content.
- Design an approximation algorithm based on the relaxation and rounding technique to solve the integer linear programming version of content placement problem.
- Propose a fuzzy logic-based caching algorithm (FCA) to find the near-optimal solution by considering content request distribution, deadline of the content and benefit (distance) of caching content. A content request distribution prediction mechanism is designed using echo state network.
- Simulation results show the efficacy of the proposed algorithm in terms of acceleration ratio, cache hit ratio and number of files satisfying deadlines.

The rest of the chapter is organized as follows. In section 3.1, system model and formulation of the content placement problem has been discussed. An approximation algorithm for the proposed problem is presented in section 3.2. Content distribution prediction using the echo state network is presented in section 3.3.1. Fuzzy inference system is discussed in section 3.3.2, a fuzzy logic based caching algorithm is presented in section 3.3.3 and a replacement strategy is discussed in section 3.3.4. Simulation environment and results are presented in section 3.4. A summary of this chapter is mentioned in section 3.5.

3.1 Mobile Edge Computing (MEC) Model and Problem Formulation

In this section, a base station integrated MEC network model is presented. Further, problem formulation is presented in detail.

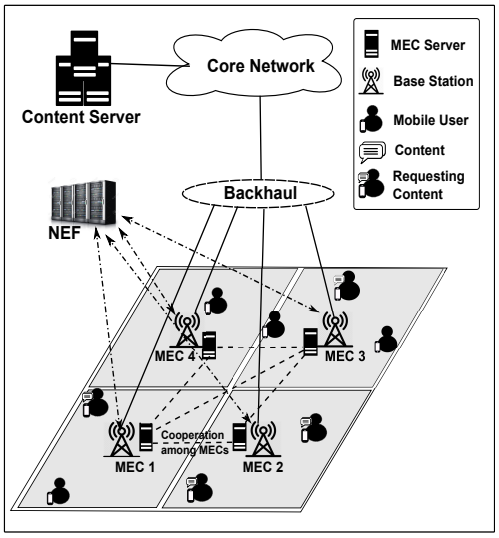


Figure 3.1: Illustration of system model

Mobile edge computing improves users' capabilities by providing cache capacity (storage), network resources and computing in close proximity to users. Consider mobile edge networks containing a set of users, set of MECs and a content server, as shown in Fig 3.1. In MENs, each MEC has computational capability, limited storage capacity and deployed with base stations. The storage of each MEC is used for content caching. The MECs are connected with each other and also to the core network through backhaul links. The content server acts as an origin server that stores all contents. Network Exposure Function (NEF) serves as a coordinator (it is a crucial network element in 5G networks) [151, 79]. NEF maintains the indexes of the content cached at individual MECs and also monitors the content requests by users at each MEC [151]. A user is directly connected to base station and the user may be in the communication range of more than one BS at any point in time. However, any user can communicate with only one MEC at a particular time. A MEC considered as a viable MEC for a user u if it is in the communication range of the user.

Mobile users are connected to the base stations according to a cellular network protocol.

The connected BSs are accountable for serving user requests. In this chapter, cooperative caching between MEC nodes has been considered. The proposed algorithm will run on the network evaluation function (NEF) [151] to compute and allocate contents to individual MEC to maximize overall saved delay of MEN. NEF provides the content statistics (indexes of cached content and request information of each content at MEC) and user context (as a central element). The working process is as follows: If the user requested content is stored at the corresponding MEC, then the MEC serves the user request. In case the content is not stored at the MEC, then the corresponding MEC will query the NEF for the requested content that is available in other MECs. Based on NEFs response, the requested MEC fetches the content from the nearby MEC. Otherwise, MEC uses the core network with the help of backhaul links to fetch the content from the content server.

Table 3.1: List of Notations

Term	**Definition**
$\mathcal{R}$	Set of base stations coupled with MEC servers
$\mathcal{F}$	Set of contents
$\mathcal{C}$	Set of content type
$\mathcal{U}$	Set of users
S_i	The cache capacity of i-th BS
B_f	The size of f-th content ($f \in \mathcal{F}$)
$\mathcal{T}_{cf}$	Deadline of content f of type c
r_f	Number of requests for content f
$p_i(f)$	Probability that content f is requested by users from BS i
d_{ui}	The delay for transmitting content from BS i to user u
d_{iI}	The delay for transmitting content to BS i from Internet I
x_{cf}^{i}	Binary variable indicating that content f of type c is exist in BS i
y_{cf}^{ui}	Binary variable indicating that user u fetches content f of type c from BS i

In the system model, the set of regions (MEC with BS) is denoted by $\mathcal{R} = \{1, 2, \cdots, i, \cdots, R\}$. The set of users in the network is indicated as $\mathcal{U} = \{1, 2, \cdots, u, \cdots, U\}$. The library contains $\mathcal{C}$ different types of contents and $\mathcal{F} = \{1, 2, \cdots, f, \cdots, F\}$ contents in each content type. Let B_f denote the size of f-th content ($f \in \mathcal{F}$). Every content in each category is

intended to serve within the maximum allowable response time. This response time is chosen as a deadline. The deadline can range from nearest (small) deadline to longest (large) deadline. $\mathcal{T}_{cf}$ represents the deadline of the file f in the category c. The files with nearest deadline are served first. Capacity of i-th BS is denoted as S_i. The average downloading time per information bit from BS i to user u is denoted as d_{ui} and d_{uI} denotes the downloading time per information bit from I (Internet) to user u. The download delay from a base station with in region d_{ui} is less than that from the other region j BS (d_{uj}). Similarly, the downloading delay from other region d_{uj} is less than that from Internet d_{uI} (i.e., $d_{ui} < d_{uj} < d_{uI}, \forall j \neq i$). List of notations used in this chapter are presented in Table 3.1.

3.1.1 Popularity of Content and Content Types

Different content types may have different popularities. The content popularity of various regions can be different from each other. Users may have significant preferences for specific content types, which motivates this study to identify users' content preferences in a region in terms of their preferred content.

Probability that the user u requests content of a specific type c for all available content types is represented as $p(c|u)$. The probability that content belongs to a content type c is requested by the user in a region i is $p_i(c)$, which can be represented as

$$p_i(c) = \frac{1}{U_i} \sum_{u=1}^{U_i} p(u) \cdot p(c|u) \tag{3.1}$$

Where U_i is the number of users in the region i and $p(u)$ is the probability that user u generates a request.

Given the overall popularity distribution of content and type of each content, identify the content popularity distribution within each content type. Let $p(f)$ denotes the overall probability of content f overall contents and $p_c(f)$ denoted as the overall probability of

content f overall content types c.

$$p_c(f) = \begin{cases} p(f), & \text{if } f \in c \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

The popularity of content f with in content type c can be expressed as

$$p_f^c = \frac{p_c(f)}{\sum_{f=1}^{F} p_c(f)} \tag{3.3}$$

Knowing the probability of request of different content types in a region and the popularity of content in each content type, derive the probability that content f is requested by users in the region i as $p_i(f)$.

$$p_i(f) = \sum_{c=1}^{C} p_f^c \cdot p_i(c) \tag{3.4}$$

3.1.2 Cache Decision Variables

Two decision variables have been defined, namely content placement decision variable and content fetch decision variable to show the cooperative caching. *Content placement decision variable* determines where to cache which content.

$$x_{cf}^i = \begin{cases} 1, & \text{content } f \text{ of type } c \text{ is placed at BS } i \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

Content fetch decision variable determines which base station should server the request.

$$y_{cf}^{ui} = \begin{cases} 1, & \text{user } u \text{ will fetch content } f \text{ of type } c \\ & \text{from BS } i \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

3.1.3 Delay

Consider a user *u* and its viable BSs are regarded as neighboring base station $\mathcal{N}(u)$. The size of neighborhood is $\mid \mathcal{N}(u) \mid$. Let i_u denotes the region index with the i^{th} smallest downloading delay to user u. x_{cf}^{i} is a decision variable indicating that the content f of type c exists on cache of BS i. The average delay per information bit for user u is denoted by D_u. D_u can be written as

$$\begin{aligned} D_u = & \sum_{i=1}^{|\mathcal{N}(u)|} d_{u,i_u} \sum_{c=1}^{C} \sum_{f=1}^{F} \left[\prod_{h=1}^{i-1} (1 - x_{cf}^{h_u}) \right] x_{cf}^{i_u} \cdot p_i(f) \\ & + d_{uI} \sum_{c=1}^{C} \sum_{f=1}^{F} \left[\prod_{h=1}^{|\mathcal{N}(u)|-1} (1 - x_{cf}^{h_u}) \right] x_{cf}^{i} \cdot p_i(f) \end{aligned} \tag{3.7}$$

Where, $\left[\prod_{h=1}^{i-1} (1 - x_{cf}^{h_u}) \right] x_{cf}^{i_u}$ is the indicator function which is defined over decision matrix X, which means the content f of type c is in the cache of the region i_u and it is not in any of the regions with delay lower than h_u, for $h = \{1, 2, \cdots, i-1\}$. Furthermore, $\left[\prod_{h=1}^{|\mathcal{N}(u)|-1} (1 - x_{cf}^{h_u}) \right] x_{cf}^{i}$ is an indicator function for the condition that content f of type c does not exist on any region.

3.1.4 Deadline

Definition 3.1.1 (Deadline)**.** *It is defined as the maximum allowable time for the response to a requested content.*

The requested content needs a response within the deadline, which describes the maximum allowable time for response. Hence, the content with specific deadline $\mathcal{T}_{cf}$ gives deadline constraint.

$$D_u \leq \mathcal{T}_{cf}, \qquad \forall f \in \mathcal{F}, c \in \mathcal{C}, u \in \mathcal{U} \tag{3.8}$$

where, $\mathcal{T}_{cf}$ is the deadline of the content f of type c. This shows that the delay to download content is less than the given deadline.

Definition 3.1.2 (Saved delay)**.** *The saved delay is defined as the difference between the*

download delay from the Internet and base station.

3.1.5 Problem Formulation

The aim of this chapter is to maximize the saved delay by placing content in the BSs. Thus the maximization problem is modeled as a multi-commodity facility location problem subject to caching capacity and deadline constraints.

Therefore, the formulation becomes:

$$\text{Max} \sum_{u=1}^{U}(I - D_u) \tag{3.9}$$

s. t.

$$\sum_{i=1}^{R} y_{cf}^{ui} = 1, \qquad \forall f\in\mathcal{F}, c\in\mathcal{C}, u\in\mathcal{U} \tag{3.10}$$

$$y_{cf}^{ui} \leq x_{cf}^{i}, \qquad \forall f\in\mathcal{F}, c\in\mathcal{C}, i\in\mathcal{R}, u\in\mathcal{U} \tag{3.11}$$

$$\sum_{c=1}^{C}\sum_{f=1}^{F} B_f \cdot x_{cf}^{i} \leq S_i, \qquad \forall i\in\mathcal{R} \tag{3.12}$$

$$D_u \leq \mathcal{T}_{cf}, \qquad \forall f\in\mathcal{F}, c\in\mathcal{C}, u\in\mathcal{U} \tag{3.13}$$

$$y_{cf}^{ui}, x_{cf}^{i} \in \{0, 1\}, \qquad \forall f\in\mathcal{F}, c\in\mathcal{C}, i\in\mathcal{R} \tag{3.14}$$

The objective (3.9) is the total saved delay caused by users of the overall network. Constraint (3.10) guarantees that each request from a user can obtain content from only one base station. Constraint (3.11) represents the availability constraint, which ensures that content can be fetched from a cache if and only if it is stored in the cache. Constraint (3.12) provides the finite capacity of each BS. Constraint (3.13) is the deadline constraint, which ensures that the maximum allowable delay for the response to a request. Thus, the BS can satisfy the users' QoS requirements. Finally, constraint (3.14) is the non-negativity and integrality of the decision variables.

Theorem 3.1.1. *The content placement problem in equation (3.9) is NP-hard.*

Proof. To show the problem presented in equation (3.9) as an NP-hard, transform the

known NP-hard problem to this problem. Knapsack problem has been considered, which is already NP-hard. *Knapsack problem:* Given a knapsack j of capacity S_j and n items $I = \{i_1, i_2, \cdots, i_n\}$, each with its own weight $\{w_1, w_2, \cdots, w_n\}$ and value $\{a_1, a_2, \cdots, a_n\}$. The objective of knapsack problem is to select the number of each item to add in a knapsack j such that the objective is to maximize the total value $P(i_k)$, i.e., $P(i_k) = \sum_{j=1}^{n} a_j x_j$ and the total weight must be less than or equal to the capacity of the knapsack i.e., $\sum_{j=1}^{n} a_j x_j \leq S_j$.

The problem in equation (3.9) is reduced to the knapsack problem as follows. Consider the number of contents $\mathcal{F} = \{1, 2, \cdots, f, \cdots, F\}$, the size of the content is considered as the weight B_f and the saved delay is considered as the value i.e., $P(i_k) = \sum_{u=1}^{U} D_u$. $x_{cf}^{j} = 1$ means that content f of type c is cached in j, otherwise 0. Thus, if problem in equation (3.9) can be solved in polynomial time, the 0-1 knapsack problem can be solved in polynomial time i.e., the 0-1 knapsack problem is polynomially time reducible to problem in equation (3.9). Since, NP-Hard problem is reducible to problem in equation (3.9), problem in equation (3.9) is NP-Hard. This completes the proof. □

3.2 Approximation Algorithm based on Relaxation and Rounding Technique

The relaxation and rounding algorithm (RAR) based on a relaxation technique is proposed to handle the problem in equation (3.9). The integer linear programming is handled by solving the relaxed fraction problem. The integer linear programming problem (0-1 binary variable) is relaxed that is x_{cf}^{i}, y_{cf}^{ui} to real numbers $\overline{x}_{cf}^{i}$, $\overline{y}_{cf}^{ui}$ extended between 0 to 1. The intuitive meaning of $\overline{x}_{cf}^{i}$ is that BS i can store a fraction of content and $\overline{y}_{cf}^{ui}$ is user can fetch part (fraction) of content. By relaxing the integer variable to non-negative integers, the Integer linear programming problem transformed into linear programming. Linear programming is known to be solvable in polynomial time [152]. The rounding technique is used to produce approximate solution for given problem as shown in Algorithm 3.1.

3.2.1 Relaxation

Convert the integer linear programming (ILP) to linear programming (LP) by relaxing the integer decision variables in equation (3.9) by introducing new variables $\overline{x}^i_{cf},\ \overline{y}^{ui}_{cf} \in [0,1]$. All the constraints exist in the given problem are linear equations. Therefore, optimal solution (fractional) can found in polynomial time represented as SD_r. Construct the optimal solution (approximate solution) by rounding the fractional solution obtained through relaxed version to integers. $\overline{x}^i_{cf}$ indicates where to cache content f of type c and $\overline{y}^{ui}_{cf}$ indicates the tendency where node i determine to obtain content f of type c.

$$\text{Max} \sum_{u=1}^{U}(I - D_u) \tag{3.15}$$

s. t.

$$\sum_{i=1}^{n} \overline{y}^{ui}_{cf} = 1, \qquad \forall f\in\mathcal{F}, c\in\mathcal{C}, u\in\mathcal{U}$$

$$\overline{y}^{ui}_{cf} \leq \overline{x}^{i}_{cf}, \qquad \forall f\in\mathcal{F}, c\in\mathcal{C}, i\in\mathcal{R}, u\in\mathcal{U}$$

$$\sum_{c=1}^{C}\sum_{f=1}^{F} B_f \cdot \overline{x}^{i}_{cf} \leq S_i, \qquad \forall i\in\mathcal{R}$$

$$D_u \leq \mathcal{T}_{cf}, \qquad \forall f\in\mathcal{F}, c\in\mathcal{C}, u\in\mathcal{U}$$

$$\overline{y}^{ui}_{cf}, \overline{x}^{i}_{cf} \in [0,1], \qquad \forall f\in\mathcal{F}, c\in\mathcal{C}, i\in\mathcal{R}$$

3.2.2 Rounding

The fractional optimal solution is obtained by solving the relaxed version of the problem presented in equation (3.15). The integral solution of the fractional optimal solutions is derived by rounding technique. Deterministic rounding algorithm [17] has been adapted by constructing the weighted bipartite graph for each content f of different content types c. A weighted bipartite graph $BP = (M, N, E, W(E))$ is constructed, where M and N are the nonadjacent nodes, E represents the edge set and $W(E)$ denoted as weight of the edge E. User nodes present in one side and BSs present on the other side of bipartite graph,

where *M* and *N* denote the user and base station respectively.

The user node set $M = \{m_1, m_2, \cdots, m_q\}$ is constructed by sorting the request rates in non-increasing order. Then, create the BS set $N = \{n_{i,v} | i = 1, \cdots, q, v = 1, \cdots, c_i\}$ based on $\overline{y}_{cf}^{ui}$, where $c_i = \lceil \sum_{u \in E} \overline{y}_{cf}^{ui} \rceil$. Edges of the graph BP is setup between M and N corresponding to the pair (u, i) such that $\overline{y}_{cf}^{ui} > 0$. For each positive i, if $c_i \leq 1$ then there exist only one edge $n_{i,1}$ in N add the edge $e_{u,i,1}$ to *E* and weight $w(e_{u,i,1}) = \overline{y}_{cf}^{ui}$ for each user *u*, $\overline{y}_{cf}^{ui} > 0$. Otherwise, multiple nodes present in *N* corresponding to BS *i*, find the minimum index i_1 such that $\sum_{u=1}^{u_1} \overline{y}_{cf}^{ui} \geq 1$. Let *E* contains the edges $(m_u, n_{i,1}), i = 1, \cdots, u_1 - 1$, for each of these edges $e_{u,i,1}$ set its weight as $w(e_{u,i,1}) = \overline{y}_{cf}^{ui}$. Moreover, add edge $e_{u_1,i,1}$ and its weight $w(e_{u,i,1}) = 1 - \sum_{u=1}^{u_1} w(e_{u,i,1})$. This provides that the sum of the components of $w(e)$ for each edge incident to $n_{i,1}$ is 1.

If $\sum_{u=1}^{u_1} \overline{y}_{cf}^{ui} > 1$ then the value of the $\overline{y}_{cf}^{ui}$ is not assigned completely, so create an edge $e_{u,i,2}$ and set its weight $w(e_{u,i,2}) = \sum_{u=1}^{u_1} \overline{y}_{cf}^{ui} - 1$. Then, construct an edge $n_{i,2}$ for client $u > u_1$, till total of one user assigned to $n_{i,2}$ and so on. The bipartite graph *BP* is constructed for every content *f* and processed according to saved delay in descending order. Then, the maximum weighted matching on *BP* is performed with capacity and deadline constraints. MT_E is the result obtained by matching, for each selected edge in MT_E, set y_{cf}^{ui} to 1, otherwise 0. Let the solution obtained from the rounding is represented as SD_{ra}

Algorithm 3.1 Relaxation-Rounding Algorithm

INPUT: $\{B_1, B_2, \cdots, B_F\}, \{S_1, S_2, \cdots, S_R\}, p_j(f), d_{u,i}$
where $u = \{1, 2, \cdots, U\}$, $i = \{1, 2, \cdots, R\}$, $c = \{1, 2, \cdots, C\}$ and $f = \{1, 2, \cdots, F\}$.
OUTPUT: y_{cf}^{ui}, x_{cf}^{i}.

1: Get the fractional solution $\overline{y}_{cf}^{ui}$, $\overline{x}_{cf}^{i}$ for the given integer decision variables by solving the relaxed version of the problem.
2: Get the integral solution y_{cf}^{ui}, x_{cf}^{i} by rounding the fractional solution.
3: Allocate the contents with suitable cache nodes.
4: **return** y_{cf}^{ui}, x_{cf}^{i}.

and SD^* is denoted as optimal integer solution obtained by equation (3.15). The lower bound of SD_{ra} is evaluated as follows.

Theorem 3.2.1. $SD_{ra} \geq \frac{1}{2} SD^*$.

Proof. The sum of *saved delay* obtained by placing the content *f* at the base stations in the

mobile edge networks is $SD_r = \sum_{\forall f \in \mathcal{F}} SD_f$. The sum of the saved delay obtained by integer rounding solution SD_{ra} is expressed as $SD_{ra} = \sum_{\forall f \in \mathcal{F}} SD_f^{ILP}$, where $SD_f^{ILP} = 0$ represents that content f is not placed in the integer rounding scheme SD_{ra}. For the integer solution SD_{ra}, a complementary solution $\overline{SD_{ra}}$ is constructed according to [17] i.e., for every edge in $E - MT_E$, set y_{cf}^{ui} to 1. Then, $\overline{SD_{ra}} = \sum_{\forall f \in \mathcal{F}} SD_f^{cmp}$. In the process of rounding some entries of y_{cf}^{ui} becomes zero, due to this some user requests may not be satisfied with the saved delay $\overline{SD_{ra}}$. In order to satisfy these requests, make (i) fractional value to an integer. (ii) the fractional value is considered if request is not met at all, otherwise, the difference between fractional and integral saved delay is considered.

$$SD_f^{cmp} = \begin{cases} SD_f, & \text{if } SD_f^{ILP} = 0, \\ 0, & \text{if } SD_f^{ILP} \geq SD_f, \\ SD_f - SD_f^{ILP}, & \text{if } SD_f^{ILP} < SD_f, \end{cases}$$

As per [17], rounding result satisfies $SD_{ra} + \overline{SD_{ra}} \geq SD_r$. It is easily understood that $SD_{ra} \geq \overline{SD_{ra}}$. Hence, $SD_{ra} \geq \frac{1}{2} SD_r \geq \frac{1}{2} SD^*$. □

The proposed relaxation and rounding technique solves the integer linear programming problem presented in equation (3.9) in polynomial time. However, the relaxation and rounding algorithm achieves a polynomial time complexity, and the complexity grows remarkably with an increase in the number of contents. For real scenarios as the scale continues to increase (large scale problems), the relaxation and rounding mechanisms are not efficient enough [75]. To address the system with a large number of nodes, contents and to ease the computational complexity, a heuristic algorithm has been designed based on the fuzzy logic in the next section.

3.3 Fuzzy Caching Algorithm based on Content Request Prediction

The content placement problem presented in equation (3.9) is to maximize the saved delay, since the popularity is determined by content request prediction (i.e., the appropriate content to be cached at each base station cooperatively requires the content popularity prediction). To address this issue, a fuzzy logic based cooperative content placement algorithm has been presented using content popularity prediction. The proposed algorithm runs on network evaluation function (NEF) [151] to compute and allocate contents to individual MEC to maximize saved delay overall MEN. The content statistics (indexes of cached content and request information of each content at MEC) and user context are provided by NEF (as a central element).

3.3.1 Popularity Prediction using Echo State Networks

A machine learning model *echo state network* [153, 15] (ESN) has been adopted to predict the content request distribution. ESN is one of the emerging recurrent neural networks with dynamic reservoir, which predicts the information and track the previous states of the network [153]. ESN is extensively used in time series prediction and dynamic system modeling because of the time-varying characteristics of the dynamic system.

ESN model is adopted to predict the content request distribution by considering the state of user content requests observed by NEF. ESN predicts the content request distribution by establishing the relationship between the requested content and user context (user information). ESN trains the neurons using simple linear regression and it has fast convergence speed. The ESN comprises four modules: a) input b) output c) agent and d) model.

- *Input*: The context of user u (which includes content request time, week, occupation, gender and age) at time t is taken as input vector $q_u^t = [q_{u1}^t, q_{u2}^t, \cdots, q_{uk}^t]$. The output vector h_u^t (content request distribution) is determined by the q_u^t, where k is the number of properties that comprise the user u context information. For example, the type of content like TV series, movies, videos which young age people or students

interested is different from the old age people or household. In reality, the content request preference affected by the user information and various demographics.

- *Output*: The output of user u at time t is represented as a vector of values $h_u^t = [h_{u1}^t, h_{u2}^t, \cdots, h_{um}^t]$. Where h_{um}^t is the output value of request m at time t.

- *Agent*: The agents in this model are the base stations. Each BS predicts for one user at time. So, the base stations execute U algorithms in every time slot.

- *Model*: The relationship between user information (input q_u^t) and request distribution (output h_u^t) is constructed by model. Model in ESN approximates function between input and output, which is a dynamic reservoir. The reservoir contains the input weight matrix W_u^{in} and the output of the previous state (recurrent) matrix W_u. In ESN output weight matrix W_u^{out} required to estimate the prediction function. W_u^{in} indicates the relationship between user information and request distribution of user u. Therefore, the dynamic reservoir of user u is represented as (W_u^{in}, W_u). The initial values of the dynamic reservoir are randomly generated using the uniform distribution. W_u^{in} is initialized randomly using uniform distribution and updated constantly in successive training. ESNs performance is decided by different parameters (sparse degree, spatial radius, input extensions and hidden layer size) of the reservoir.

Assume that the number of nodes in hidden layer is l of user u then the ESNs state at time t is represented as: $z_u^t = [z_{u1}^t, z_{u2}^t, \cdots, z_{ul}^t]$. The reservoir state of user u at time t is represented as z_u^t, and stores the state of user u. The updated equation is represented as:

$$z_u^{t+1} = tanh(W_u^{in} \cdot q_u^{t+1} + W_u \cdot z_u^t + W_u^{fb} \cdot h_u^t), \tag{3.16}$$

where W_u^{fb} is the weight matrix of the output of the previous state to the dynamic reservoir of the next state. For each request of user u, the output vector of the ESN model records the content request distribution. The ESN model output at time $t+1$ is:

$$h_u^{t+1} = tanh(W_u^{out}[z_u^{t+1}; q_u^{t+1}]), \tag{3.17}$$

where [;] represents the vertical concatenation of two vectors and W_u^{out} is output matrix at time t. The expected output of the ESN model is $H(e)$. $[z_u^{t+1}; q_u^{t+1}]$ is collected and stored in vector Z and corresponding output is stored in H, therefore,

$$H = W_u^{out} \cdot Z \tag{3.18}$$

The difference between the predicted output H and original output $H(e)$ should be minimized by adjusting W_u^{out}. The W_u^{out} is adjusted by training.

$$H(e) = W_u^{out} \cdot Z \tag{3.19}$$

where $H(e)$ is expected output. By applying pseudo inverse

$$H(e) \cdot Z^{\dagger} = W_u^{out} \tag{3.20}$$

$$Z^{\dagger} = (Z^T Z)^{-1} \cdot Z^T \tag{3.21}$$

with (3.20) and (3.21) the W_u^{out} becomes

$$W_u^{out} = H(e) \cdot (Z^T Z)^{-1} \cdot Z^T \tag{3.22}$$

Due to noise, the W_u^{out} may result in larger weights. Therefore, output weights are computed using the ridge regression with Tikhonov regularization:

$$W_u^{out} = H(e) \cdot Z^T (Z \cdot Z^T + c \cdot I)^{-1} \tag{3.23}$$

where c is the regularization coefficient and I is identity matrix.

3.3.2 Fuzzy Inference System for cache node selection

Fuzzy logic has been widely used in large number of applications because of its easy adaptation, interpretation of the rules and study on different inputs. The ability to use het-

erogeneous inputs facilitates the blending of several factors most desirably without using mathematical relations [154, 155]. In the content placement problem, a content with high benefit, large deadline and more popularity gets good opportunity to cache. A fuzzy logic system has been presented to adjust the various content properties and their influencing factors, choosing the more priority content to be cached based on well-chosen factors.

The uncertainties involved in computing the chance to become a caching node are handled by fuzzy inference system (FIS) [154]. FIS utilizes the benefit, content popularity prediction and deadline of the content to determine the chance of becoming a cache node. The content with the nearest deadline needs to be cached in such a way that the delay should be less than the deadline. The content with low popularity prediction means that content has less chance of requests in the future. Similarly, a node with less benefit is also not preferable. Therefore, by employing these parameters, each node calculates its chance to cache the content by FIS.

Fuzzy modelling involves two distinct identification aspects, structure identification and parameter identification. Structure identification includes the selection of input-output variables, selecting a specific FIS, defining the linguistic terms for input and output variables and generating rules. Parameter identification includes choosing an appropriate membership function (MF), applying heuristic selection and refining the MFs with suitable optimization techniques. To tune the generated membership function, heuristic selection (common-sense knowledge, general information about the system) is used. Algorithm 3.2 shows the process of the fuzzy logic system. The fuzzy inference system, as shown in Fig. 3.2 consists of four components: fuzzification, rule-base, inference process and defuzzification.

Fuzzification: Fuzzification is the process of mapping data to suitable linguistic variables. It determines the degree to which a crisp input belongs to each of the suitable fuzzy set.

$$F : R- > \mu(x)$$

The triangular membership function (line 1, Algorithm 3.2) is used as the parameterized

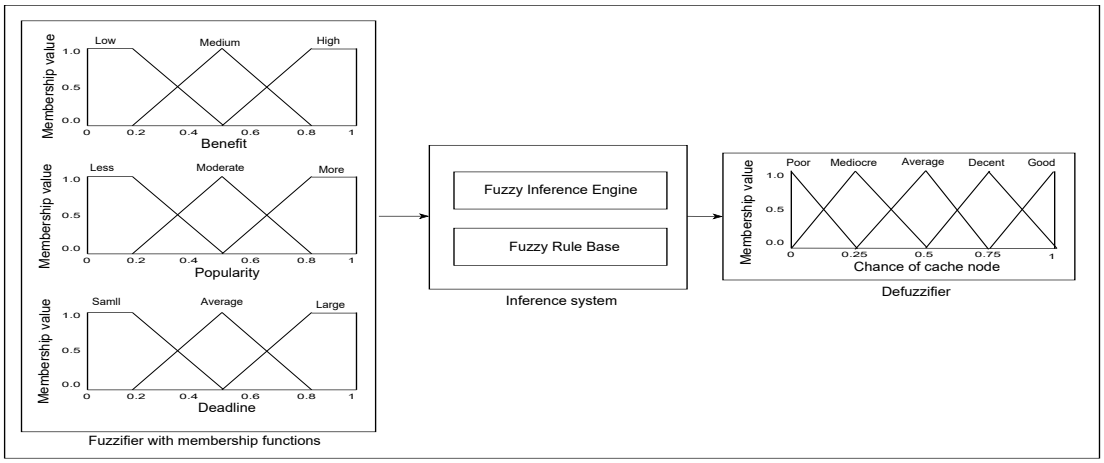


Figure 3.2: Fuzzy inference system

membership function, as shown in Fig. 3.2.

$$\mu(x) = max\Big(min\Big(\frac{x-k}{l-k}, \frac{m-x}{m-l}\Big), 0\Big) \tag{3.24}$$

The parameters k, l, m with $k < l < m$ determine the x-coordinates of the three corners of the triangular membership function. The linguistic variables of the input are considered as *benefit*, *content popularity prediction* and *deadline*. The linguistic terms for benefit are Low, Medium, High, content popularity are less, moderate, more and deadline are small, average, large. The linguistic variables of output variable is considered as *chance* with linguistic terms poor, mediocre, average, decent, good as shown in Table 3.2.

Table 3.2: Fuzzy input or output variable with their linguistic values

In/Output linguistic variable	**Linguistic values**
Content Popularity	Less, Moderate, More
Deadline	Small, Average, Large
Benefit	Low, Medium, High
Chance of Cache	Poor, Mediocre, Average, Decent, Good

Inference process: It is a mapping from the given input space to output space using the rules. Mamdani [154] type fuzzy model is used in this chapter.

Rule-base: It specifies the control goals and control policies of domain experts by a set of linguistic rules. Generally, fuzzy rules are produced based on experimental or expert

opinion (heuristic data) [156]. FIS consists of a set of rules of the form *IF (set of conditions satisfied) THEN (set of consequences can be inferred)*. Here, the linguistic statements are produced based on heuristic data (if-then rules) according to the following principle: a node with less benefit, low content popularity and small deadline have less chance to become the node to cache the content. According to three fuzzy input parameter and one output parameter, Table 3.3, shows the 27 fuzzy rules.

Table 3.3: Fuzzy Rules

Inputs			Output
Benefit	Deadline	popularity	Chance of Cache
Low	Small	Less	Poor
Low	Small	Moderate	Poor
Low	Small	More	Poor
Low	Average	Less	Poor
Low	Average	Moderate	Poor
Low	Average	More	Mediocre
Low	Large	Less	Mediocre
Low	Large	Moderate	Mediocre
Low	Large	More	Average
Medium	Small	Less	Poor
Medium	Small	Moderate	Mediocre
Medium	Small	More	Average
Medium	Average	Less	Poor
Medium	Average	Moderate	Mediocre
Medium	Average	More	Average
Medium	Large	Less	Mediocre
Medium	Large	Moderate	Average
Medium	Large	More	Decent
High	Small	Less	Mediocre
High	Small	Moderate	Average
High	Small	More	Mediocre
High	Average	Less	Average
High	Average	Moderate	Decent
High	Average	More	Good
High	Large	Less	Average
High	Large	Moderate	Good
High	Large	More	Good

Defuzzification: It is the inverse of fuzzification. It maps the fuzzy sets into a crisp output.

$$DF : \mu(x) -> R$$

Algorithm 3.2 Fuzzy_Cache_Node(B, D, P)

INPUT: Benefit(B), Deadline(D), Content Popularity Prediction(P), Rule Base
OUTPUT: Probability to cache

Find the membership values ($\mu(D)$, $\mu(B)$ and $\mu(P)$) and membership levels using *triangular membership function (3.24)*;
Empty the list $p(value, membershiplevel)$;
RuleBase = {A set of all combinations of linguistic levels}
for all rules in the $RuleBase$ **do**
    if $\mu(D)$, $\mu(B)$, $\mu(P)$ **then**
        fit the membership levels of this rule;
        add an entry to the list p with;
        value = maximum($\mu(B)$, $\mu(D)$, $\mu(P)$);
        membershiplevel = output membership level of this rule;
    end if
end for
Chance = Defuzzify(p) using equation (3.25);
return *Chance*;

Center of Gravity (COG) is considered for the defuzzification process.

$$chance = \frac{\sum_{i=1}^{n} x_i.\mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)} \tag{3.25}$$

Input parameters:

The input parameters considered for the selection of the cache node are

1. Benefit (B): The saved delay of content f at base station i is denoted as the benefit B

$$B = \arg\max \Big(\sum_{u \in i} \frac{Ben_{ui}(f)}{B_f} \Big) \tag{3.26}$$

where

$$Ben_{ui}(f) = r_f \times (I - d_{u,i})$$

2. Deadline (D): The response time of the content is denoted as D.

3. Content request distribution prediction (P): Echo state network gives the content request distribution for each user using equation (3.18).

Output: Probability for a content to be cached at a BS.

Algorithm 3.2 shows the process of the fuzzy logic system. Line 1 calculates the membership values using triangular membership function with the membership levels. Line 3 derives the all combinations of membership levels. Lines 4 - 11 show the fuzzy inference process. Further, the fuzzy output is converted into crisp value by the defuzzifier in line 12.

3.3.3 Fuzzy Caching Algorithm

In this section, the fuzzy caching algorithm (FCA) is constructed. The idea of the fuzzy caching algorithm (Algorithm 3.3) is to cooperatively cache more popular content with minimal delay by considering content benefit, deadline and request prediction (popularity) to imporve the performance in terms of hit ratio, acceleration ratio and the number of requests satisfying deadline. Algorithm 3.3, relies on computing the content request distribution prediction based on the ESN by considering user context and request information.

Algorithm 3.3 shows the method of caching a content in appropriate BSs fuzzy logic. The content request distribution of each user in the communication range of a base station is computed in lines 2-4. The average of each predicted content request distribution of different types is computed by line 5. Line 6 sorts the average of content in a base station. Line 9 shows the computation of benefit and line 10 shows the chance of each content to be placed in a base station using fuzzy logic. Line 13 sorts the chance of each content in non-increasing order. Lines 15-21 show caching an item if the cache is empty, by choosing the first element from the chance computed and sorted. Lines 22-35 show the cooperative caching. While the cache is not full, the content which is not cached so far overall BSs is chosen and the content will be cached if it satisfies the storage capacity. Lines 36-45 show user request allocation. For all the users in the communication range of node i, if the requested content is cached at i then the request is served. Otherwise, find the nearest node with less distance and serve the content.

3.3.4 Replacement Strategy

Caching algorithm (Algorithm 3.3) provides content placement which can be done during peak-off time. However, re-configuring large scale system during peak hours causes extra

Algorithm 3.3 Fuzzy Caching Algorithm

INPUT: $\{B_1, B_2, \cdots, B_F\}, p_i(f), d_{u,i}, \{S_1, S_2, \cdots, S_R\}$
where $u = \{1, 2, \cdots, U\}$, $i = \{1, 2, \cdots, R\}$, $c = \{1, 2, \cdots, C\}$ and $f = \{1, 2, \cdots, F\}$.
OUTPUT: x_{cf}^{i} : Content placment matrix and y_{cf}^{ui} : Content fetch matrix.

for all $i \in R$ **do**
  for all u in communication range of i **do**
    $P_i^u =$ compute request distribution for user u using equation (3.18)
  end for
  $P_i =$ compute average of each f form P_i^u
  sort P_i in non-increasing order
  for all $c \in \mathcal{C}$ **do**
    for all $f \in \mathcal{F}$ **do**
      $b_i^f =$ compute benefit using equation (3.26)
      $fz_i^f = Fuzzy\_Cache\_Node(b_i^f, \mathcal{T}_{cf}, P_i(f))$
    end for
  end for
  sort fz_i in non-increasing order
end for
for all $i \in R$ **do**
  f = first content of sorted fz_i
  if cache is empty **then**
    x_{cf}^{i} = 1, (i.e., cache f in BS i and remove f from fz_i)
    $S_i = S_i - B_f$
  end if
end for
for all $i \in R$ **do**
  while cache is not full **do**
    f = first content of sorted fz_i
    if f is already cached in some BS **then**
      f = succeeding item from fz_i to cache
    end if
    if $B_f <= S_i$ **then**
      x_{cf}^{i} = 1, (i.e., Cache f in BS i)
      $S_i = S_i - B_f$
    else
      Remove f from fz_i
    end if
  end while
end for
for all $i \in R$ **do**
  for all u in communication range of i **do**
    if $f \in S_i$ **then**
      y_{cf}^{ui} = 1
    else
      j = the node with less distance to user
      y_{cf}^{uj} = 1
    end if
  end for
end for

burden to the backhaul. Hence, a replacement strategy is presented to reconfigure the content placement for a dynamic network. When the requested content is not available at a node and the cache is full, then the requested content will be downloaded from the content server to node. This results in the content miss and the overall delay will be increased. As the cache of the MEC is full, NEF decides the replacement of the new content with existing content. NEF takes the decision on the content replacement based on the advantage of the MEC.

Algorithm 3.4 Cache Replacement Algorithm

1: For each request for a content f: $f \notin X$
2: **for all** $l = 0 : R$ **do**
3: f_{rep} = the content which is having less chance to cache using fuzzy logic among all nodes
4: **if** $B_f <=$ left over capacity of node **then**
5: replace the f_{rep} with f
6: **end if**
7: **end for**

Algorithm 3.4 shows the cache replacement strategy when there is a content miss. Line 1 represents that when there is a cache miss, then the process will be initiated. Line 2 shows the search for the R+1 iterations. Line 3 finds the content with less chance to cache among all the base stations. Lines 4-6 show to replace the requested content with the content which is having less chance to cache.

3.4 Performance Evaluation

In this section, the performance of the proposed cache placement algorithms (FCA, RAR) has been validated using simulations. FCA, RAR has been compared with the existing algorithms [157, 114, 139, 18, 49] based on publicly available real-world datasets.

3.4.1 Description of Data Set

In this simulation, the *MovieLens* 1M Dataset [37] has been used to evaluate the proposed FCA and RAR, that has 6040 users with demographic information (age, gender, location

and occupation) and 1000209 ratings of 3952 movies. The dataset consists of user ID, movie ID, movie ratings and time stamp. The timestamps divided as slots of one day each, and assigned the user context information to the user requests [158]. A movie rating from users is considered as the number of requests of that movie [131]. The user request generation probability is computed as the fraction of requests generated by a user over total requests of the all users in region. The content popularity is obtained in each time slot. The popularity of content is computed as the fraction of requests for a movie over requests for all movies. Content is updated based on the network traffic pattern. The wireless traffic presents a regular high and low every day. Thus, content updation can be carried out in the off-peak time to reduce the burden on backhaul [159]. It has been observed that more than 90% of the ratings existed within the first year. Therefore, only the first year of the dataset [129] has been used. ESN uses the traces from dataset to train predict the distribution. The performance of content popularity prediction is measured as an error. The error of content distribution prediction is defined as the sum deviation from the estimated distribution of content request to its original distribution [15].

3.4.2 Simulation Environment

In order to evaluate the performance of the proposed fuzzy caching algorithm, the experiments have been executed based on the following settings. A cellular network with 15 BSs associated with MEC servers and 90 mobile users have been considered. In the given simulation area, the MECs along with base stations are randomly deployed and there is a link between the base stations. The users under each base station are placed uniformly. In the content server, there are 3952 contents of 1128 labels (as per MovieLens dataset). Six hundred movies have been chosen for simulation. The content sizes are chosen uniformly at random from the range 300 K to 1999 K. The cache capacity of each BS is 20 MB. The communication range of the base stations is 100 m respectively. The data rate of the BSs is 5 M. The values of the simulation parameters are presented in Table 3.4. All the simulation results shown are average of 50 runs. The download delay of retrieving content from the content server to the BS is considered as 25 ms [160], while the latency of retrieving the

content from the BS to the user is calculated based on the positioning of the user and the BS. The deadlines of each file are assigned randomly from the range of 5 ms to 20 ms.

3.4.3 Performance Metrics

To compare the performance of cache placement schemes three metrics has been considered:

(1) *Cache hit ratio*: the fraction of requests satisfied (i.e., cache hits) from the available caches over sum of cache hits and cache misses.

$$\text{Cache hit ratio} = \frac{\text{cache hits}}{\text{cache hits} + \text{cache misses}} \tag{3.27}$$

(2) *Acceleration ratio*: the fraction of saved transmission delay and original Internet delay can be formulated as:

$$\text{Acceleration ratio} = \frac{\text{saved delay}}{\text{original delay } \text{(from Internet)}} \tag{3.28}$$

(3) *Number of requests satisfying deadline.*

(4) *Cache utilization*: the proportion that content in caches of base stations is accessed by users. i.e., the cache storage utilization indicates the utilization of content cached by BSs.

Table 3.4: Simulation Parameters

Parameters	**Values**
Simulation area	$4500/m \times 3400/m$
Capacity of base station	20 MB
Communication range of BS	100 m
Communication speed of BS	5 M
Latency from content server to the BS	25ms

3.4.4 Reference Algorithms

In this section, the proposed *Fuzzy Caching Algorithm (FCA)* and *Relaxation-Rounding Algorithm (RAR)* has been compared with Cooperative Prediction Caching Algorithm (CPCA), Non-cooperative Prediction Caching Algorithm (NPCA) [157], Random Caching (RC)

[114, 139, 15], Most Popular Content (MPC) [18, 49] and Least Recently Used (LRU) [47].

1. MPC: In the most popular caching scheme, each base station caches the most popular content estimated based on the user request statistics. Each base station caches the popular content till the cache is full [18, 49].

2. NPCA: In non-cooperative prediction based caching scheme, each BS predicts the user request distribution based on content request statistics. Each base station caches the predicted popular content non-cooperatively till the cache is full [157].

3. CPCA: In cooperative prediction based caching scheme, each BS predicts the user request distribution based on user context and content request statistics. The base stations cache the predicted popular content cooperatively till the cache is full.

4. RC: In random caching, each BS caches the content randomly irrespective of the content popularity till the cache is full [114, 139, 15].

5. LRU: It keeps a record with least access time for content and the newly requested content is replaced with the content which has been idle for a long time when the cache is full [47].

The proposed algorithms have been executed on a desktop with a dual-core Intel i5-5200U 3.20 GHz and 8 GB of installed RAM in this simulation. To find the solution for RAR, first the relaxed version of the problem is solved and obtained the fractional solution. A general idea for rounding the fractional values is to view the fractional values as probabilities. This technique is called randomized rounding. The main drawback of using the probabilities as caching variables in randomized rounding mechanism is that with certain inputs, it may take more time or sometimes it may fail [161]. Therefore, the fractional solution is rounded using the proposed deterministic algorithm.

Four scenarios are considered to show the performance of the proposed fuzzy caching algorithm. In *scenario 1*, the number of MECs is 7, the number of requests is 50% of total number of requests, the number of contents is 100% and the algorithms are compared in

terms of performance metrics by varying cache capacity from 5 GB to 10 GB with step size 1. In *scenario 2*, the cache capacity is 7 GB, the number of requests is 50% of total number of requests, the number of contents is 100% and the algorithms are compared in terms of performance metrics by varying number of MECs from 5 to 10 with step size 1. In *scenario 3*, the cache capacity is 7 GB, the number of contents is 100%, the number of MECs is 7 and the algorithms are compared in terms of performance metrics by varying, number of requests from 20% to 100% with step size 20. In *scenario 4*, the cache capacity is 7 GB, number of MECs is 7, number of requests is 50% of total number of requests and the algorithms are compared in terms of performance metrics by varying, number of contents from 20% to 100% with step size 20.

3.4.5 Impact of Cache Capacity

The impact of cache hit ratio, acceleration ratio and number of requests satisfying deadline on caching capacity of all schemes are presented in Fig. 3.3a, 3.3b and 3.3c respectively. The simulation results are computed by taking the inputs specified in *scenario 1*.

Fig. 3.3a shows the increase of hit ratio as the cache capacity increases. It can be observed from Fig. 3.3a that the hit ratio of proposed algorithms grow slowly with little cache capacity and grow quickly with increase of cache capacity. The proposed FCA provides an improvement in the cache hit ratio compared to other algorithms. The reason is that caching decision is made based on content request prediction, benefit and deadline. Therefore, the FCA caches more appropriate content compared to other algorithms. The CPCA caches content cooperatively based on prediction, whereas NPCA caches predicted content with non-cooperation and MPC caches based on the popularity of the content. It can be observed that the cooperative caching algorithms (FCA, RAR and CPCA) are performing better than non-cooperative caching mechanisms (NPCA, MPC, LRU and RC). The LRU is performing better than RC because it considers recency based caching. As the cache size increases, the hit ratio of proposed algorithms is significantly improved over other schemes. FCA achieves 8%, 15%, 20%, 19% and 18% better than CPCA, NPCA, RC, LRU and MPC, respectively.

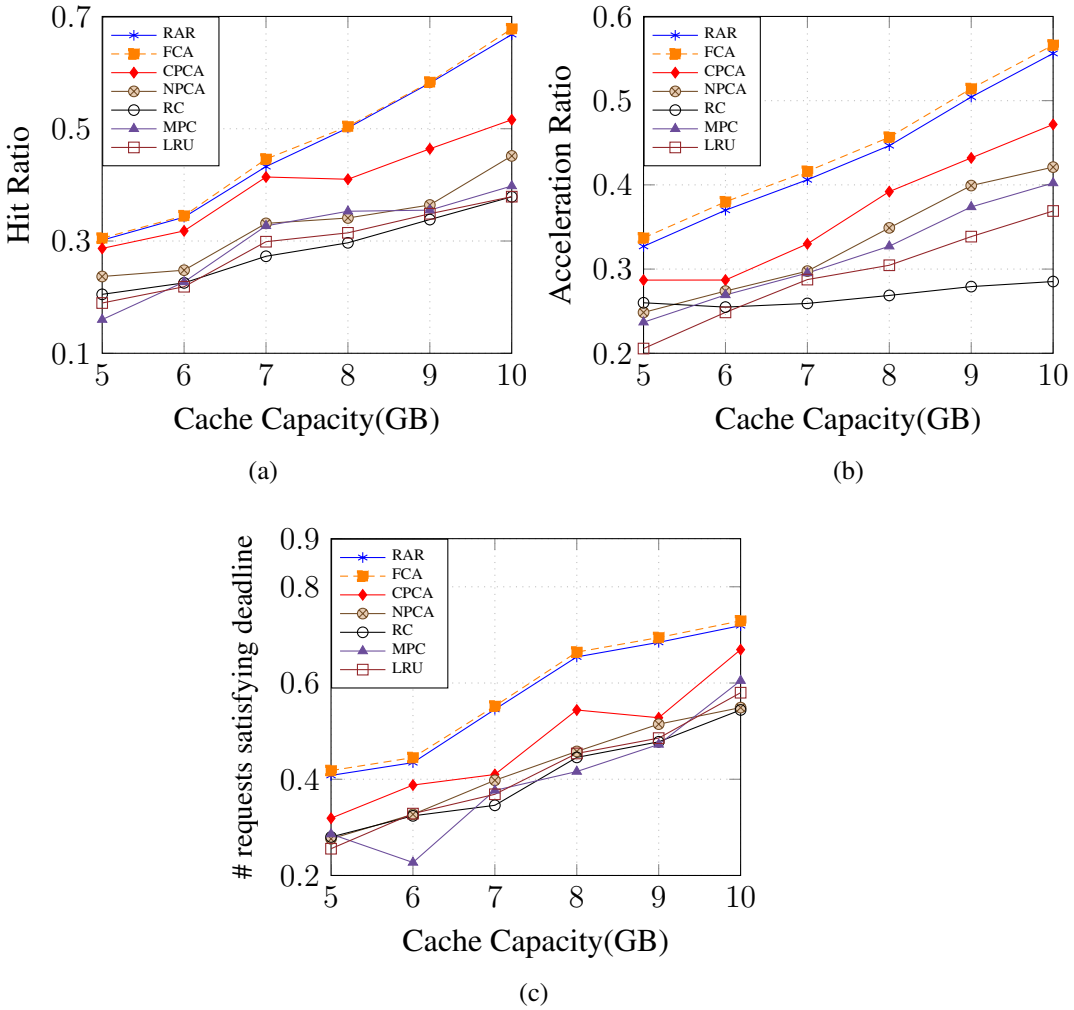


Figure 3.3: Comparison of caching schemes using cache capacity vs (a) cache hit ratio (b) acceleration ratio (c) number of requests satisfying deadline. The cache capacity is measured when $R = 7$, $r = 50\%$ and $F = 100\%$.

Fig. 3.3b shows the improvement of acceleration ratio among FCA, RAR, RC, MPC, LRU, CPCA and NPCA with various cache capacities. It can be observed from Fig. 3.3b that as cache size increases the acceleration ratio of all schemes increases. The increase in the acceleration ratio specifies that the increase in saved delay. The proposed caching algorithms perform better than other caching algorithms. The reason is that FCA makes the decision based on benefit, whereas CPCA and NPCA cache the content based on the prediction with cooperation and non-cooperation. Therefore, the content with more benefit is cached in the nodes. The acceleration ratio of RC increases slowly than other algorithms

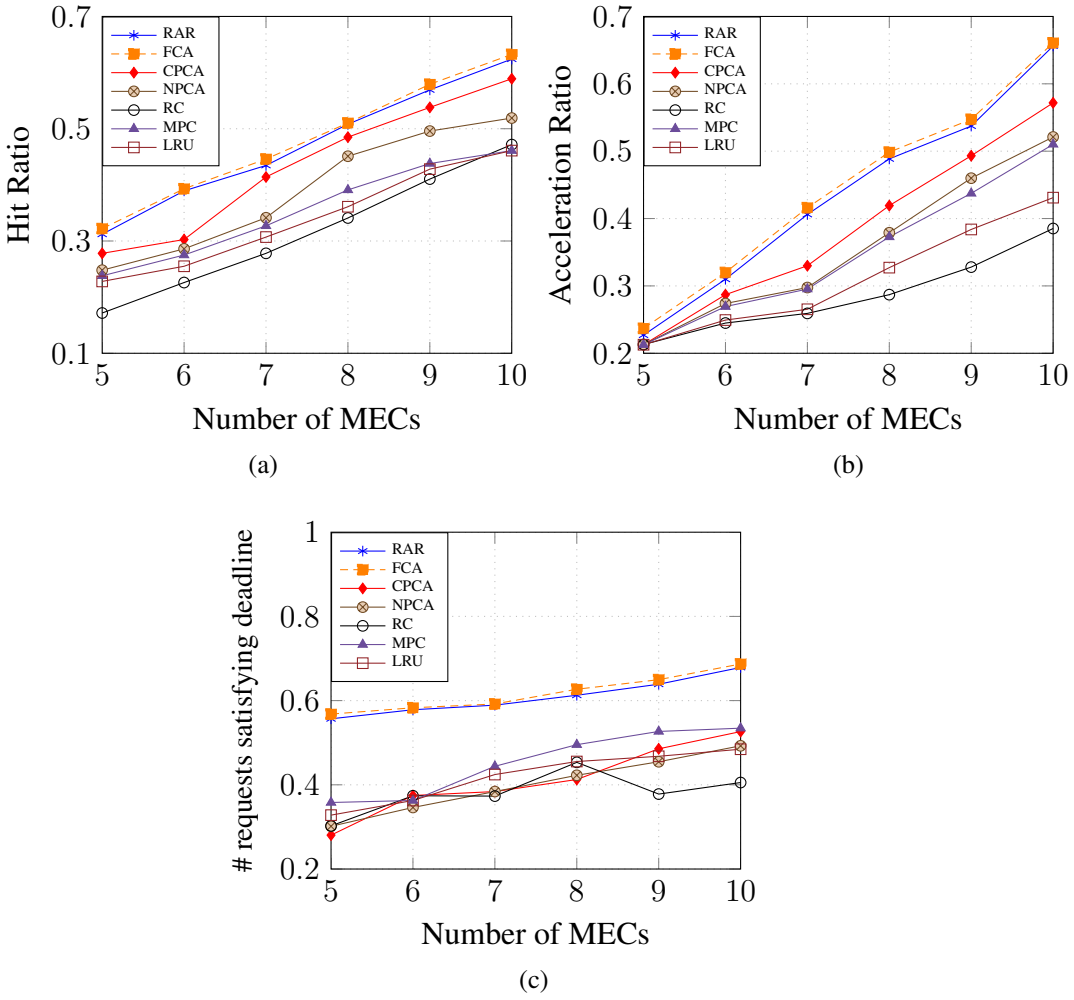


Figure 3.4: Comparison of caching schemes using number of MECs vs (a) cache hit ratio (b) acceleration ratio (c) number of requests satisfying deadline, when S = 7 GB, r = 50% and F = 100%.

due to caching random content at nodes. FCA achieves 8%, 12%, 18%, 15% and 13% better than CPCA, NPCA, RC, LRU and MPC.

Fig. 3.3c shows the number of nodes satisfying deadline among FCA, RAR, RC, MPC, CPCA and NPCA with various cache capacities. It can be observed from Fig. 3.3c that the proposed caching algorithms outperform with other mentioned caching algorithms. The reason is that proposed algorithms consider the deadline of the content in decision making. The number of nodes satisfying the deadline is less with a smaller cache size and grow quickly when the cache size increases. FCA achieves 11%, 16%, 18%, 17% and 21% better than CPCA, NPCA, RC, LRU and MPC.

3.4.6 Impact of number of MECs

In this section, the impact of the cache hit ratio, acceleration ratio and the number of requests satisfying deadline on the number of MECs of all the schemes is presented in Fig. 3.4a, 3.4b and 3.4c. The simulation results are computed by taking the inputs specified in *scenario 2*.

Fig. 3.4a shows that the proposed caching algorithms achieve better performance compared to other schemes in terms of hit ratio. The reason is that as the number of MECs increases, the proposed algorithms collaboratively caches more popular content results increase of cache hit ratio. FCA achieves 4%, 9%, 16%, 14% and 13% better than CPCA, NPCA, RC, LRU and MPC, respectively.

Fig. 3.4b shows that the proposed algorithms perform better than other schemes with respect to acceleration ratio. As the number of nodes increases, CPCA grows slowly and the proposed algorithms increases quickly. The reason is that the proposed algorithms caches popular content along with benefit. Therefore, proposed algorithms caches the content near (with less delay) to users. FCA achieves 6%, 9%, 16%, 13% and 10% better than CPCA, NPCA, RC, LRU and MPC, respectively.

Fig. 3.4c shows that the proposed caching algorithms outperform other schemes in terms of the number of deadlines. It can be observed from Fig. 3.4c that MPC is performing better than CPCA and NCPA. The reason is that the prediction based mechanisms CPCA and NPCA caches predicted content which may not satisfy the content deadlines. However, the proposed caching algorithms cache the predicted content that makes available more content to users. FCA achieves 21%, 22%, 24%, 20% and 17% better than CPCA, NPCA, RC, LRU and MPC, respectively.

3.4.7 Impact of Number of Requests

The effect of hit ratio, acceleration ratio and the number of nodes satisfying deadline with varying percentage of requests is presented in Fig. 3.5a, 3.5b and 3.5c. The simulation results are obtained by taking the inputs specified in *scenario 3*.

Fig. 3.5a shows that the proposed caching algorithms increase slowly when the number

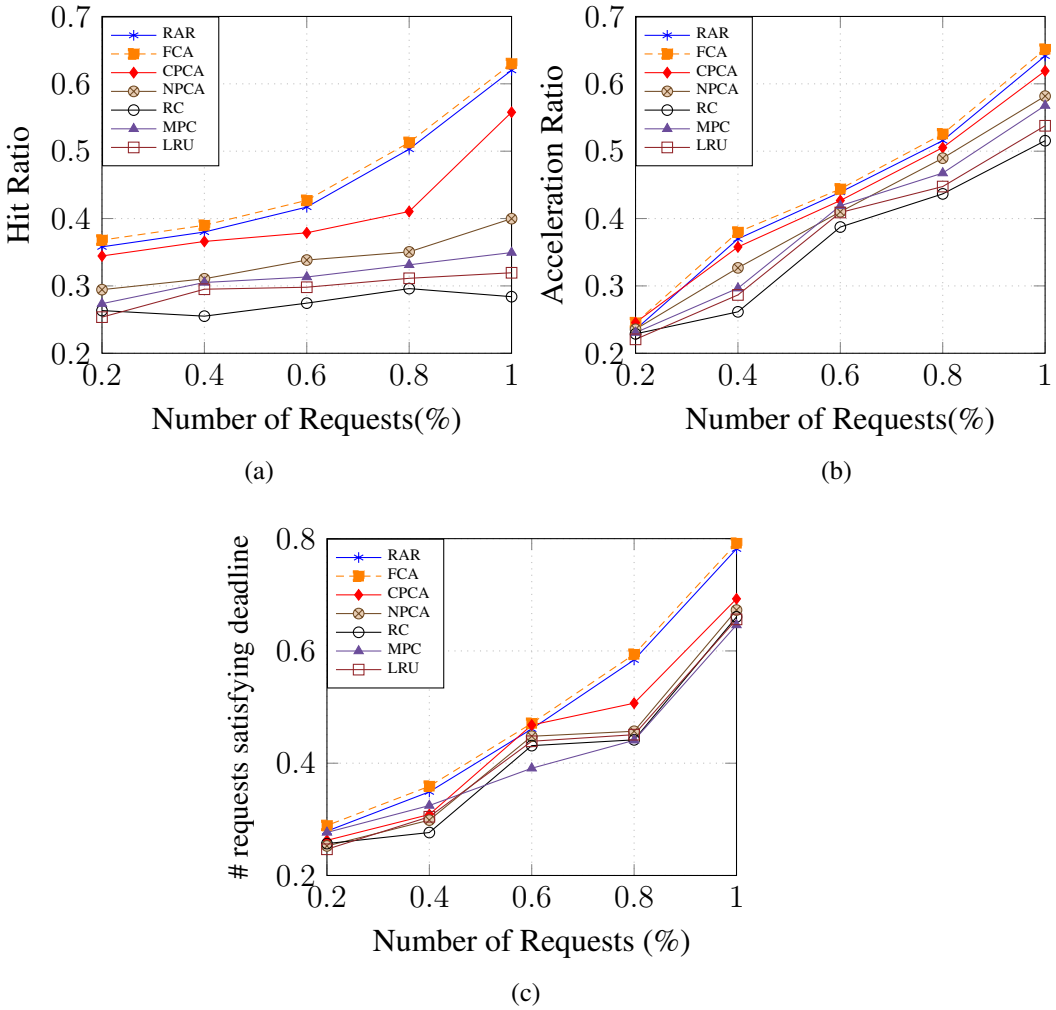


Figure 3.5: Comparison of caching schemes using number of requests vs (a) cache hit ratio (b) acceleration ratio (c) number of requests satisfying deadline, when S = 7 GB, R = 7 and F = 100%.

of requests are less and increases quickly with an increasing number of requests. It can be observed from Fig. 3.5a that FCA, RAR and CPCA are increasing quickly compared to other algorithms (NPCA, MPC, LRU and RC). The reason is that the cooperative caching mechanisms (FCA, RAR and CPCA) cache contents cooperatively, which improves the hit ratio with the increase in the number of requests. FCA achieves 5%, 12%, 19%, 17% and 15% better than CPCA, NPCA, RC, LRU and MPC, respectively.

Fig. 3.5b, shows the proposed algorithms compared with other caching schemes. FCA and RAR are performing better than other mentioned caching algorithms. The reason is

that with fewer requests, the saved delay is nearly equal for all algorithms. However, as the number of requests increases, the saved delay to access content is better with proposed algorithms because cooperatively caching the more benefit content. FCA achieves 2%, 4%, 8%, 7% and 5% better than CPCA, NPCA, RC, LRU and MPC, respectively.

Fig. 3.5c shows that the proposed caching algorithms increase quickly with the number of requests. FCA achieves 7%, 8%, 9%, 8% and 9% better than CPCA, NPCA, RC, LRU and MPC, respectively. The reason is that the proposed mechanisms consider deadline, along with prediction and benefit. NPCA, LRU and MPC are performing similarly because both the mechanisms are caching content non-cooperatively.

3.4.8 Impact of Number of Contents

The effect of hit ratio, acceleration ratio and deadline with the varying number of requests has been shown in Fig. 3.6a, 3.6b and 3.6c. The simulation results represented by taking the inputs are specified in *scenario 4*.

Fig. 3.6a shows that with less number of contents, the hit ratio is high. As the number of contents increases the hit ratio decreases. It can be observed from Fig. 3.6a that the cooperative caching mechanisms (FCA, RAR and CPCA) are performing relatively better than non-cooperative caching mechanisms (NPCA, MPC and RC). The reason is that the cooperative caching schemes cache more content cooperatively with less delay leads to more hit rate. FCA achieves 3%, 9%, 17%, 12% and 11% better than CPCA, NPCA, RC, LRU and MPC, respectively.

Fig. 3.6b shows that as the number of contents increases the acceleration ratio decreases. From Fig. 3.6b, it can noticed that the proposed algorithms is performing better than other mentioned caching mechanisms with increase in the number of contents. FCA and RAR decreases slowly compared to other mechanisms. The reason is that content caching decision is made using the benefit, so a large portion of the content is cached near to users. FCA achieves 4%, 8%, 13%, 12% and 9% better than CPCA, NPCA, RC, LRU and MPC, respectively.

Fig. 3.6c shows that the proposed caching mechanism decreases quickly when the

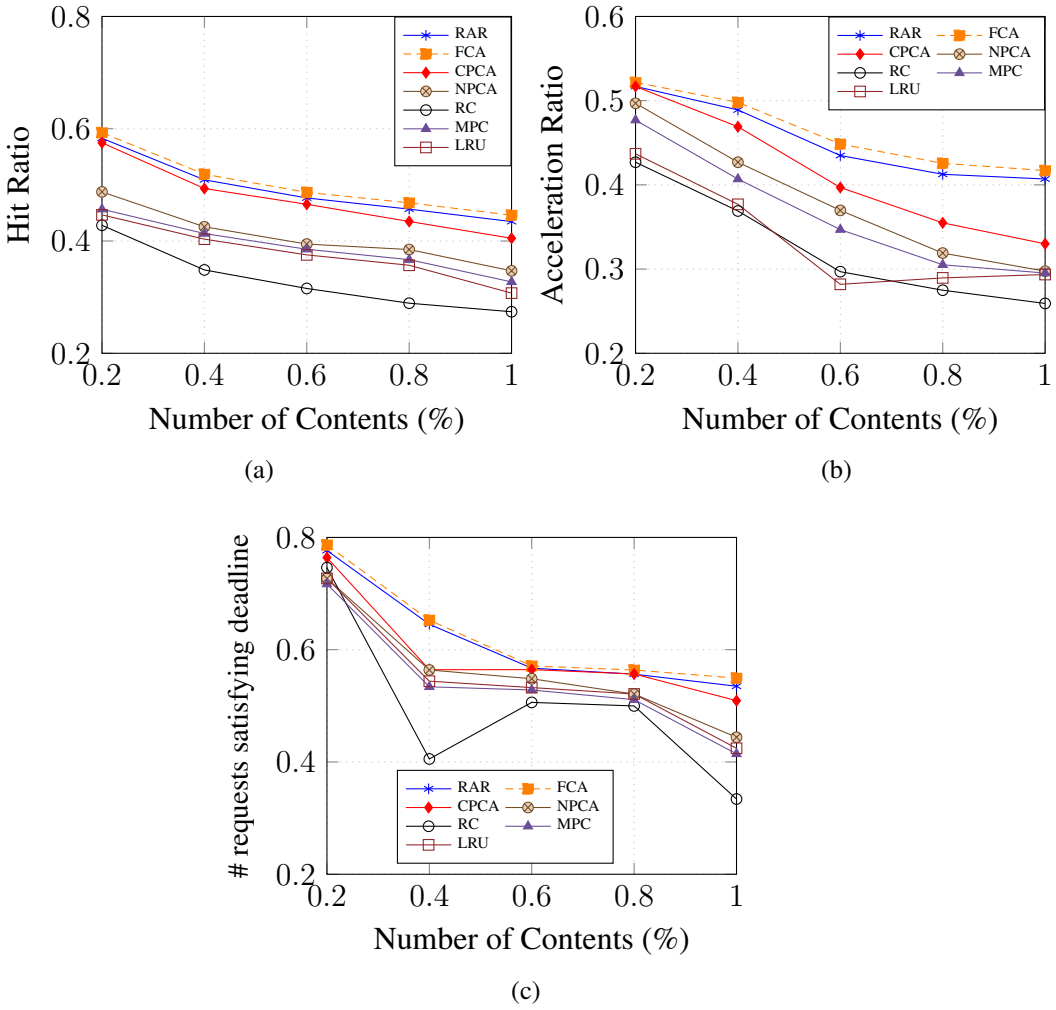
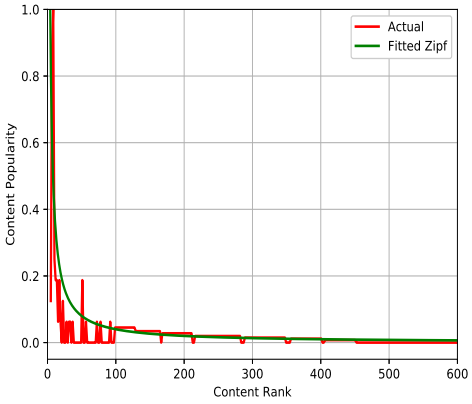


Figure 3.6: Comparison of caching schemes using number of contents vs (a) cache hit ratio (b) acceleration ratio (c) number of requests satisfying deadline, when S = 7 GB, R = 7 and r = 50%.

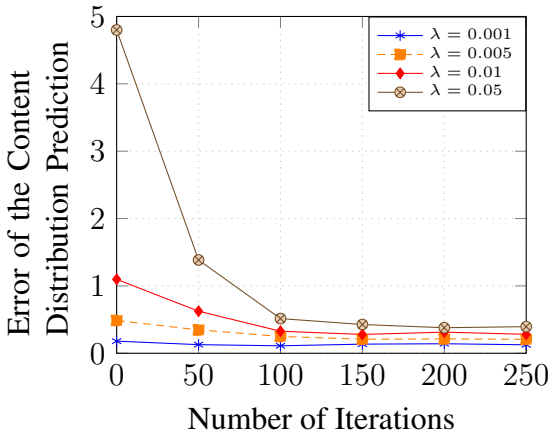
number of contents is less and decreases slowly with increasing of contents. The proposed algorithms outperforms other algorithms by satisfying content deadlines. FCA achieves 3%, 6%, 13%, 7% and 8% better than CPCA, NPCA, RC, LRU and MPC, respectively. The reason is that FCA considers deadline, along with prediction and benefit.

3.4.9 Impact of Content popularity

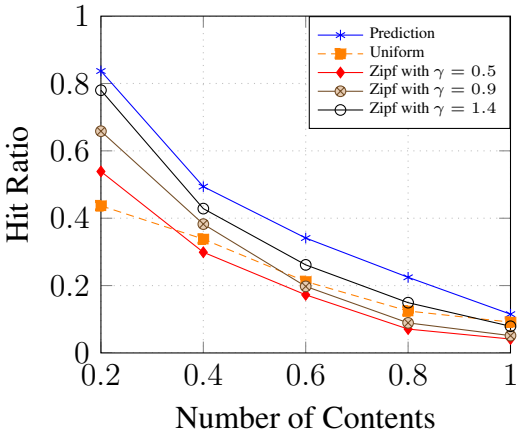
The effect of content popularity has been shown in Fig. 3.7a. From Fig. 3.7a, it can be noticed that the content popularity computed using Eq. 3.4 can be fitted by Zipf distribution



(a) Content popularity



(b) Accuracy of prediction

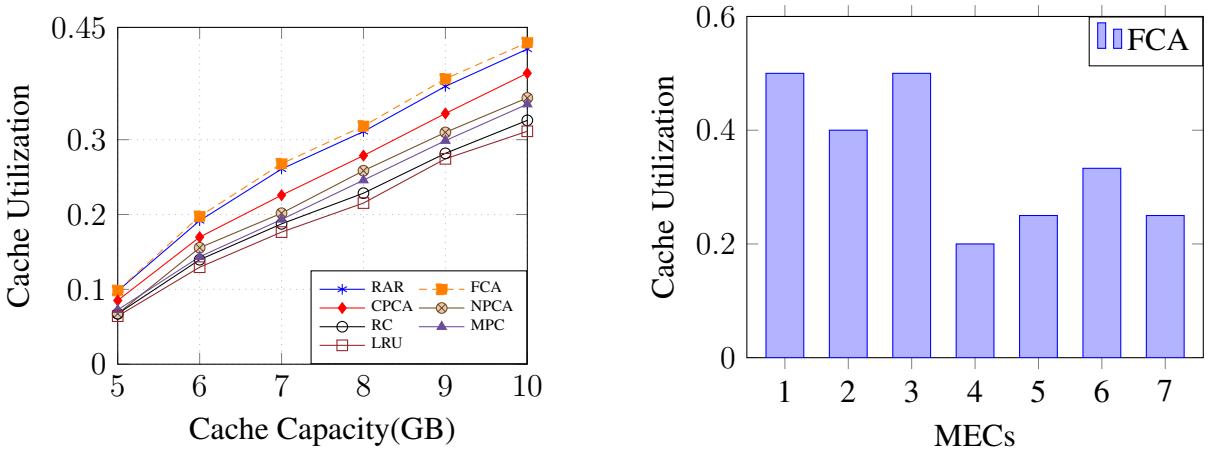


(c) Performance of prediction

Figure 3.7: Comparison of content popularity vs content rank, error as the number of iterations varies and performance of prediction vs number of contents.

with skewness parameter γ = 1.4.

The error of the ESN based prediction with a varying number of iterations is shown in Fig. 3.7b. From Fig. 3.7b, it can be observed that the error of the prediction is decreased by increasing number of iterations. It can be noted that ESN needs less than 50 iterations to predict the content distribution of each user for learning rate 0.001, 0.005 and 0.01. Whereas for the learning rate, 0.05 ESN takes less than 110 iterations. Since ESN require to trains only the output matrix. It also present that the error of 0.12, 0.2, 0.28 and 0.39 percentage for learning rates 0.001, 0.005, 0.01 and 0.05 respectively. Form Fig. 3.7b it can be observed that the learning rate affects the accuracy of prediction.



(a) Cache utilization vs capacity

(b) Cache utilization of each MEC

Figure 3.8: Comparison of caching schemes using cache capacity vs cache utilization. The cache capacity is measured when $R = 7$, $r = 50\%$ and $F = 100\%$.

The effect of the cache hit ratio with varying library size is presented in Fig.3.7c. In Fig. 3.7c, the performance of the content placement mechanism under different popularity distribution (uniform and Zipf [162]) is evaluated. The Zipf popularity distribution assumes the request probability of content j^{th} most popular content (form the set F) at node i can be computed as $p_{ij} = \frac{\frac{1}{j^{\gamma}}}{\sum_{n=1}^{F} \frac{1}{n^{\gamma}}}, \forall i \in \mathcal{R}$ where $\gamma \geq 0$ is the Zipf parameter. γ decides the rate of popularity decline as j values increases. The uniform distribution assumes the equal request probability of content j (i.e., $p_{ij} = \frac{1}{|F|}, \forall i \in F$). It can be observed from Fig. 3.7c that, as γ increases cache hit ratio also increases. It can also notice that as the number of contents increases, all the distributions decreases. Since with more number of contents the smaller set of contents report the identical popularities. The content request distribution prediction shows superiority over other distribution since the future requesting content becomes more popular, whereas in Zipf distribution evaluates popularity based on the content frequency.

3.4.10 Impact of Cache Storage Utilization

The effect of cache storage utilization with varying cache capacity of all schemes is presented in Fig. 3.8a. It can be observed from Fig. 3.8a that the rise in the cache capacity improves the cache storage utilization. Random caching is performing relatively well com-

pared to the LRU scheme. This is because, RC is free from the influence of popularity, which caches the content randomly. In LRU, the frequently used content is cached. However, there exist many contents which are not used at all. The prediction based caching schemes (FCA, CPCA and NPCA) is performing better compared to the non-prediction based schemes (RC, MPC and LRU). The performance of CPCA is superior to NPCA. Since CPCA uses cooperation among MECs. The FCA outperforms other algorithms because FCA caches content based on the popularity and benefit of the content. The cache utilization of each MEC is presented in Fig. 3.8b. It can observe that all the caches are utilized, and few caches are getting more utilization due to MEC located near other MECs. It can be observed from the Fig. 3.8b that the minimum cache utilization is 0.2.

3.5 Summary

This chapter addresses a cache placement problem in mobile edge networks, aiming to maximize saved delay by considering the capacity and deadline constraints. The placement problem is designed as an integer linear programming. A relaxation and rounding technique is used to design an approximation algorithm. Further, a *fuzzy caching algorithm* is proposed as a solution to the cache placement problem. In the proposed algorithm, first, an *echo state network* is used to predict content request distribution. Then the content to be cached (in the base station) is evaluated based on prediction result, benefit and deadline of the content request. Caching performance parameters, such as acceleration ratio, hit ratio and the number of files satisfying deadlines, are taken for comparison of the proposed prediction based fuzzy logic algorithm. Further, the performance of the proposed schemes are measured by comparing with the existing most popular content based algorithms, the prediction based caching algorithms (CPCA, NPCA), LRU and random caching algorithms. From the simulation results, it has been observed that there is an improvement of up to 20% on acceleration ratio, up to 18% on hit ratio and up to 24% on number of deadline satisfied. The next chapter presents a user preference-based cooperative cache placement strategy by considering the uneven distribution of users and the heterogeneity of user preferences and activity levels to improve cache utilization.

Chapter 4

User Preference Prediction based Cache Placement for Mobile Edge Networks with Adaptive User Clustering

Content popularity indicates the average interest of multiple users but not exhibits the individual user preferences [19]. Most of the existing literature considers that all users have the same content distribution (homogeneous popularity). However, various users have diverse preferences. The assumption made on homogeneous popularity ignores the users' preferences and this results in losing valuable information. Less than 20% of users generate 80% of traffic, which shows that the users' activity level is heterogeneous [20]. In the literature, most proactive caching approaches ignored user behaviour, such as heterogeneous user preferences and activity levels, introducing new challenges into mobile edge networks. Therefore, employing the individual user activity levels and preferences facilitate design of efficient cooperative caching strategies.

This chapter aims to maximize the saved delay by considering the uneven distribution of users, user preferences and activity levels for accessing a large volume of data. The content request deadline is considered for generality and practicality, which is reasonable in latency-sensitive mobile and IoT applications. The novelty of this chapter lies in designing a cooperative caching scheme for mobile edge networks with uneven user distribution, heterogeneous user preferences, and activity levels. User preferences are predicted using

the recurrent neural network mechanism LSTM using the historical user behaviour. Further, users are clustered depending on estimated user preferences. Then the cooperative cache placement problem is modeled as Integer linear programming to maximize saved delay with deadline and capacity constraints. For the modeled cache placement problem, a solution using a submodular function with matroid constraints has been designed to maximize saved delay.

The contributions of this chapter are as follows:

- Design a user preference prediction mechanism by adopting the long short-term memory network.
- Design a user preference-based clustering mechanism and formulate a clustered cooperative caching problem as an integer linear programming problem in mobile edge networks to maximize the saved download delay subjective to the deadline of the content and cache capacity.
- Design a submodular optimization based cooperative content caching algorithm by utilizing the clustering and prediction mechanisms to solve the proposed problem.
- Extensive simulations have been performed to show the efficacy of the proposed greedy cooperative caching algorithm by considering acceleration ratio, hit rate, and cache utilization.

The rest of this chapter is organized as follows. The system model is presented in Section 4.1. The user preference prediction using LSTM, user clustering and problem formulation are presented in Section 4.2. A greedy approximation mechanism has been presented in section 4.3. The simulation environment, and the results are discussed in Section 4.4. A summary of this chapter is mentioned in Section 4.5.

4.1 System Model

In this section, the system model is presented in detail.

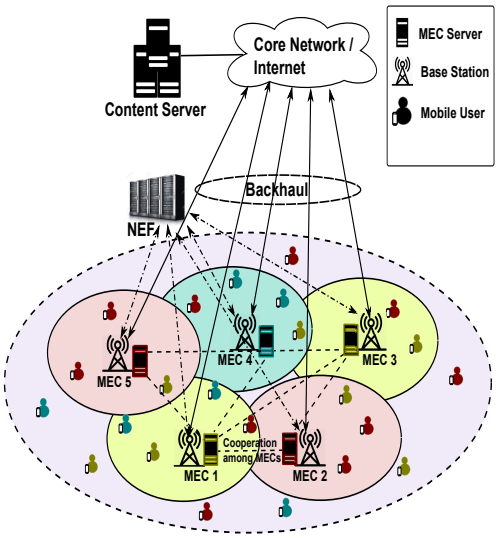


Figure 4.1: Illustration of the proposed system model.

Mobile edge computing improves users' capabilities by providing cache capacity (i.e., storage), network resources and computing near users. Consider a mobile edge network containing a set $\mathcal{R}$ of R small base stations (SBS) (i.e., edge node) is equipped with a MEC server, a set $\mathcal{U}$ of U mobile users, a content server and a central coordinator NEF as shown in Fig. 4.1. The distribution of base stations modelled as a Poisson point process (PPP) with density λ_R. The base station voronoi cells are non-overlapping. The users are distributed within the coverage of BSs as per PPP with density λ_U. The users in a voronoi cell associates with MEC located in that particular cell. Each MEC $i \in \mathcal{R}$ has a limited cache S_i called local storage. The storage of each MEC is used for content caching. The MECs are connected and also to the core network through the backhaul link. The content server acts as an origin server that stores all contents. Network Exposure Function (NEF) serves as a coordinator (it is a crucial network element in 5G networks) [151]. NEF has a global view, maintains the content cached at individual MECs and monitors users' content requests at each MEC [151]. Each user has different activity levels at each BS, 80% of the total traffic generated by less than 20% of the all users. A user directly connected to a base station and the user may be in the communication range of more than one BS at any point in time. However, any user can communicate with only one BS at a particular time. Mobile users are attached to the base stations according to a cellular network protocol. The connected base stations are accountable for serving user requests. Consider a set $\mathcal{F}$ of F

contents in the content library located in the content server. Each content f is determined with two features B_f indicates the content size and dl_f indicates maximum allowed access latency to get content f. The summary of notations is shown in Table 4.1.

Table 4.1: List of Notations

Term	Definition
$\mathcal{R}, \mathcal{F}, \mathcal{U}$	Set of base stations, contents and users, respectively
S_i	The cache capacity of MEC i
λ_R, λ_U	Density of the BSs and users in PPP model
B_f	The size of content f
dl_f	The deadline of content f
p_f^i	Local content popularity in cell i
p_f	Global popularity of content f
$p_{f\|u}$	User preference of content f
ρ_f, ρ_u	Total number requests for content f and all requests by user u
ρ_f^u	Number of times content f requested by user u.
$P(v_i)$	Activity probability of user at cell i
$i(t), o(t), f(t)$	input, output and forget gates
W^x, W^h	input and hidden state weight matrices
$x(t), h(t-1)$	present input and previous hidden state
b_f, b_i, b_o	bias of forget, input and output gates
$d_{i,u}, d_{i,j}, d_{i,c}$	Delay from local MEC i, neighbour MEC j and content server
x_f^i	The content f cached in MEC i
D_u	The expected saved delay
C	Delay to get content from the content server
η_k	Set of BSs in cluster k

4.2 User Preferences Prediction and Content based Clustering

In this section, user preference prediction, content-based clustering, and maximization of saved delay optimization problem are presented in detail.

The overall process is illustrated in Fig. 4.2. Firstly, in this section, the user request information is predicted using LSTM by taking user historical information from NEF. Secondly, the user preferences is computed from the predicted user request information. Thirdly, the users are clustered into logical groups using content-based clustering and a

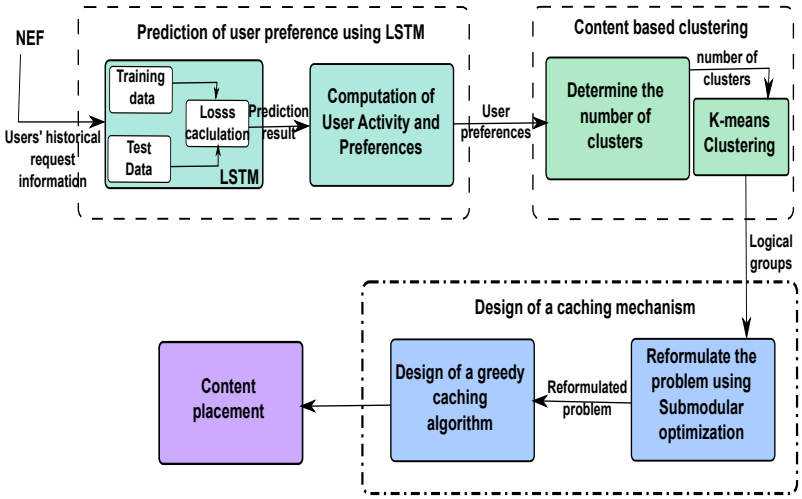


Figure 4.2: Content placement strategy based on user preference prediction and content based clustering.

clustered cooperative cache placement problem is formulated. In section 4.3, the proposed problem is reformulated into a submodular optimization problem by showing that the given objective and constraints satisfy the monotone submodular property and matroid constraints and a greedy caching algorithm has been designed to solve the content placement problem efficiently.

User preferences

The frequency of content accessed or requested by the user is expressed as the popularity of the content. The popularity of the content considered in a single cell is known as local content popularity (p_f^i) and the popularity of all cells is known as global content popularity (p_f). In general, the popularity of content is modelled with Zipf distribution. The popularity of the content f at MEC i can be expressed as $p_f^i = \frac{\frac{1}{i^\gamma}}{\sum_{n=1}^{F} \frac{1}{n^\gamma}}, \forall i \in \mathcal{R}$ where $\gamma \geq 0$ is the skewness parameter [11]. In reality, the content popularity cannot reflect the individual user behaviour. Each user has its content preferences that may not reflect global content popularity. The probability that user u demands a content f stated that the user u generates a request is known as user preference $p_{f|u}$ of content $f \in \mathcal{F}$ ($\sum_{f=1}^{F} p_{f|u} = 1$). Moreover, $p_{f|u} = P(f|v_i)$ where $P(v_i)$ is the probability that user u requests.

User activity level is the probability that a user forward a request for content from a cell and denoted as v_i. The number of times content f is requested by user u is denoted as

ρ_f^u. The total number of requests for all contents by user u and total number of requests for content f are denoted as ρ_u and ρ_f respectively. Therefore, total number of requests for all contents over total users is denoted as $\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}}\rho_f^u$. The activity of user is as follows

$$v_i = \frac{\rho_u}{\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}}\rho_f^u} \tag{4.1}$$

and the user preference $p_{f|u}$ can be written as

$$p_{f|u} = \frac{\rho_f^u}{\rho_u} \tag{4.2}$$

From (4.1) and (4.2) the probability that user u request for content f is computed as

$$p_f^u = v_i p_{f|u} \tag{4.3}$$

which is $P(v_i)p_{f|u}$. Therefore, it can be observed from (4.3) that which content requested by the user and active level of user need to be predicted.

4.2.1 User Preference Prediction based on LSTM

In MEC based network, the nearest edge nodes can serve user requests if the content is well placed and reduces the burden on backhaul links. In contrast, if the content is not appropriately placed at edge nodes affects the user experience. Therefore, user preference prediction plays a significant role in placing appropriate content in appropriate location to improve the effectiveness of caching. An LSTM model has been adopted to achieve a better cache placement strategy by considering the user regularity of the request pattern. The LSTM is extensively adopted RNN (recurrent neural network) for sequential data processing [163] and very successful in various applications like image captioning, speech recognition and machine translation. Hence, it is well suitable for user preference prediction.

The LSTM network comprises three layers an input layer, hidden layer (memory blocks) and an output layer to predict the user preferences. LSTM is an implementation of a classical RNN where the LSTM replaces the hidden layers with a new structure known as a

memory block [163]. The memory block includes three neural network blocks, such as an input gate, an output gate and a forget gate, and one or more self-connected memory cells. These gates enable the LSTM memory cell to store and access data over a long time, which helps solve the disappearing gradient problem. The input to the LSTM cell is the user preferences from t-1 time represented as $x(t-1) = x1(t-1), \ldots, xn(t-1)$. Output of the LSTM cell is predicted user preferences in t time denoted as $x(t) = x1(t), \ldots, xn(t)$. Let Y indicates the output variable. The input layer prepares the given dataset to satisfy the network input demands. The predicted values update the output layer. The BPTT algorithm (back propagation through time) trains the network, optimising the loss function. BPTT is an easier and computationally effective training mechanism. The loss function of the LSTM depends on both the output layer and hidden layer of the next timestamp. The loss function is defined as the squared error.

$$e(t) = \sum (y(t) - p(t))^2 \tag{4.4}$$

where $y(t)$ is actual values, $p(t)$ is predicted values and e(t) denotes the error.

The user preference predictions contain the user activity and the user's content requests in the next time slot. First the network will be trained for the given data then prediction can be done. In the training phase, the data of each user is supplied into the LSTM network. Once the network is trained, the values are predicted and this indicates the number of times a user requests content in time t. Based on the prediction result, compute the user activity level v_i and user preferences $p_{f|u}$. The user activity level is calculated as

$$\hat{v}_i = \frac{\hat{\rho}_u}{\sum_{u \in \mathcal{U}} \sum_{f \in \mathcal{F}} \hat{\rho}_f^u} \tag{4.5}$$

The conditional probability that the user request content given that the user really makes a request is calculated as

$$\hat{p}_{f|u} = \frac{\hat{\rho}_f^u}{\hat{\rho}_u} \tag{4.6}$$

Therefore, the user preference is expressed as

$$\hat{p}_{f,u} = \hat{v}_i \cdot \hat{p}_{f|u} \tag{4.7}$$

Placing the content at an individual time slot is not a cost-efficient solution, so the long-term request probability has been considered. Hence, the fixed time window M^{ft} has been assumed. The user preference probability $\psi_u(f)$ is computed as the average of the predicted preferences between the next time slot and fixed time window.

$$\psi_u(f) = \frac{\sum_{t0=M+1}^{M+M^{ft}} \hat{p}_{f,u}}{M^{ft}} \tag{4.8}$$

4.2.2 Content based User Clustering

Proactively caching the content at the BSs during off peak time require efficient features like user preferences and popularity of content to identify the frequently requested content accurately. Most of the literature assumes that each user's popularity is similar, whereas this work utilizes a clustering-based mechanism. User clustering based on user preferences allows us to evaluate the relationship between the users and design an effective caching strategy. Unlike conventional location-based clustering mechanisms, content-based clustering allows us to discover user request patterns to understand user preferences effectively. Content popularity based clustering has been considered by grouping the correlated users into the same group based on user popularity. The correlation is defined by the euclidean distance among the users' preference profiles.

4.2.2.1 Clustering Algorithm

Heterogeneous user preferences have been assumed in this work, but the user request pattern may have some correlation based on social relations among the user. Thus, the content-based clustering approach has been adopted to decrease the difference among the content popularity distribution of users in the respective cluster. The cluster number is not known in advance and need to be determined. In order to maintain the network to handle user request pattern modification, the system needs to determine the clusters based on user interest

Algorithm 4.1 Preference Based User Clustering Algorithm

1: Initialize N_{kmin} and N_{kmax}
2: **for** k = N_{kmin} to N_{kmax} **do**
3: Run K-means algorithm
4: Calculate Silhouette Coefficient SC_K using Eq. (4.9)
5: **end for**
6: Choose the k value as the best Silhouette Coefficient value

changes systematically. The number of clusters is determined using the Silhouette method [164] since the label information is unknown, so evaluation needs to be performed using the model. The Silhouette coefficient determines the coherence among the points within a cluster separated from other cluster points. The Silhouette coefficient $(SC(j))$ is computed using the mean nearest-cluster distance $(ncd(j))$ and the mean intra-cluster distance $(icd(j))$. The SC for a data point j is

$$SC(j) = \frac{(ncd(j)) - icd(j))}{max(icd(j), ncd(j))} \tag{4.9}$$

$ncd(j)$ is the average distance between j, and all other clusters to where j do not belong and $icd(j) = \frac{1}{|C_j|-1} \sum_{i \in C_j, j \neq i} d(j,i)$ is the average distance among all the other samples in the cluster and j. The distance between data point j and i is denoted $d_{j,i}$ and computed using the euclidean distance. SC values lie between -1 and 1 where 0 is overlapping clusters, -1 is the worst value, and 1 is the best value. The algorithm 4.1 computes the average SC for every k chosen from [min, max] and groups the users depend on the K-means algorithm [165]. In the K-means algorithm, group the users based on the preferences to the nearest centroid with a minimum distance between the user preference and cluster. The maximum SC value is obtained with the optimal K value.

4.2.3 Maximization of Saved Delay Optimization Problem

In this section, download delay is defined and further a problem formulation is done to maximizing saved delay with capacity constraints. The delay for getting content f from MEC i to user u is denoted as $d_{i,u}$. If the content demanded by the user retrieved from the local storage of the corresponding MEC, then the delay is considered as 0. In case of the

content is not available at corresponding MEC i then i forwards the request to neighbouring MECs as per the NEF direction. The delay is considered as the number of hops between MEC j and user u (j is the neighbouring node of MEC i) and denoted as $d_{j,u}$. If the content requested by the user is unavailable within the network, then the user fetches the content from the central server $d_{u,c}$ and $d_{c,u} > d_{j,i} > d_{i,u}, \forall j \neq i, j \in \mathcal{R}$.

Let $\eta_k(u)$ is set of base stations in cluster k, and its size is denoted as $\mid \eta_k(u) \mid$. Let $k \in K$ represents the serving cluster of a user. Suppose j_u represents j^{th} nearest BS (i.e., j^{th} smallest delay) index with user u. A decision variable x_f^i represents that BS i has the content f. The average download delay can be represented as

$$\begin{aligned} D_u = \sum_{f=1}^{F} \psi_u(f) \Bigg(& \sum_{i=1}^{|\eta_0(u)|} d_{u,i_u} x_f^{i_u} \Big[\prod_{h=1}^{i-1} (1 - x_f^{h_u}) \Big] \\ & + \sum_{k' \in G \setminus k} \sum_{j=1}^{|\eta_{k'}(u)|} d_{i_u,j_u} x_f^{j_u} \Big[\prod_{h=1}^{|\eta_{k'}(u)|} (1 - x_f^{h_u}) \Big] \\ & + \sum_{k \in G} d_{u,c} x_f^{c} \Big[\prod_{h=1}^{|\eta_k(u)|} (1 - x_f^{h_u}) \Big] \Bigg) \end{aligned} \tag{4.10}$$

In the above equation (4.10), it can be observed that $x_f^{i_u} \big[\prod_{h=1}^{i-1} (1 - x_f^{h_u}) \big]$ is the indicator function that describes no other BS in the serving cluster with a delay lower than h_u, i.e., h_u has the lower delay among all the BS in the cluster. $x_f^{j_u} \big[\prod_{h=1}^{|\eta_{k'}(u)|} (1 - x_f^{h_u}) \big]$ is an indicator function that describes content f is not in the cache of the BSs in the serving cluster and it is cached at the BS of other clusters. Moreover, the indicator function $x_f^{c} \big[\prod_{h=1}^{|\eta_k(u)|} (1 - x_f^{h_u}) \big]$ describes content f in not available with any BS in the network.

Definition 4.2.1 (Saved delay)**.** *The difference in delay from the content server and MEC node is defined as the saved delay.*

This chapter aims to find the caching mechanism that maximizes the overall saved delay of the requested contents at each MEC subjective to cache deadline and capacity constraints. Thus, the formulation becomes:

$$\text{Max } \frac{1}{U} \sum_{u=1}^{U} (C - D_u) \tag{4.11}$$

s. t.

$$\sum_{f=1}^{F} B_f \cdot x_f^i \leq S_i, \qquad \forall i \in \mathcal{R} \tag{4.12}$$

$$\sum_{f=1}^{F} x_f^i \leq 1, \qquad \forall i \in \mathcal{R} \tag{4.13}$$

$$D_u \leq dl_f, \qquad \forall f \in \mathcal{F}, \forall i \in \mathcal{R} \tag{4.14}$$

$$x_f^i \in \{0, 1\}, \qquad \forall f \in \mathcal{F}, i \in \mathcal{R} \tag{4.15}$$

The objective (4.11) is the total saved delay of the overall network. Constraint (4.13) guarantees that the MEC node is not allowed to cache duplicate content. Constraints (4.12) provides the finite capacity of each BS. Constraints (4.14) is the deadline constraint, which ensures that the maximum allowable delay for the response to a request. Thus, the BS can satisfy the users' QoS requirements. Finally, constraint (4.15) is the non-negativity and integrality of the decision variables.

The cooperative cache placement problem as shown in Eq. (4.11) is proved as NP-hard [24, 11]. Considering the combinatorial nature of the problem in Eq. (4.11), the optimal solution for the problem typically arrives with exponential computational complexity. Due to the exponential complexity, it is impractical to implement. Hence, a sub-optimal solution with low computational complexity needs to be designed for practical systems to implement efficiently. Therefore, the problem has been formulated as a submodular optimization problem that enables a way to apply a greedy approach for placement, which produces a $\frac{1}{2}$ approximation in the worst case [166]. The proposed problem Eq. (4.11) is reformulated into the sub-modular optimization problem, and then a greedy approximation algorithm has been designed in the next section.

4.3 User Preference based Content Placement Mechanism using Sub-modular Optimization

In this section, a sub-optimal solution based on submodular optimization [166] has been designed to solve the problem Eq. (4.11). The original problem presented in Eq. (4.11) is

reformulated into maximizing the monotone submodular function over matroid constraints, and a greedy algorithm is designed. First, the essential background knowledge like properties and definitions of submodular optimization have been presented.

Definition 4.3.1 (Submodular function)**.** *Consider a finite ground set N and a real-valued function $g : 2^N \to \mathbb{R}_+$. If the following properties are satisfied, then the function g is said to be submodular.*

1. $g(A) + g(B) \geq g(A \cup B) + g(A \cap B)$, *for all* $A, B \subseteq N$.
2. $g(A) \leq g(B)$, *for all* $A \subseteq B \subseteq N$.

The submodular function g is monotone submodular function if it satisfies the following properties. Suppose $g_B(j) = g(B + \{j\}) - g(B)$ where $g_B(j)$ denotes the marginal value of an element $j \in N$ concerning a subset $B \subseteq N$.

$$g_A(j) \geq g_B(j) \geq 0, \text{ for all } A \subseteq B \subseteq N \text{ and } j \in N - B. \tag{4.16}$$

The intuitive explanation of the monotone submodular function is that the gain of adding a new element decreases when the set becomes large.

Definition 4.3.2 (Matroid)**.** *Suppose N is a finite ground set and $\mathcal{M} \subseteq 2^N$ is a collection of subsets of N and then a pair $\{N, \mathcal{I}\}$ is called a matroid if it satisfies the following properties:*

1. $\phi \in \mathcal{I}$, *i.e.,* $\mathcal{I}$ *is nonempty.*
2. *If* $B \in \mathcal{I}$ *and* $A \subseteq B$ *then* $A \in \mathcal{I}$*.(downward closed)*
3. *If* $A, B \in \mathcal{I}$ *and* $|A| \leq |B|$*, then* $\exists\, j \in B - A$ *such that* $A \cup \{j\} \in \mathcal{I}$.

Matroids induce the notion of linear independence observed in linear algebra to general sets. The sets defined earlier are termed independent.

The ground set N is denoted as $N = \{y_f^i \mid f \in \mathcal{F}, i \in \mathcal{R}\}$ and $Y \subseteq N$ is the cache placement scheme. $y_f^i \subseteq Y$ denotes that content f is cached at MEC i. Suppose

$N^i = \{y_f^i | f = 1, 2, \ldots, F\}$ represents that overall configuration of content f is cached at MEC i. The relationship between cache placement Y and x^i is

$$x_f^i = |Y \cap N^i| \tag{4.17}$$

The following is a lemma.

Lemma 4.3.0.1. *The objective function in Eq.* (4.11) *is a monotone submodular function.*

Proof. The monotonicity of the objective functions is clear because any new placement of a file cannot decrease the objective function value. A new file $f \in N \setminus A$, suppose $A \cup y_f^i$. It is simple to demonstrate that $g(A \cup y_f^i) > g(A)$. Therefore, $g(A \cup y_f^i)$ is a monotonic function $A \subseteq N$. Suppose $B \subseteq N$ is another placement scheme and since g is a monotonic function we have Eq. (4.16). Because the sum of submodular functions is submodular, it is sufficient to establish that the set function $g(A) = C - D_u$ is submodular for a user u to verify the specified goal functions submodularity. The marginal benefit gained by adding a content f to a randomly chosen MEC node reduces while the placement set A grows big. The gain obtained by including content to the placement set A increases the objective function $g(A)$. The submodularity of function g can be proved by satisfying the following condition Eq. (4.16). Suppose two sets A and B are content placement sets where $A \subset B \subset N$. Suppose adding an element $y_f^i \in N \setminus B$ to the placement sets A and B for some $i \in \mid \eta_k(u) \mid$. When new content is added to the MEC node, it is not placed in either placement A or B. The following scenarios exist, depending on the cached content and the size of the MECs.

1. As per placement B, user u receives content f from MEC j with $j < i$, i.e., $y_f^j \in B$. From this it can be perceived that $g_u(B \cup y_f^i) - g_u(B) = 0$, i.e., there exist any MEC j with less delay than the MEC i, hence the marginal gain of including i to the content placement B is zero. As per the placement A, user u receives the content f form MEC k with $k \geq j$. If $k < i$ meaning that there exist a MEC k in the placement of A with less delay than i, so adding i to placement A gives marginal benefit zero. However, if $k > i$ the marginal gain is $g_u(A \cup y_f^i) - g_u(A) = \psi_u(f)(d_{u,k_u} - d_{u,i_u}) > 0$.

2. As per placement B, user u receives content f form MEC j with $j > i$. Therefore, the

marginal gain is $g_u(B \cup y_f^i) - g_u(B) = \psi_u(f)(d_{u,j_u} - d_{u,i_u})$. Since the user u receives the content from k with $k \geq j$ in the content placement A, the marginal value is $g_u(A \cup y_f^i) - g_u(A) = \psi_u(f)(d_{u,k_u} - d_{u,i_u}) > 0$. The difference between the marginal gains is $(g_u(A \cup y_f^i) - g_u(A)) - (g_u(B \cup y_f^i) - g_u(B)) = \psi_u(f)(d_{u,k_u} - d_{u,j_u}) \geq 0$. It can be seen that the difference between these two situations is always bigger than zero, indicating that g_u is a monotone submodular function in N.

Hence, the monotone submodularity of Eq. (4.11) is proved. □

Lemma 4.3.0.2. *Let N^i, where $i \in \mathbb{R}$ represents the set of contents that may be cached at MEC i, which is $N^i = \{y_f^i | f \in \mathcal{F}\}$. Then, (4.12), (4.13) and (4.14) can be rewritten as $A \in \mathcal{I}$, where*

$$\mathcal{I} = \{A \subseteq N \big| |A \cap N^i| \leq S_i, |A \cap N^i| \leq 1\} \tag{4.18}$$

which is a matroid constraint.

Proof. The pair $\{N, \mathcal{I}\}$ be a member of partition matroid, which is typical matroid [166]. □

The submodularity of the objective function with a matroid constraint can be decided from the above proofs.

4.3.1 Greedy algorithm for user preference prediction based cooperative content caching

The greedy approach gives an efficient solution with an approximation of $\frac{1}{2}$ to solve the maximization of monotonically submodular function with matroid constrain problems [166]. First, the greedy approach commences with an empty set of cache placement Y. Later, for each element, the marginal gain is calculated, and the element with a maximum marginal gain while fulfilling the matroid constraint is added to Y. The algorithm adds the elements till there are no more elements to be added or the MECs cache is full. Depending on the results from the lemma 2, it can be shown that the objective function is submodular. Thus, the marginal gain reduces as adding additional content to Y. The algorithm continues to add elements and stops when the marginal gain turns out to be zero.

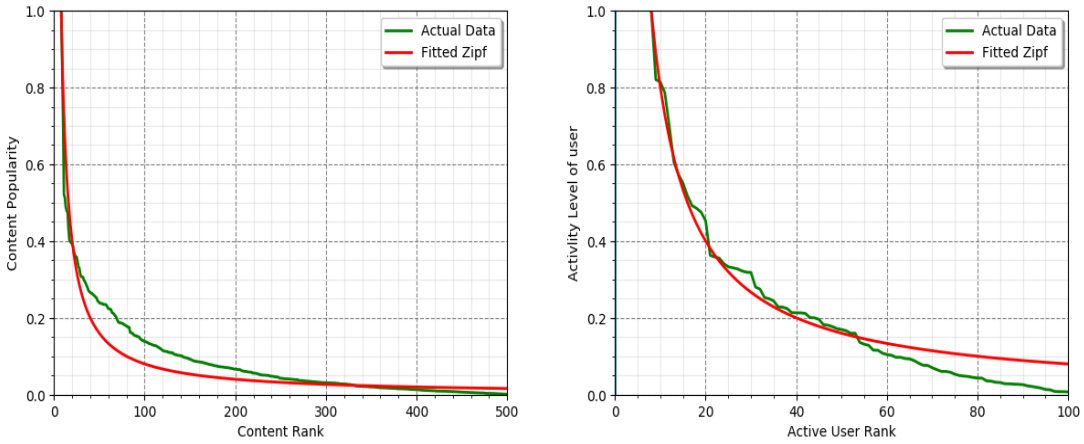
Algorithm 4.2 User Preference Prediction based Greedy Cooperative Content Placement Algorithm

1: Initialize $A = \emptyset$; /*i.e., $x_f^i = 0 \; \forall i \in \mathcal{R}$ and $f \in \mathcal{F}$ */
2: N_r = Set of all elements that may be added to A; /*i.e., $N_r = N$ assigning the ground set */
3: **repeat**
4: $\quad y_{f^*}^{i*} = \arg\max_{y_f^i \in N_r} [g_u(A \cup \{y_f^i\}) - g_u(A)]$;
5: $\quad A = A \cup \{y_{f^*}^{r*}\}$; /*i.e., $x_f^i = 1$ */
6: $\quad N_r = N_r - \{y_{f^*}^{r*}\}$;
7: $\quad x_{f^*}^{i^*} = 1$
8: $\quad$ **if** j^* is full **then** /*i.e., $|A \cap N^i| == S_i$ */
9: $\quad\quad$ Remove all the elements of N^i from N_r;
10: $\quad$ **end if**
11: **until** $|A| > \sum_{i \in \mathcal{R}} S_i$
12: **return** $x_f^i = \{A \cap N^i \mid i \in \mathcal{R}, f \in \mathcal{F}\}$;

The greedy algorithm is shown in Algorithm 4.2, where N_r represents the remaining set, consists of the elements included to X. Line 4 shows the highest marginal gain computation. Adding an element $y_{f^*}^{r*}$ to the placement A is shown in line 5. The inclusion of an element $y_{f^*}^{r*}$ to A should be removed from N_r. If the MEC i is full and no vacancy to store, then the corresponding content of the MEC i should be removed from the remaining set N_r shown in lines 8 - 10. Further, following the inclusion of an element $y_{f^*}^{r*}$ to A, based on Eq. (4.16), the marginal gain needs be updated.

4.4 Performance Evaluation

In this section, the proposed clustered cooperative caching mechanism has been validated using simulations. The real-world Lastfm 1K Dataset [38] has been used in the simulations to investigate the user behaviour of requesting content. A song dataset has been chosen to show the individual user request frequency for each content since songs may be accessed multiple times. The Lastfm-1k dataset has been considered, consisting of 19,098,852 records such as user ID, timestamp, artist ID, artist name, track ID and track name, comprising 107,295 artists and 1K users [38]. The top 500 popular content requested by users and the 100 most active users have been considered to analyze the user request statistics (that account for 90% of the requests are the Lastfm dataset).



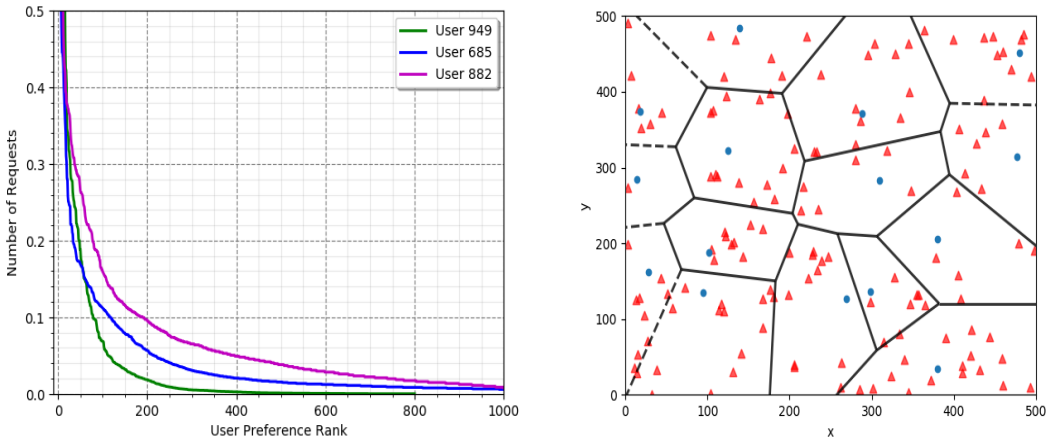
(a) (b)

Figure 4.3: (a) Comparison of content popularity vs content rank (b) Comparison of user activity level vs user activity rank of Lastfm dataset

Fig. 4.3a shows that the content popularity of Lastfm dataset can be fitted by Zipf distribution with $p_f^i = f^{-\alpha} / \sum_{n=1}^{F} n^{-\alpha}$ the skewness parameter $\alpha = 1.4$. Similarly the activity level of the users is shown in Fig. 4.3b. Activity level of the users can be fitted by Zipf distribution with $\alpha = 0.8$ for the Lastfm dataset as shown in Fig. 4.3b. The user preferences specify the number of requests for content. The top 100 users have been considered with their demands for content. The user preferences can be fitted by Zipf distribution. In Fig. 4.4a, the comparison of three users with their preferences has been shown.

4.4.1 Simulation Environment

In order to evaluate the performance of the proposed caching algorithm, the experiments have been executed based on the following settings. A square region with an area of $500m \times 500m$ is considered. The Poisson point process (PPP) has been considered for base stations in the given simulation area. The users are distributed within each base station coverage based on PPP shown in Fig. 4.4b. Five hundred contents with size determined uniformly at random from the range of 10 MB to 100 MB, 15 BSs, and 90 users have been considered. Each content has a deadline picked randomly from [10 to 30ms]. Each MEC can cache 10% of the total files. The latency to fetch content from the base station to the user is



(a) (b)

Figure 4.4: (a) Comparison of three user preferences (1st, 25th and 50th active users with user ids 949, 685 and 882 respectively) (b) Voronoi cell diagram with size $500m \times 500m$ where blue circle indicates the BSs and red triangles are mobile users.

specified using uniform distribution ranges from [5 to 25ms]. The latency to fetch content from the content server to BS is taken as the 80ms. The clusters are determined by the predicted user preferences. The number of clusters is determined using Algorithm 4.1. The simulations results presented are an average of 100 runs. The simulation parameters are shown in Table 4.2.

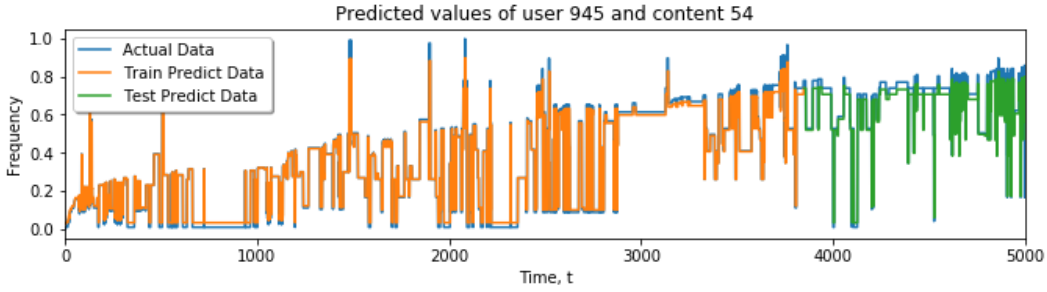


Figure 4.5: Predicted value for user 945 and Content 54

The LSTM model is used to predict the user preferences using the user activity level. The preferences predicted by the LSTM model for content 54 by user 945 are shown in Fig. 4.5. Once the model predicts the user preferences, k-means clustering (Algorithm 2) has been performed on the predicted user preferences. Then by utilizing the predicted

Table 4.2: Simulation Parameters

Parameters	**Values**
Simulation area	$500/m \times 500/m$
BS	15
Users	90
Contents	500
Content size	(10, 100] MB
Deadline of Contents	(10, 30]ms
Latency from BS to user	[5,25]ms
Latency between BSs	20ms
Latency from content server to the BS	80ms

user preferences, clustering, the content has been cached at each cluster using the proposed algorithm.

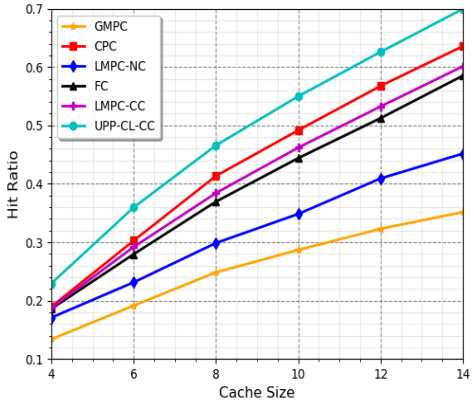
4.4.2 Performance Metrics

To compare the performance of cache placement schemes, the following metrics have been considered:

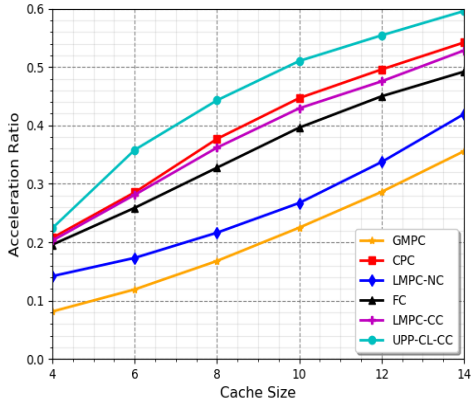
1. *Cache Hit Ratio*: The fraction of requests served over the total requests.
2. *Acceleration ratio*: The fraction of saved delay and overall delay (from the content server).
3. *Cache Utilization*: The amount of content cached in BS that the users accessed, i.e., utilization of content cached by the base station.
4. *Local Hit*: The fraction of requests served within the cluster.
5. *Neighbouring Hit*: The fraction of requests served within the network and not within the cluster.

4.4.3 Reference Algorithms

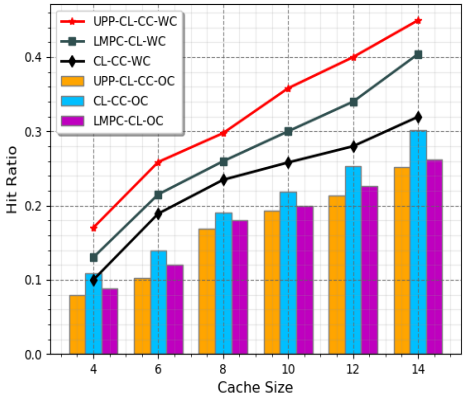
In this section, the proposed algorithm has been compared with the following caching algorithms to show the superiority of the proposed mechanism: Globally most popular caching (GMPC) [60], Locally most popular caching (LMPC) [52], Femtocaching (FC) [24], Cooperative prediction caching (CPC) [167] and Clustered cooperative caching (CL-CC) [18].



(a)



(b)



(c)

Figure 4.6: Comparison of caching schemes using cache capacity vs (a) Cache Hit Ratio (b) Acceleration Ratio (c) Local and Neighbour cluster Cache Hit Ratio.

The first two caching mechanisms cache the content without cooperation, whereas other algorithms cooperatively cache the content at MEC till the cache is full. In the fifth caching mechanism, clustering is considered, whereas the fourth algorithm employs popularity prediction in caching decisions.

4.4.4 Impact of Cache Size

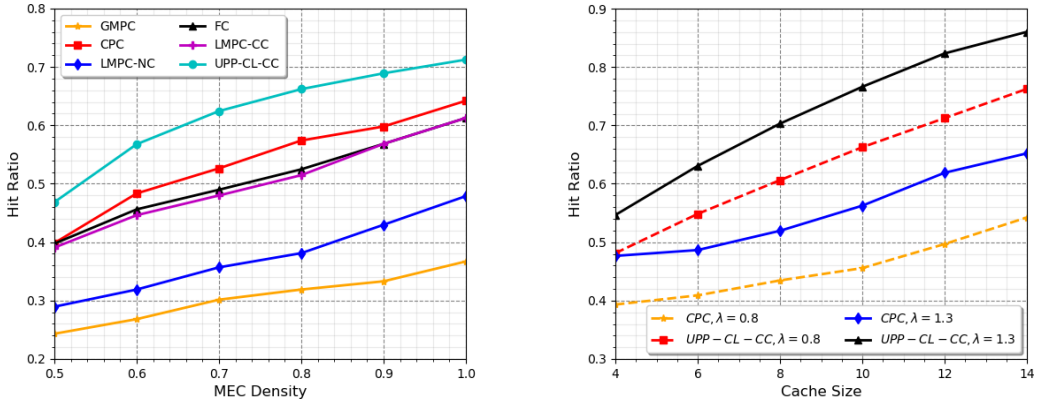
The impact of cache size on hit ratio and acceleration ratio is shown in Fig. 4.6. In this simulation, the number of MECs is 15, MEC density is 0.8, user activity level is 0.4, user preference similarity is 0.2, the number of clusters is three and the cache size varies from

4% to 14% total library size.

The effect of cache size on the cache hit rate is shown in Fig. 4.6a. The curves indicate an upward trend as the cache size grows. The cooperative caching schemes are showing superiority over non-cooperative caching schemes. It can also notice that LMPC-CC is shown to maintain a higher hit rate compared to FC. The reason is that even though both mechanisms follow a cooperative scheme, LMPC-CC caches content based on local popularity, which enables caching user preferences. It can be seen that CPC is performing better than FC and LMPC-CC due to the content popularity prediction. The proposed mechanism outperforms the other algorithms since it utilizes the user preference prediction to cache the more appropriate content at MECs.

The effect of cache size on the acceleration ratio is presented in Fig. 4.6b. The curves indicate an upward trend as the cache capacity increases. The cooperative caching schemes FC, LMPC-CC, CPC, and UPP-CL-CC, show superiority over non-cooperative caching schemes LMPC-NC and GMPC. It can be observed that LMPC-NC is shown to maintain a higher acceleration ratio than GMPC due to its local popularity. The proposed algorithm performs well compared to other algorithms because the proposed mechanism caches the predicted user preferences that permit content accessible to users with a minimal delay compared to other mechanisms. In the FC and CPC, the content is cached farther to the requested users. Even though FC and LMPC-CC mechanisms follow a cooperative scheme, LMPC shows superiority because of the local popularity.

The effect of the cache size on the local and neighbouring cluster cache hit is shown in Fig. 4.6c. The hit ratio between the clustering-based algorithms have been compared concerning cache hit rate within-cluster and outside the cluster (within the network). LMPC-CL is a clustered LMPC mechanism, WC denotes cache hit within the cluster, and OC denotes cache hit outside the cluster. The proposed UPP-CL-CC-WC mechanism is showing superiority over the other mechanisms. In the proposed mechanism, the content caches based on the predicted user preferences allow the users to access the preferred content within the cluster. It can also notice that the LMPC-CL mechanism has a higher hit rate from outside clusters than the other schemes. LMPC-CL caches local popular content at each cluster, which may differ from user preferences, leading to a lower hit rate within the



(a) (b)

Figure 4.7: Comparison of caching schemes using MEC density vs Hit ratio.

cluster and a higher hit rate outside the cluster. The CL-CC caches content based on global popularity, so all the clusters have the same set of contents, leading to a higher cache hit ratio outside the cluster. It is worth noting that the UPP-CL-CC mechanism has a higher hit ratio within the cluster and a lower hit ratio outside the cluster compared to other schemes.

4.4.5 Impact of number of MECs

The impact of the cache hit ratio on MEC density has been shown in Fig. 4.7. In these simulations, user activity level is 0.4, user preference similarity is 0.2, the number of clusters is three, the cache size varies is 10% total library size and MEC density varies from 0.5 to 1.0.

In Fig. 4.7a, the impact of the cache hit ratio is presented. The curves denoted an upward trend as the MEC density increases. It can be observed that the cooperative mechanism performs well compared to the non-cooperative mechanism because, in traditional non-cooperative mechanisms, all MECs caches the same content without cooperation. It can be seen that the CPC shows superiority over LMPC-CC and FC because CPC caches the content based on the popularity prediction. Both FC and LMPC-CC use cooperation among the MEC FC to perform better due to the greedy policy. The proposed mechanism shows superiority over other mechanisms since it uses the activity levels of users and pre-

dicted user preferences.

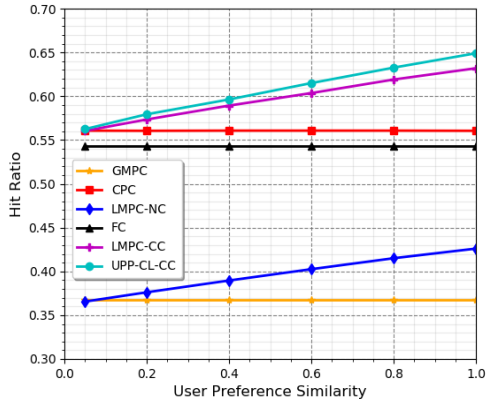
Fig. 4.7b shows the effect of cache size on cache hit rate for different MEC densities. Fig. 4.7b shows the comparison between the global content popularity based scheme (CPC), where the same content popularity is assumed between all users and the content preference predicted based scheme (UPP-CL-CC). It can be observed that the proposed prediction-based clustering mechanism shows superiority over non-clustered globally popular mechanisms. The proposed mechanism with the MEC density of 0.8 MEC/m^2, shows an increase of 20.7% in hit ratio. As the MEC density rises, the cache size of all MECs also increases allows to cache more appropriate content at MECs. Thus, the average delay in fetching the content is reduced by using user clustering. The curves show an upward trend with the growth in cache size. The benefit in hit ratio with MEC density of 1.3 MEC/m^2, shows an increase of 21.2%.

4.4.6 Impact of user preference similarity

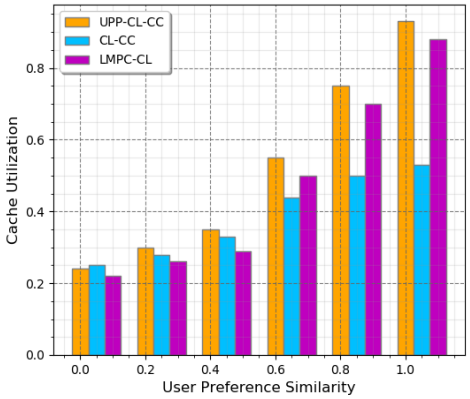
The impact of user preference similarity on cache hit ratio and acceleration ratio have been shown in Fig. 4.8. In this simulations, the number of MECs is 15, MEC density is 0.8, user activity level is 0.4, the number of clusters is three, the cache capacity is 10% total library size and user preference similarity is [0.0, 1.0] (0.0 is homogeneous and 1.0 is heterogeneous preferences).

In Fig. 4.8a, the effect of the user preference similarity on the hit ratio has been shown. The user preference heterogeneity does not affect the hit ratio for global popularity mechanisms (i.e., GMPC, FC, and CPC). Instead, the preference-based mechanisms show superiority over the global popularity mechanisms because, in the preference base mechanisms, the heterogeneity of user preferences allows the mechanisms to cache the appropriate content, leading to a higher cache hit ratio. The preference-based mechanisms show an upward trend as the user preference similarity increases. Furthermore, it can notice that when user preference similarity (heterogeneous content) is more, the proposed mechanism has better performance than other preference-based mechanisms (i.e., LMPC-CC and LMPC-NC).

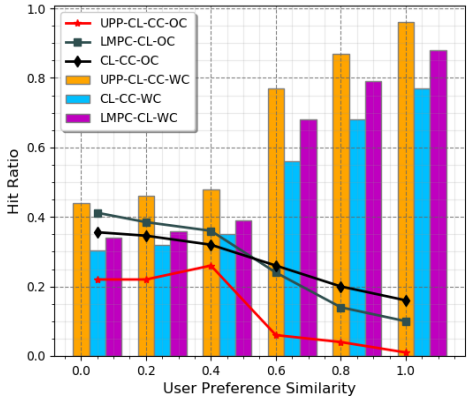
The effect of user preference similarity on cache utility is shown in Fig. 4.8b. It can



(a)



(b)



(c)

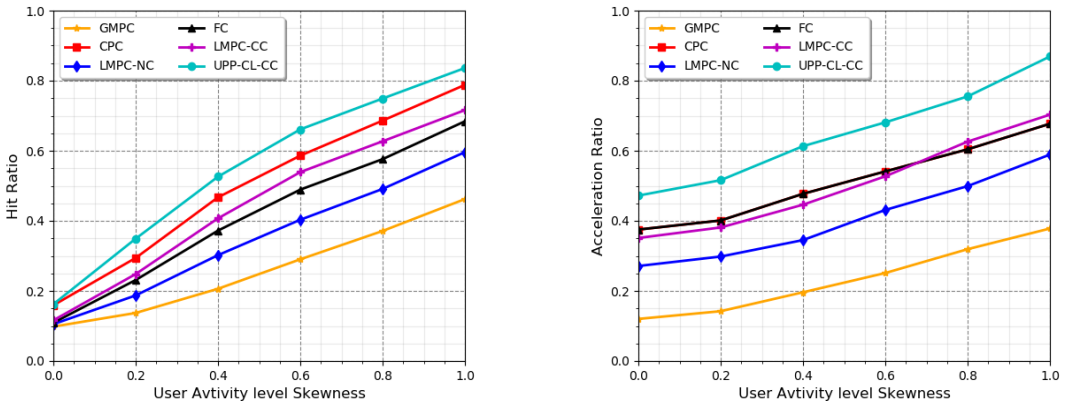
Figure 4.8: Comparison of caching schemes using user preference similarity vs (a) Cache Hit Ratio (b) Cache Utilization (c) Local and Neighbour cluster Cache Hit Ratio.

be seen that the preference-based mechanisms have better performance than the global popularity based mechanism. As the heterogeneity of user preferences increases, global popularity has more deviation, whereas the local popularity and predicted mechanism has the user preferred content within the cluster. Further, it can notice that CL-CC has good performance when the preference similarity is low and nearly flatten when the similarity increases.

In Fig. 4.8c, the effect of the user preference similarity on the local and neighbouring cluster cache hit is shown. The clustering-based algorithms concerning cache hit rate within-cluster and outside the cluster (within the network) has been compared. LMPC-CL

is a clustered LMPC mechanism, WC denotes cache hit within the cluster, and OC denotes cache hit outside the cluster. The proposed mechanism (i.e., UPP-CL-CC) has a higher cache hit ratio within the cluster because of predicting the user preferences and caching the appropriate at each MEC in the cluster. The CL-CC has the lowest among the clustered caching mechanisms since caching the content based on global content popularity differs from the user's preferred content. Further, it can be noticed that the proposed mechanism has the lowest hit rate in the outside cluster because most of the user demands are satisfied by the MECs within the cluster. It can be observed that both the user preference based caching mechanisms have a higher hit rate when the user preference similarity is low and increases as the heterogeneity increases. CL-CC-WC has a lower hit rate when the preference similarity is less and grows as the similarity rises. The reason is that MECs in CL-CC cache the globally popular content that may not satisfy the user preferences.

4.4.7 Impact of User activity level skewness



(a) (b)

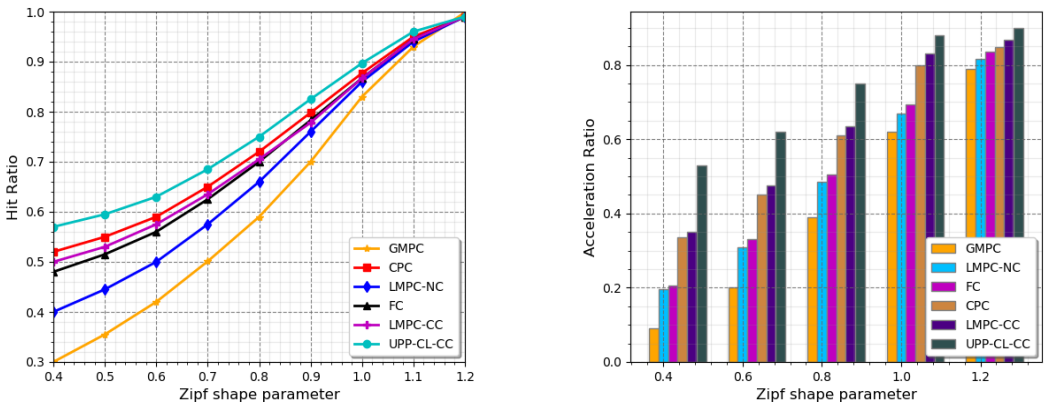
Figure 4.9: Comparison of caching schemes using User activity level skewness vs (a) Cache Hit Ratio (b) Acceleration Ratio.

The impact of user activity level skewness on acceleration ratio cache hit ratio has been shown in Fig. 4.9. In this simulation, the number of MECs is 15, MEC density is 0.8, user preference similarity is 0.2, the number of clusters is three, the cache size is 10% total library size and user activity level varies from 0 to 1 with step size 0.2.

The effect of the cache hit ratio with different user activity levels has been shown in Fig. 4.9a. It can see that all the curves indicate an upward trend as the user activity level skewness increases. The reason is that the more skewed user activity level means the users are more actively requesting content. Since the users are highly active, the caching mechanisms can cache those active users' preferences, leading to an increase in the average rate of these active users. The proposed mechanism is shown superiority over other mechanisms since it considers users' preference similarity and activity level.

In Fig. 4.9b, the impact of user activity level on acceleration ratio is presented. The highly skewed user activity level allows the caching mechanisms to cache highly active user preferences. Thus the saved delay is increased as the user activity level increases. It can be seen that the FC is performing almost the same as CPC. LMPC-CC performs well as the activity level increases compared to FC and CPC because of the local content popularity, which captures the user preferences.

4.4.8 Impact of Zipf parameter



(a) (b)

Figure 4.10: Comparison of caching schemes using Zipf shape parameter vs (a) Cache Hit Ratio (b) Acceleration Ratio.

The impact of content popularity on acceleration ratio and cache hit ratio over the different caching mechanisms is shown in Fig. 4.10a and 4.10b. It can be seen that the increase in the performance of the proposed mechanism as the Zipf shape parameter α increases.

The proposed caching mechanism always outperforms other caching mechanisms, and as the α increases (means content requests are more concentrated), the gap between the proposed and other algorithm decreases.

4.4.9 Impact of Number of clusters

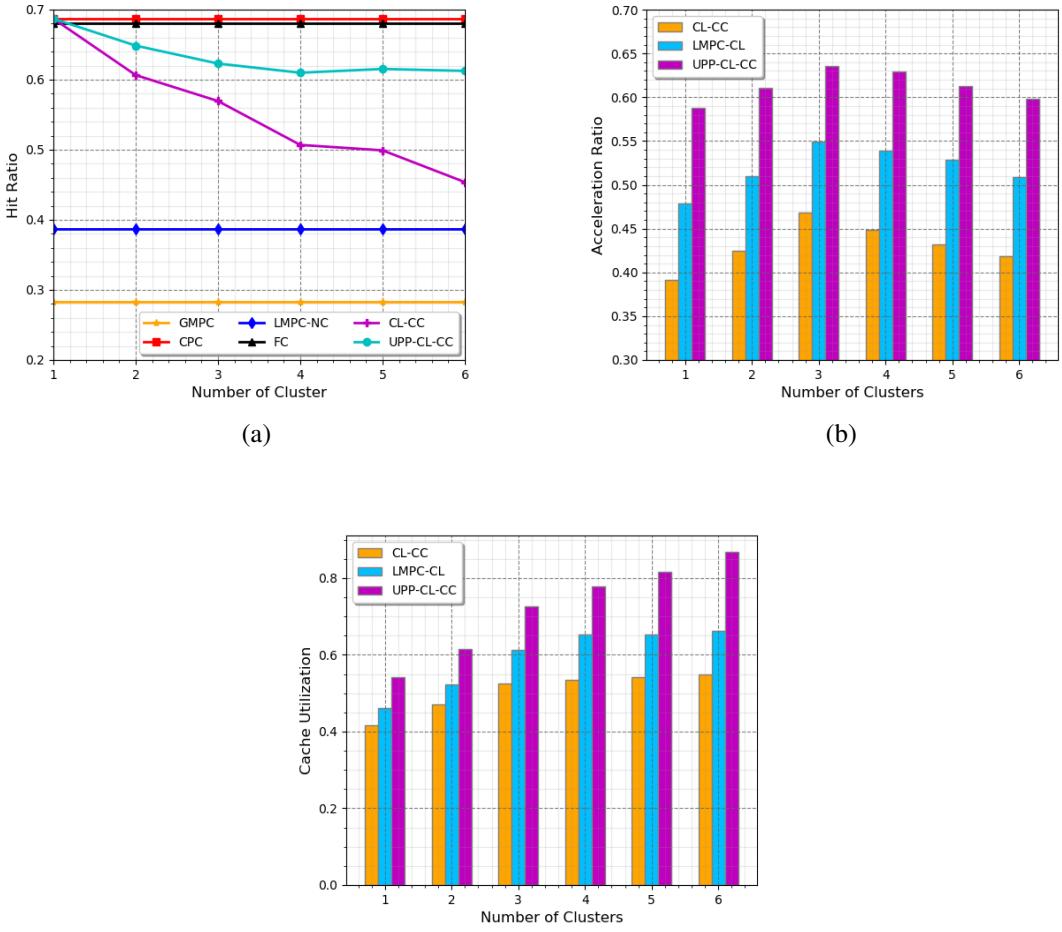


Figure 4.11: Comparison of caching schemes using Number of clusters vs (a) Cache Hit Ratio (b) Acceleration Ratio (c) Cache Utilization.

The impact of the number of clusters on cache hit ratio, acceleration ratio, and cache utilization has been shown in Fig. 4.11. In this simulation, the number of MECs is 15, MEC density is 0.8, user preference similarity is 0.2, the cache size is 10% total library

size, user activity level varies is 0.4, and the number of clusters varies from 1 to 6 with step size 1.

Fig. 4.11a shows the effect of the number of clusters on the cache hit rate. It can be seen that the number of clusters has no effect on the cache hit rate of the non-clustered mechanisms like GMPC, LMPC-NC, FC and CPC. The non-clustered cooperative caching mechanisms FC and CPC are constant and better than CL-CC and UPP-CL-CC. The reason is that the non-clustered mechanisms cooperatively cache the content at MECs, whereas the clustered mechanisms cache the redundant content among different clusters. It can be seen that the proposed mechanism starts decreasing as the number of clusters increasing and remains constant, whereas the CL-CC (cluster only) declines as the number of clusters rises. Since the cluster-only mechanism caches globally popular content redundantly among the clusters, the proposed mechanism caches the predicted user preferences in different clusters, leading to more hit rates than CL-CC. However, the proposed mechanism loses the hit rate of 8% compared to CPC and FC.

In Fig. 4.11b, the impact of the number of clusters on the acceleration ratio is presented. Three clustering mechanisms has been considered, namely CL-CC (clustered cooperative caching with global popularity), LMPC-CL (clustered cooperative caching with local popularity) and UPP-CL-CC (user preference predicted clustered cooperative). It can be seen that the acceleration ratio increase till it reaches an optimal number of clusters and also notices a downward trend as the number of clusters rises after the optimal cluster size. It can be observed from fig. 4.11b, with cluster size one, the acceleration ratio is low because with one cluster, MEC density is more (i.e., more number of MECs present in each cluster), thus fetching content from long distance MECs leads to more delay. As the number of clusters increases, redundant content is cached among the clusters, leading to less diversity of content. It can be observed that the LMPC-CL performs well compared to CL-CC because LMPC-CL caches the content based on local popularity, whereas CL-CC caches based on global popularity. Thus the LMPC-CL allows the clusters to cache relatively user-preferred content. The proposed mechanism outperforms the other mechanisms since it caches the predicted user preferences leading to cache appropriate content that reduces the likelihood of obtaining content from the content server.

The effect on cache utilization with a varying number of clusters has been shown in Fig. 4.11c. The cache storage utilization of the proposed mechanism shows superiority over the other two mechanisms. The reason is that each MEC in each cluster caches the user preferred content leads to higher utilization of cache storage. CL-CC has less cache utilization than LMPC-CL because it caches the same redundant content among the clusters based on global popularity, differing from the user-preferred content. Even though the proposed mechanism loses hit ratio, it can be noticed that it achieves a better acceleration ratio and cache utilization from the above results.

4.5 Summary

In this chapter, a clustered cooperative cache placement has been analyzed in large-scale mobile edge networks, aiming to maximize the saved delay by considering the heterogeneity of user preferences, activity level, and uneven user distribution. The LSTM model has been considered to capture the dynamics of user activity and preferences. Content-based clustering is used to group the MECs using K-means clustering and an efficient greedy approach has been proposed to solve the cache placement problem. Simulation results illustrate the relation between user preferences and local and global content popularity. It has been observed that there is a significant performance gain in mobile edge network for cache placement decision by exploiting the individual user behavior with the realistic setting, such as higher user preference similarity, skewed user activity level distribution, and unevenly distributed users. Simulation results show that the proposed content placement mechanism improves up to 18, 21, and 23 percent on cache utilization, acceleration ratio, and hit ratio over existing algorithms. The next chapter presents a contact duration-aware cooperative cache placement strategy using a genetic algorithm by considering the user's mobility.

Chapter 5

Contact Duration-Aware Cooperative Cache Placement with User Mobility Across MECs using Genetic Algorithm for Mobile Edge Networks

Most of the existing works [11, 60, 24] focus on caching content cooperatively at BS for static networks. This assumption made by the existing works [11, 60, 24] is unrealistic in a dense network. This chapter considers the cache placement problem in a realistic scenario where the users with different speeds intermittently connect to the BSs at irregular intervals. The users will frequently move between BSs and can download only parts of the requested content from different encountered BSs along the moving path. If the user fails to download the complete content from encountered BSs, then the requested content is downloaded from a macro base station (MBS), this in turn increases the overall delay and affects the QoS. Consider an example; customers move around a shopping mall with three BSs. If a user wishes to download content, then the content should be replicated in all BSs. Replicating the same content parts (segments) at three BSs is a wastage of resources, so disjoint content parts should be cached at the BSs. Hence, the caching mechanism should consider the user mobility pattern. Although [21, 22] assuming the user mobility, the randomness of

contact duration is not considered in content placement decision. According to [23], data transmission is associated with contact duration (sojourn time). If the contact duration is short, the user moves at high speed and if the contact duration is long, it means the user moves at low speed. Thus, contact duration randomness caused by user mobility affects the transmission of data and this in turn affects the content placement. Therefore, this work aims to design caching methods by considering content popularity, the randomness of contact duration (speed of the user) and user mobility along with content popularity and resource limitation.

This chapter presents a content placement mechanism for dynamic networks where the moving users intermittently connect to the BSs at irregular intervals of time. User mobility is modeled as a Markov renewal process to predict contact duration and user moving paths. Then the contact duration aware content placement is designed by formulating the maximum saved delay problem. For the contact duration aware content placement problem, a submodular function with matroid constraints can maximise saved download delay. Further, a heuristic search mechanism has been designed based on a genetic algorithm to efficiently obtain content placement solutions for large scale problems (the scenarios that scale to large video library sizes).

The contributions of this chapter are as follows:

- Formulation of a mixed integer non linear programming problem for contact duration aware content placement problem: maximization of saved download delay subject to constraints, namely cache capacity and popularity of the content in mobile edge networks.

- Modeling user mobility as a Markov renewal process to predict the user moving paths and contact duration.

- Design of a greedy algorithm by adopting submodular optimization to solve the problem and development of a heuristic search mechanism based on a genetic algorithm to solve the content placement problem for large scale problems efficiently.

- Extensive simulations have been performed to show the efficacy of proposed algo-

rithms with different parameters, including cache hit ratio and acceleration ratio using real-world data sets.

The rest of the chapter is organized as follows. In Section 5.1, system model, motivation and formulation of the contact duration aware content placement problem are presented. A greedy approximation algorithm for the proposed problem is presented in Section 5.2. Then, a genetic algorithm-based heuristic caching algorithm is presented in Section 5.3. Simulation environment and results are presented in Section 5.4. A summary of this chapter is mentioned in Section 5.5.

5.1 MEC System Model and Problem Formulation

In this section, the network model, mobility model, content request model along with motivation and problem formulation are presented in detail.

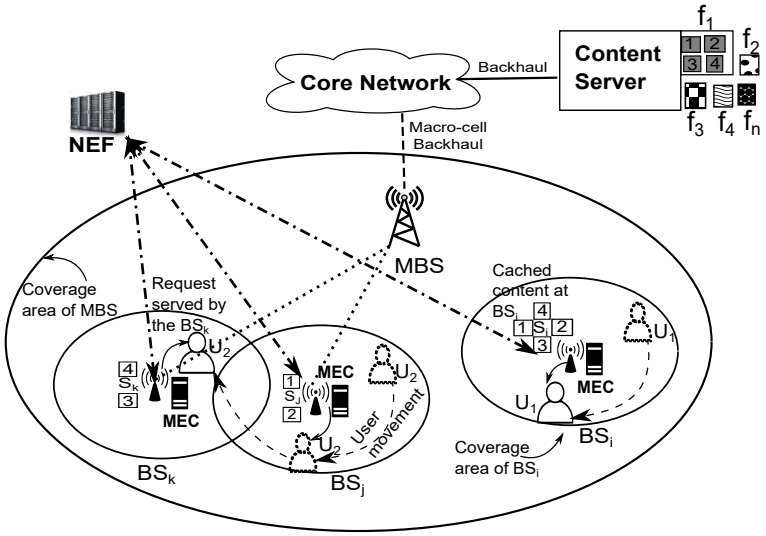


Figure 5.1: Illustration of the proposed system model.

5.1.1 Network Model

Mobile edge computing improves users' capabilities by providing cache capacity (i.e., storage), network resources and computing power near to users. Consider a mobile edge network containing a macro base station, a set $\mathcal{R}$ of R small base stations (BS) equipped with

MEC server, a set $\mathcal{U}$ of U mobile users, a content server and a central coordinator NEF as shown in Fig 5.1. Each MEC $r \in \mathcal{R}$ has a limited cache S_r called local storage and transmission capacity β (bandwidth). The storage of each MEC is used for content caching and β_r represents the amount of data transmitted between MEC r and a mobile user u at a time slot. There is a set of locations (such as an educational hub, commercial area, sports city, financial district and housing) in the macro cellular area. The MECs are connected and also to the core network through backhaul links. The content server acts as an origin server that stores all contents. Network Exposure Function (NEF) serves as a coordinator (it is a crucial network element in 5G networks) [79, 151]. NEF maintains the indexes of the content cached at individual MECs and monitors users' content requests at each MEC [151]. A user is directly connected to the base station, and the user may be in the communication range of more than one BS at any point in time. However, any user can communicate with only one BS at a particular time. Users may move across different base stations, so in a different time, the user connects with different BSs.

5.1.2 Mobility Model

The requested content is transmitted successfully between the mobile user and BS when the user is in communication range of BS. Time split into segments and each time slot is denoted by T. The user moves from one BS to another in a time slot T, leading to user content requests uncertainty. The mobility of user u is modeled as *Markov renewal process* [84, 168] $\{(X_v, T_v) : v \geq 0\}$ to predict the user moving path and MEC contact duration (sojourn time), where $X_v \in \mathcal{R}$ is v^{th} transition state and T_v is the v^{th} transitions time instant. The base station sojourn time is defined as the time period a user served by a specified BS, which influence the amount of data obtained from the BS. The sojourn time is denoted as δ_u^r. The sojourn time is estimated using the user moving statistics. P_u is the transition probability of Markov chain for user u. The distribution of time that the semi Markov process of user u spend at BS r before making a transition is denoted as H_u^r.

5.1.3 Content Request Model

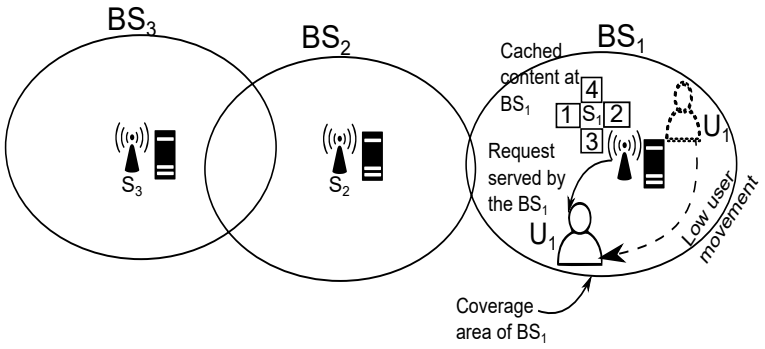
Consider a set $\mathcal{F}$ of F contents of length B bits in the content library located in content server. Assume the demand for content is known already and content f is requested with p_f probability. Given limited contact duration, complete content cannot be transmitted. The simple content fragmentation may considerably decrease data access efficiency [169]. To improve the cache efficiency and reduce the redundant storage of content at BS, the Maximum Distance Separable (MDS) code is adopted [169, 170, 86]. In MEC server local storage, the encoded segments of the contents are cached instead of original content. The content splits into multiple encoded segments. The number of encoded segments (of each content) that need to be cached at each MEC needs to be determined. User has to collect at least B bits to get the original content.

Mobile users are attached to the base stations according to a cellular network protocol. The connected base stations are accountable for serving user requests. Upon receiving a request from a user, the connected MEC server checks its local storage for the content. If the requested content exists in the local storage, then the MEC serves the request immediately. This improves the user QoE by reducing the download delay. Moreover, no extra burden is added on backhaul links reducing network traffic. Otherwise, the content is fetched from the MBS. If the user moves from one BS to others, the user continues downloading content from the corresponding BS.

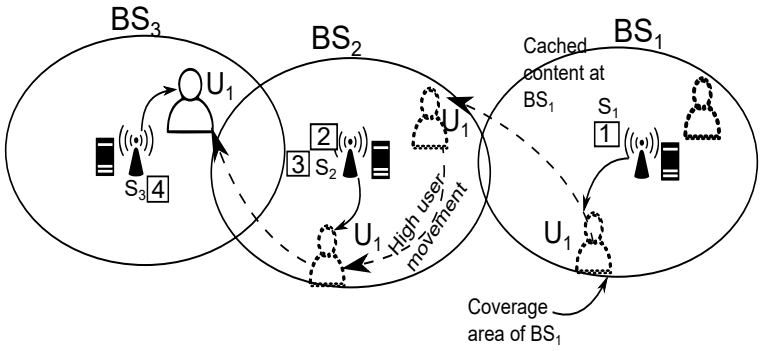
5.1.4 Motivation

To show the need for the contact duration aware content placement using user mobility, consider the scenario depicted in Fig. 5.2a, 5.2b with a user U_1, a file f and three MEC servers with limited cache capacity.

In a system without mobility, the requested content is served by the corresponding BS. In real-world, the user moves across different locations. Due to mobility, user pass through different MECs and this affects the optimal content placement. User cannot be served by only one base station because of movement. Different MECs serve the content to the user based on the contact. To serve the user request the same content need to be stored



(a)



(b)

Figure 5.2: Illustration of user mobility speed (a) Low mobility movement (b) High mobility movement.

in different MECs, leads to inefficient use of MEC storage. Therefore, to utilize the MEC storage efficiently code the content and store the encoded content instead of raw content. Each content divided into different segments. Then, the encoded segments need to place at different MECs based on the user trajectory. However, if the contact duration is taken into consideration, then the optimal content placement changes. The number of contacts with multiple MECs will be very small if the user moves slowly where as the contact duration with the connected MEC will be larger. Consider Fig. 5.2a, user U_1 requests a content f of four segments. The user gets four segments from S_1. In this case, the codes are cached at MEC_1. In Fig. 5.2b, user U_1 requests a content f with four segments. The user moves with high speed this time. Therefore, the user obtains only one segment from MEC_1, two segments from MEC_2 and one segment from MEC_3. In the second scenario, the user obtains the content from three different MECs due to high movement. Therefore, placing the content at different MECs becomes difficult because the three MECs provide

the requested encoded segments based on the user moving rate. Hence, the last two cases indicate the importance of contact duration and user mobility in cache placement schemes.

5.1.5 Static and mobility aware caching scenarios

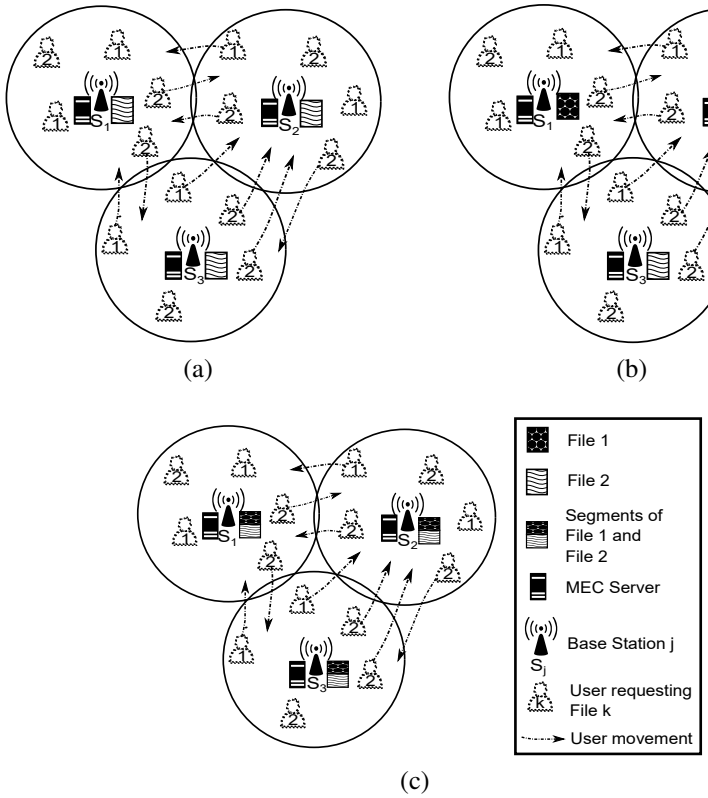


Figure 5.3: Illustration of caching scenarios for static and mobility cases (a) Static / MAUC (*case 1*) (b) MAUC (*case 2*) and (c) MACC scenarios.

The most popular content (MPC) mechanism is a heuristic caching mechanism that caches the content based only on global popularity distribution irrespective of user mobility [60]. A globally more popular content may not be popular among the users associated with a specific BS. The variety (different types) of content is not cached in BSs at different locations as each BS caches the globally popular content [21, 22, 34]. The proposed mechanism considers user demand, the information of user mobility pattern and places MDS (maximum distance separable) encoded content segments at the BSs instead of the entire content. The proposed mechanism makes informed caching decisions by considering user mobility. In the worst case (all the users move across the BSs), the proposed mechanism

caches multiple encoded segments of the same content at all BSs, which differs from MPC. The MPC caches the complete content, whereas the proposed mechanism caches the MDS encoded content allows the BSs to cache different types of coded content.

To illustrate the efficacy of the proposed mechanism over MPC, consider an example shown in Fig. 5.3. Users move around a shopping mall with three BSs and each BS is provided with unit-sized storage. Within a deadline of 2-time slots, mobile user contacts (users come into the coverage of BS) uniformly at random two base stations while moving. During the contact time, at the most, half of the data file can be transmitted. There exist four files (F1, F2, F3, F4) of size one unit each. Assume that the users, as indicated in Fig. 5.3, request content K from the corresponding BS (in Fig. 5.3 the user with k represents user requesting k^{th} content, i.e., a user with 1 requesting $F1$). Consider the content request pattern $\{F1, F2, F2, F1, F2\}$ at each BS is identical unless specified. The proposed mechanism's efficacy over MPC is demonstrated using two metrics (Network overhead and Hit rate). Table 5.1 shows a summary of the findings. Network overhead is defined as the total transmission cost (in terms of hops and assumes the unit cost is 1 per hop) of all requested content. The hit rate is expressed as the hit percentage of all the user requests on the BSs. In this example, consider three cases to show the efficacy of the proposed mechanism over MPC, as shown in Fig. 5.3.

Table 5.1: Hit ratio and network overhead for caching scenarios

Metric	**Static**	**MAUC**	**MACC**
Network overhead	12	12	0
Hit ratio (%)	60	60	100

Static scenario: In a static scenario, the mobile users remain static within a BS where the corresponding BSs can serve the users. Consider the case when users at a BS have repeatedly requested two files (F1, F2). Therefore, either F1 or F2 will be cached at each BS and the other file will be fetched from MBS (macro base station). In this case, each BS caches the file F2 as shown in Fig. 5.3a, which is globally popular (MPC) (consider the scenario shown in Fig. 5.3a without user mobility). Since each BS caches F2, all the requests for file F1 will be fetched from MBS. Therefore, network overhead is computed as the cost of the number of requests missed at each base station. By considering the number

of missed requests (F1) as 2, the number of hops as 2 and number of BSs as 3, the network overhead is computed as 12, i.e., $2 * 2 * 3 = 12$. Further, by considering the number of hit requests as 9 and total requests as 15, the hit ratio is computed as 60%, i.e., $9/15 * 100 = 60\%$.

Mobility aware uncoded caching scenario (MAUC): In the MAUC scenario, users move across BSs; the caching decisions are based on user demand and mobility patterns. *Case 1:* Assume that all the BSs cache popular file F2, as shown in Fig. 5.3a. On the moving path, the user requested for file F2 can be download from the corresponding BSs (i.e., a user gets half of the file from each BS as the user comes into contact with two BSs at different time slots). Since each BS stores F2, all the requests for file F1 will be fetched from MBS. By considering the number of missed requests (F1) as 2, the number of hops as 2 and number of BSs as 3, the network overhead is computed as 12, i.e., $2 * 2 * 3 = 12$. Further, by considering the number of hit requests as 9 and total requests as 15, the hit ratio is computed as 60%, i.e., $9/15 * 100 = 60\%$. *Case 2:* Assume that one of the three BSs (BS1) caches F1 (F1 has more demand at BS1), and other BSs cache file F2 as shown in Fig. 5.3b. The users moving across BS2, BS3 within the deadline of 2-time slots (by considering the use mobility pattern) can download the file F2 successfully. All other user requests will be forwarded to MBS since half of each file downloaded from the BSs that came across.

Mobility aware coded caching scenario (MACC): In the MACC scenario, users move across BSs; the caching decisions are based on user demand and mobility patterns. In this scenario, each BS caches half of the files' (F1, F2) encoded data, as shown in Fig. 5.3c. Therefore, users will get half of the files' encoded data from each BS irrespective of the files requested by the users. Users can recover the requested file by downloading the encoded data at least equal to the requested file size. Since each BS stores F1, F2 encoded segments, the requested file can be downloaded from BSs. By considering the number of missed requests (F1) as 0, the number of hops as 1 and number of of BSs as 3, the network overhead is computed as 0, i.e., $0 * 1 * 3 = 0$. Further, by considering the number of hit requests as 15 and total requests as 15, the hit ratio is computed as 100%, i.e., $15/15 * 100 = 100\%$.

The three cases show that in the worst case (if each user path is independent and multiple users are requesting content), the mobility aware coded caching scheme caches the encoded segments at each BS. This encoded caching scheme enables the BS to store different types of content in the worst case. These scenarios show that the proposed mechanism has a notable advantage over the reference algorithms in the worst-case scenario. List of notations used in this chapter are presented in Table 5.2.

Table 5.2: List of Notations

Term	**Definition**
$\mathcal{R}$	Set of base stations coupled with MEC servers
$\mathcal{F}$	Set of contents
$\mathcal{U}$	Set of mobile users
S_r, β_r	The cache, transmission capacity of r-th MEC
B_f	The size of f-th content
$t \in T$	Time slot
d	Deadline
τ	Delivery time of content
δ_u^r	Contact duration of user u at MEC r
μ_u^r	Average time user u stays at MEC r
X_v, T_v	v^{th} transition state and time
$m_u^v \in M_u$	v^{th} path of user u
$\mathcal{T}_u^{v,i}$	i^{th} transition of v^{th} path of user u
H_u^r	Probability density function of sojourn time
α_u^r	Number of appearances of MEC in v
p_f	Probability that content f is requested
P_u^v	Probability that user u taking path v
ψ_f^{av}	The average amount of f downloaded by a user
d_u^r, d_u^m	The delay for transmitting unit of coded content to mobile user u from BS, MBS
x_f^r	Number of coded segments of f cached at MEC r
$y_f^{r,l}$	Useful content f downloaded from MEC r in l^{th} contact
$P(n)$	Genetic population
$n_{f,k}^r$	k^{th} segment of content f cached at MEC r
$\mathcal{M}$	Matroid with finite ground set N
c_p, m_p	Crossover and mutation probabilities
N_{pop}, gen	Population size and number of iterations

5.1.6 Mobility and sojourn time prediction

Mobile user moves from one state (MEC) to another state over time T. The time duration of a user resides in the state r is not known. The sojourn time of the user u resides in state r is predicted using *Markov renewal process* [84, 168]. To predict the sojourn time of user, average time is computed as

$$\mu_u^r = \int_0^\infty x H_u^r(x) dx \tag{5.1}$$

where $H_u^r(x)$ is the probability density function of sojourn time for user u at MEC (state) r. i.e., the user u likely to move from one state to another state (transition) by staying average time (μ_u^r) duration at state r.

User may move in multiple paths. The user u moving along a path in time T is defined as $M_u = \{m_u^1, m_u^2, \ldots m_u^v\}$ where $v \geq 1$. The initial state of the path v is represented as $m_u^{v,0} = m_u^v$. i.e., the initial state of path v is initial state for all paths of user u. Therefore, the v^{th} path of user u is denoted as $\{m_u^{v,0}, m_u^{v,1}, \ldots, m_u^{v,d}\}$ where $d \geq 0$. In a particular path, a user may move to the same MEC multiple times, therefore the user path m_u^v is multi-set. The user resides in initial state for t_u^0 and moves to 1^{st} state. For all paths the 1^{st} transition is predicted to happen at time instant

$$\mathcal{T}_u^{v,1} = \int_0^\infty x H_u^{m_u^{v,0}}(x) dx \tag{5.2}$$

where $\mathcal{T}_u^{v,0}$ is 0 and $\mathcal{T}_u^{v,1}$ is same between all the paths of a user u. From this, the v^{th} path i^{th} transition is predicted as

$$\mathcal{T}_u^{v,i} = \mathcal{T}_u^{v,i-1} + \mu_u^{m_u^{v,i-1}} \tag{5.3}$$

where $i \leq d$. The last transition occur before the end of time instant T.

The sojourn time of v^{th} path m_u^v is represented as $\{\delta_u^v = \delta_u^{m_u^{v,0}}, \delta_u^{m_u^{v,1}}, \ldots, \delta_u^{m_u^{v,d}}\}$. The sojourn time is $\delta_u^{m_u^{v,i}} = \mathcal{T}_u^{v,i} - \mathcal{T}_u^{v,i-1}$ and delivery time of content is considered as τ.

Therefore, the sojourn time is derived as

$$\delta_u^{m_u^{v,i}} = \begin{cases} [\mathcal{T}_u^{v,1} - \tau], & i = 0, \\ [\mu^r_{m_u^{v,i}} - \tau], & 1 \leq i \leq d-1 \\ [\mathcal{T} - \sum_{j=0}^{d-1} \delta_u^{m_u^{v,j}} - \tau], & i = d \end{cases} \tag{5.4}$$

where $x = [a]$ encloses the value $x = 0$ when $a < 0$ and $x = a$ while $a \geq 0$. The probability that user u taking path m_u^v in time T is denoted as

$$P_u^v = \prod_{i=0}^{d-1} P_{m_u^{v,i}, m_u^{v,i+1}} \tag{5.5}$$

The average sojourn time of user u at state r in time T is represented as

$$\delta_u^r = \sum_{v \in \mathcal{V}} P_u^v \sum_{j=0}^{d} \delta_u^{m_u^{v,i}} \, 1_{(m_u^{v,j} = r)}, \qquad r \in \mathcal{R} \tag{5.6}$$

5.1.7 Problem Formulation

The caching scheme of coded segments in MEC is denoted as $X_{r \times f}$, $x_f^r \in X$ indicates the number of encoded segments of content f cached in MEC r. Due to user mobility, the user may experience communication with the same MEC many times. Therefore, caching the same content at different base stations in the user path is wastage of resources (storage). The useful content downloaded by a user during the first contact with MEC r is denoted as

$$y_f^{r,1} = \min \left\{ x_f^r, \delta_u^r \frac{\beta_r}{B_f} \right\} \tag{5.7}$$

Downloading the useful content f by user during the l^{th}contact ($l \in \{2, 3, \ldots, \alpha_u^r\}$) with r^{th} MEC is $x_f^{r,l} = x_f^r - \sum_{t=1}^{l-1} y_f^{r,t}$ represented as

$$y_f^{r,l} = \min \left\{ x_f^{r,l}, \delta_u^r \frac{\beta_r}{B_f} \right\} \tag{5.8}$$

The successful download of a content means that the overall coded segments downloaded by MEC must satisfy at least the size of the total number of encoded segments from the requested content. i.e., overall segments downloaded $\leq$ total segments. A user proceeds with path v by requesting content f, the overall content downloaded by the encountered base stations is $\psi_f = \sum_{r\in R}\sum_{l\in\alpha_u^r} y_f^{r,l}$. Then by considering contents and mobility paths, ψ_f can be written as $\psi_f^{av} = \sum_{f\in\mathcal{F}} p_f \sum_{v\in\mathcal{V}} P_u^v \psi_f(d_u^m - d_u^r)$. ψ_f^{av} is the average amount of coded data downloaded from the BS by a user. Therefore, the coded data downloaded from MEC by average number of users is represented as

$$\psi_f^{av} = \frac{1}{U}\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} p_f \sum_{v\in\mathcal{V}} P_u^v \psi_f(d_u^m - d_u^r) \tag{5.9}$$

By changing the order of summation

$$\psi_f^{av} = \frac{1}{U}\sum_{r\in R}\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} p_f \sum_{v\in\mathcal{V}} P_u^v (d_u^m - d_u^r) \sum_{k\in\alpha_u^r} y_f^{r,k} \tag{5.10}$$

Definition 5.1.1 (Saved delay)**.** *The saved delay defined as the difference between the download delay from the macro base station and the small base station (MEC servers).*

Aim of this study is to find the caching scheme that maximizes the overall saved delay of requested contents. The problem is modeled as follows:

$$\max\ \psi_f^{av} \tag{5.11}$$

s.t.

$$\sum_{f\in\mathcal{F}} B_f \cdot x_f^r \leq S_r, \qquad \forall r\in\mathcal{R} \tag{5.12}$$

$$x_f^r \in \{0, 1, \ldots, B\}, \qquad \forall r\in\mathcal{R}, f\in\mathcal{F} \tag{5.13}$$

The objective (5.11) is the total saved delay caused by users of the overall network. The constraints (5.12) provide the finite capacity of each base station. i.e., the total quantity of content placed in the storage of MEC should not exceed the capacity of MEC. The

constraints (5.13) is the non-negativity constraint of the decision variables.

5.2 Greedy Algorithm for Contact duration Aware Cooperative Content Placement

The content placement problem presented in Eq. (5.11) is a mixed integer non-linear programming (MINLP) problem and proved as NP-hard [21, 22]. To solve the problem presented in (5.11), a sub-optimal greedy algorithm has been designed using submodular optimization [166]. The greedy algorithm designed for the given constraints are matroid constraints, and the objective function is monotone submodular. Greedy approximation algorithm gives $\frac{1}{2}$ approximation in the worst case [166]. A greedy algorithm has been presented by reformulating the problem presented in Eq. (5.11) into a monotone submodular function with matroid constraints.

Definition 5.2.1 (Submodular function)**.** *Let N be a finite ground set and* $g : 2^N \rightarrow \mathbb{R}_+$ *is submodular if the following properties are satisfied:*

1. $g(A) + g(B) \geq g(A \cup B) + g(A \cap B)$, *for all* $A, B \subseteq N$.
2. $g(A) \leq g(B)$, *for all*$A \subseteq B \subseteq N$.

Equivalently, g is said to be a monotone submodular function if the following condition satisfies. Let $g_A(j) = g(A + \{j\}) - g(A)$. *Here,* $g_A(j)$ *indicates the marginal value of an element* $j \in N$ *with respect to a subset* $A \subseteq N$.

$$g_A(j) \geq g_B(j) \geq 0, \text{ for all } A \subseteq B \subseteq N \text{ and } j \in N - B. \tag{5.14}$$

The intuition of monotone submodular function is that the benefit of adding a new element decreases when the set becomes large.

Definition 5.2.2 (Matroid)**.** *A tuple* $\{N, \mathcal{I}\}$ *is called a matroid* $\mathcal{M}$*, if* N *is a finite ground set and* $\mathcal{M}$ *is a nonempty collection of subsets of* N *which satisfies the following conditions:*

1. $\phi \in \mathcal{I}$, *i.e.,* $\mathcal{I}$ *is nonempty.*

2. *If $B \in \mathcal{I}$ and $A \subseteq B$ then $A \in \mathcal{I}$.(downward closed)*

3. *If $A, B \in \mathcal{I}$ and $|A| \leq |B|$, then $\exists\, j \in B - A$ such that $A \cup \{j\} \in \mathcal{I}$.*

Define the ground set $N = \{n^r_{f,k} \mid f \in \mathcal{F}, r \in \mathcal{R}, k = \{1, 2, \ldots, B\}\}$ and $A \subseteq N$ is the cache placement scheme. $n^r_{f,k} \subseteq A$ represents that the k^{th} segment of content f is cached at MEC r. Let $N^r_f = \{n^r_{f,k} | k = 1, 2, \ldots, B\}$ represents that all B segments of content f is cached at MEC r. The relationship between cache placement A and x^r_f is

$$x^r_f = |A \cap N^r_f| \tag{5.15}$$

Then, the original objective function (5.11) can be rewritten as:

$$g(A) = \frac{1}{U} \sum_{u \in \mathcal{U}} \sum_{f \in \mathcal{F}} p_f \sum_{v \in \mathcal{V}} P^v_u \sum_{r \in R} \min \left\{ |A \cap N^r_f|, \delta^u_r \frac{\beta_r}{B_f} \right\} (d^m_u - d^r_u) \tag{5.16}$$

If $g(A)$ satisfies the property of monotone submodular function (5.14), then the function $g(A)$ is said to be monotone submodular function. To prove submodularity of function $g(A)$ we prove $g(A \cup n^r_{f,k}) - g(A) \geq g(B \cup n^r_{f,k}) - g(B) \geq 0$ where $A \subseteq B \subseteq N$ and $n^r_{f,k} \in N - A$.

Lemma 5.2.0.1. *The function in* (5.16) *is a monotone submodular function.*

Proof. See Appendix for proof of lemma 1. □

Next, The constraints present in the problem (5.11) can be written as matroid constraints on N.

Lemma 5.2.0.2. *Let N_r, where $r \in \mathbb{R}$ represents the set of content segments that may be cached at MEC r, which is $N_r = \{n^r_{f,k} | f \in \mathcal{F}$ and $k = \{1, 2, \ldots, B\}\}$. Then,* (5.12) *and* (5.13) *can be rewritten as $A \in \mathcal{I}$, where*

$$\mathcal{I} = \{A \subseteq N \big| |A \cap N_r| \leq S_r, \gamma \leq |A \cap N_r|, \forall r \in \mathcal{R}\} \tag{5.17}$$

Algorithm 5.1 Greedy Cooperative Content Placement Algorithm

// $\mathcal{F}$: set of contents
// $\mathcal{U}$: set of users
// S_r: storage capacity of $r \in \mathcal{R}$
// β_r: data transmission rate $r \in \mathcal{R}$
// p_f: content popularity $f \in \mathcal{F}$
// P_u^v: mobile probability
Output: X: Content Placement.

Initialize $A = \emptyset$; /*i.e., $x_f^r = 0 \; \forall r \in \mathcal{R}$ and $f \in \mathcal{F}$ */
N_{rem} = Set of all elements that may be add to X; /*i.e., $N_{rem} = N$ assigning the ground set */
Sort N_{rem} in non-increasing order;
for all $u \in \mathcal{U}$ **do**
    Compute user moving probability from Eq. (5.5);
    Compute sojourn time from Eq. (5.6);
end for
repeat
    $n_{f',k'}^{r'} = \arg\max_{n_{f,k}^r \in N_{rem}} [g(A + \{n_{f,k}^r\}) - g(A)]$;
    $A = A + n_{f',k'}^{r'}$; /*i.e., $x_f^r = 1$ */
    $N_{rem} = N_{rem} - n_{f',k'}^{r'}$;
    if $j^{'}$ is full **then** /*i.e., $|A \cap N_r| == S_r$ */
        Remove all the elements of N_r from N_{rem};
    end if
until $|A| > \sum_{r \in \mathcal{R}} S_r$
return $X = x_f^r$;

which is a matroid constraint.

Proof. The tuple $(N, \mathcal{I})$ be a member of partition matroid, which is typical matroid. □

Finally, the problem (5.11) can be rewritten as

$$\max g(A) \tag{5.18}$$

s.t.

$$A \subseteq \mathcal{I} \tag{5.19}$$

5.2.1 Greedy Algorithm for Contact duration Aware Cooperative Content Placement

The converted submodular optimization problem presented in Eq. (5.18) can be solved using a greedy algorithm with an approximation of $\frac{1}{2}$ in the worst case. The contact duration aware cooperative caching algorithm is presented in Algorithm 5.1. The greedy algorithm initially starts with an empty set. The marginal value is then computed for each element, then the maximum marginal value is added to the content placement. The function presented is a submodular function. Therefore, by adding more number of elements to X, the marginal value decreases. The algorithm stops when the marginal value becomes zero.

In Algorithm 5.1, N_{rem} indicates the remaining set, which contains the elements that are added to X. Line 9 computes the highest marginal value. After adding an item $n^{r'}_{f',k'}$ to X as shown in line 10, it should be removed from the remaining set (N_{rem}) as shown in line 11. Line 12-14 show that if the MEC i is full, then the segments stored at $r^{'}$ are removed. The process repeats until the cache capacity is full.

The proposed greedy algorithm achieves a polynomial time complexity; the complexity grows with an increase in the number of contents. For real scenarios as the scale continues to increase (large scale problems where hundreds of users, tens of BSs), the complexity can be very high, making it impossible for implementation [24, 25]. Therefore, low complexity sub-optimal algorithms are required due to cheapness and delay sensitive implementations [25]. To address the system with a large number of nodes, contents and mobility paths, and to simplify the computational complexity, a heuristic algorithm has been designed based on the genetic algorithm (GA) and presented in the next section. GA gives a near optimal and robust solution (video segment placement in content delivery networks [26] and base station placement in heterogeneous network [27]) for NP-hard problems.

5.3 GA based Cooperative Content Placement for large scale problems

Genetic Algorithm (GA) is a well known stochastic optimization technique which is inspired by the principle of natural evolution [171, 172]. It is observed that GA is fitted for parallel optimization [172]. GA is an iterative approach; each iteration is known as a generation. GA is a population (all possible individuals) based technique. From the given population, GA generates individuals randomly, the generated individuals converted into genetic form by encoding. The evolution of the encoded individuals is done by repeating the following steps till termination criteria satisfies.

1. The fitness function determines the strongest individuals with high fitness values and these strongest members are selected as parents for the next generation.

2. The genetic operators (crossover and mutations) carried out on the selected parents to produce a new generation from the current.

This process continues where the individuals being adaptable to the environment. With this idea of natural selection, a genetic algorithm has been proposed for contact duration aware cooperative content placement.

The contact duration aware cooperative content placement based on a genetic algorithm is presented as follows. The major steps involved in the genetic algorithm is presented below:

1. Genetic coding generation: The first step in GA is to encode the genetic information of population into a chromosome. In the proposed problem the decision variable x_f^r is an integer variable, so an integer based encoding scheme is used. x_f^r indicates that the number of segments of content f is cached at MEC r. The chromosome is represented as $x^r = \{x_1^r, x_2^r, \ldots, x_f^r, \ldots x_F^r\}$ from this $X = \{x^1, x^2, \ldots, x^r, \ldots x^R\}$.

2. Initialization: Initialize the genetic parameters including the number of iterations, crossover rate c_p, mutation rate m_p, population size and genetic population $P(n)$ composed of N_{pop} chromosomes. Generate initial population randomly.

Algorithm 5.2 Genetic Algorithm for Cooperative Content Placement

// N_{pop}: `Population size`
// c_p: `Crossover probability`
// m_p: `Mutation probability`
// gen: `Number of iterations`
Output: X: Content Placement.

```
t = 0;
Initialize P(n);
Repair P(n);
Evaluate P(n);
Store best solutions of P(n) in old B(m);
while t < gen do
    Selection P(n);
    Crossover P(n);
    Mutation P(n);
    Repair P(n);
    Evaluate P(n);
    Store the best fitness individuals of P(n) in new B(m);
    if FIT(old B(m)) > FIT(new B(m)) then
        new B(m) = old B(m);
    end if
    old B(m) = new B(m);
    find the worst fitness value in P(n) and replace it with new B(m);
    t = t + 1;
end while
```

3. Re-pairing: In the given content placement, each MEC should not cache more than its capacity ($\sum_{f \in \mathcal{F}} B_f \cdot x_f^r \leq S_r$). This condition satisfied by performing the repair algorithm shown below (Algorithm 5.3).

4. Evaluation: The fitness of each individual is computed using Eq. (5.10). The population is evaluated based on the fitness calculated. The fitness values of an elite individual in the current iteration and previous iteration are identified. This identified result is used to speed up the evaluation by replacing individual with the worst fitness value.

5. Termination criteria: Once the iteration reaches the termination criteria, stop the evaluation process and output the result otherwise continue the process.

6. Selection: The selection process chooses the elite individuals whose fitness value

Algorithm 5.3 Repairing Process

for $r \in \mathcal{R}$ **do**
  while $\sum_{f \in \mathcal{F}} B_f \cdot x_f^r > S_r$ **do**
    Remove content from x_f^r in ascending order of popularity of contents at MEC r;
    $S_r = S_r - B_f$;
  end while
  while $\sum_{f \in \mathcal{F}} B_f \cdot x_f^r \leq S_r$ **do**
    Add content to $x_f r$ in descending order of popularity of contents at MEC r;
    $S_r = S_r + B_f$;
  end while
end for

is useful to next-generation (placing them in the mating pool). The roulette wheel method is used to select the elite individuals. The individuals are selected by selection probability corresponding to its fitness function. The probability of selecting each individual is defined as

$$P_s = \frac{FIT(j)}{\sum_{j \in N_{pop}} FIT(j)} \tag{5.20}$$

where *FIT(j)* represents fitness value of j.

Algorithm 5.4 Selection Process

Compute selection probability of each individual by Eq. (5.20);
for $j \in N_{pop}$ **do**
  i = 0; /* chromosome index */
  P_N = 0; /* accumulation probability of individuals */
  while $P_N <$ random$(0, 1)$ **do**
    $i++$;
    $P_N = P_N + P_s(i)$;
  end while
  Selected individual = i;
end for

7. Recombination: The crossover (recombination) is process blending genetic information of the parent chromosomes to produce new solutions. Crossover is carried out as per crossover probability. Each row from a parent forms a pair from two individual matrices. Two-point crossover mechanism is adopted in this chapter, where the two

crossover points are picked randomly within the length of the chromosome. Here, the fragment from the chromosome of the first parent between two crossover points and the fragment from the second parent is switched. The mutation process randomly alters the points based on the mutation probability.

Algorithm 5.5 Crossover Process

1: **for** pick two chromosomes (Ch1, Ch2) from the given population with step size 2 **do**
2: **if** c_p > random[0,1] **then**
3: cp1 = random[0, F];
4: cp2 = random[p1, F];
5: Switch the string fragment between two crossover points cp1, cp2 in two individuals Ch1, Ch2;
6: **end if**
7: **end for**

5.4 Performance Evaluation

This section validates the performance of the proposed contact duration aware cooperative content caching (Greedy) algorithm and genetic caching algorithm (GA) using simulations. The proposed algorithms have been compared with the existing algorithms based on publicly available real-world datasets available at WTD Project [36] and MovieLens [37].

5.4.1 Simulation Environment

A cellular network with 15 BSs associated with MEC servers and 90 mobile users has been considered. In the given simulation area, MECs are randomly deployed and connected. Assume that the mobile users' initial locations are uniformly distributed over MECs at the beginning of the simulations [24, 34, 35]. The content server holds a total of 3952 contents (MovieLens dataset) with a content size of 40 MB similar to [21] and each content is encoded into two segments similar to [22]. Content request probability follows Zipf distribution with γ = 0.6 [24]. The cache capacity of each MEC is 10 per cent of the entire video library. The data transmission capacity of MEC is 8 Mbps. The deadlines of each file are 600s. The values of the simulation parameters are present in Table 5.3.

The presented simulation results are obtained by taking average of 50 runs. In order to evaluate the performance of the proposed caching algorithms, the proposed mechanisms experimented using publicly available code in the Visual Studio environment [173, 21]. The code in [173] provides the simulation area setup (i.e., distribution of MECs and users) and the mobility prediction using Markov chain [168] by considering the contents' deadline. The proposed contact duration aware cooperative content caching (Greedy algorithm), and Genetic algorithm as well as the existing MCFD (Mobility aware caching with fixed amount data delivery) [22, 21, 34], Femto caching [57] and most popular content [49, 60] caching mechanisms have been implemented.

Table 5.3: Simulation Parameters

Parameters	**Values**
Simulation area	$500/m \times 500/m$
Content size	40 MB
Capacity of base station	10 % of library
Communication range of BS	100 m

5.4.2 Performance Metrics

To compare the performance of cache placement schemes, two metrics have been considered:

(1) *Cache Hit Ratio*: The fraction of requests satisfied (hits) from the available caches over the sum of cache hits and cache misses.

$$\text{Cache hit ratio} = \frac{\text{cache hits}}{\text{cache hits} + \text{cache misses}} \tag{5.21}$$

(2) *Acceleration ratio* [75] : the fraction of saved transmission delay and original Internet delay (from content server) can be formulated as:

$$\text{Acceleration Ratio} = \frac{\text{saved delay}}{\text{original delay (from Internet)}} \tag{5.22}$$

5.4.3 Reference Algorithms

The proposed algorithms is compared with the following caching mechanisms: Most popular caching [49, 60], Femtocaching [57] and Mobility aware caching with fixed data delivery [22, 21, 34].

1. MPC (Popular Caching): In the most popular caching scheme, each base station (MEC) caches the most popular content based on the user request statistics. Each MEC caches the popular content until the cache is full.

2. FC (Femtocaching): Initially, all the caches are empty. In femtocaching, all the users are distributed uniformly, and users remain static in the allocated cell, i.e., each user is associated with the same MEC during the evaluation. The FC iteratively caches the content into the MEC maximizing the saved delay. This process continues until the cache is full.

3. MCFD (Mobility aware caching with fixed amount data can be delivered): In MCFD, the mobility of users is considered, and each MEC delivers only fixed amount of data, i.e., the contact duration of the user and MEC are not considered. So, a fixed amount of content is cached at every MEC.

The first two caching mechanisms cache the content only based on user request rate, whereas the third mechanism considers mobility of the user and coded segments are cached at each MEC instead of the entire file. In the third caching mechanism contact duration is not considered. So, a fixed amount of content is cached at each MEC.

5.4.4 Mobility Model

To demonstrate the efficacy of the proposed content placement mechanism, the real trace of mobile users released by the Wireless Topology Discovery (WTD) project has been used [36]. The trace comprises data from 275 personal digital assistant (PDA) users for 11 weeks period from 22/09/2002 to 8/12/2002. Each user holding the PDA device identifies the WiFi access points encountered in its moving path for every 20 seconds. In the WTD project, 400 access points (APs) are densely deployed and the locations of access points

are recorded by (x, y) values. Because of densely deployed access points user may be under the coverage of multiple APs at a time. This simulation, considers a sub-area with 15 densely positioned access points, and replaces the MEC with APs [34, 21]. The busiest day (16/10/2002) of the 11-week duration and four different one hour time intervals are considered during simulation [34]. In this simulations, time is divided into slots of each 20 seconds (i.e., in the WTD project traces, the users record the MECs at each sampling point (time interval between two sampling points is 20 seconds [36])). The locations of MECs covered by the users during mobility are treated as users locations. These sets of locations (as covered by a user) are called a walk of the user. Based on the traces, the value of $P_{i,j}$ is computed as the fraction of the number of sequential visits to MECs (locations) i and j over the frequency of visits to i. The $P_{i,j}$ value is increased if the user remains in the same location (i.e., the user does not visit any of the MECs by the end of the time slot).

5.4.5 Demand Model

This simulation, uses the MovieLens 1M Dataset [37] to model user request demands. Dataset has 6040 users with demographic information (age, gender, location and occupation) and 1000209 ratings of 3952 movies. The dataset consists of user ID, movie ID, movie ratings and timestamp. The rating of users in timestamps has been assumed as the content request of the user [129, 160, 158] (the assumption is that a user rates a movie after watching it). It has been observed that more than 90% of the ratings existed within the first year. Therefore, only the first year of the dataset has been used [129]. First, the timestamps are divided as slots one hour each, later the user context information to the user requests is assigned [158].

5.4.6 Impact of number of MECs

In this section, the influence of the number of MECs on acceleration ratio and cache hit ratio is shown in Fig. 5.4a and 5.4b. In this simulations, cache capacity of MECs is 10%, delay deadline is 3 slots, the data transmission rate is 8 Mbps, contact duration is 6 min, and the number of MECs varies from 5 to 15 with step size 2.

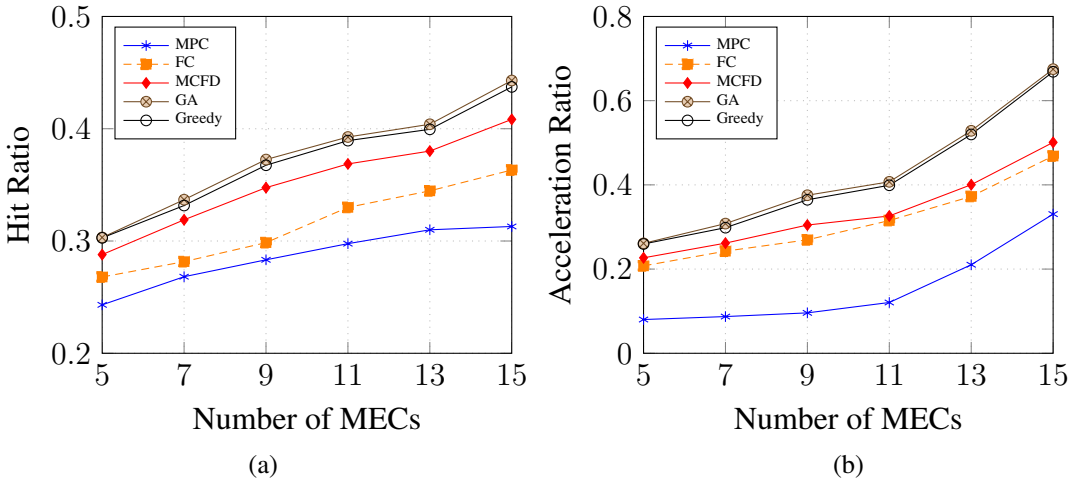


Figure 5.4: Comparison of caching schemes using number of MECs vs (a) cache hit ratio (b) acceleration ratio. When C = 10 %, d = 3 slots and b = 8 Mbps.

The impact of the number of MECs on cache hit ratio is shown in Fig. 5.4a. It can be observed from Fig. 5.4a that the cache hit ratio of all the algorithms is increasing with the large number of MECs. It can be observe that, the mobility aware caching mechanisms are outperformed the stationary caching mechanism because the users come across more MECs in their mobility path within the deadline. The proposed algorithms perform better than the MCFD because the proposed caching mechanisms consider the contact duration, which provides the space for unpopular content by caching the more popular content based on contact duration. The gain of contact duration aware cooperative content caching (Greedy algorithm) has been shown over existing algorithms MPC, FC and MCFD by 8, 5 and 1.3 per cent respectively. GA outperforms the MPC, FC and MCFD by 9, 6.3 and 2.5 per cent respectively.

The influence of the number of MECs on acceleration ratio is shown in Fig. 5.4b. The acceleration ratio grows slower with a small number of MECs and then grows faster for the large number of MECs. The reason is that the majority of the requests are intended for most popular cached content. MPC caches global popular content. The mobile users move across MECs. Therefore, the requests are forwarded to the content server. FC performs better compare to MPC because FC caches the content cooperatively. MCFD and contact duration aware algorithms perform relatively similar in case of a small number of MECs. The

proposed mechanisms show better performance when the number of MECs increases. The reason is that even though both mechanisms consider mobility, the proposed mechanisms which cache requested content use contact duration along with mobility. Therefore, the proposed greedy and genetic algorithms outperform the other algorithms.

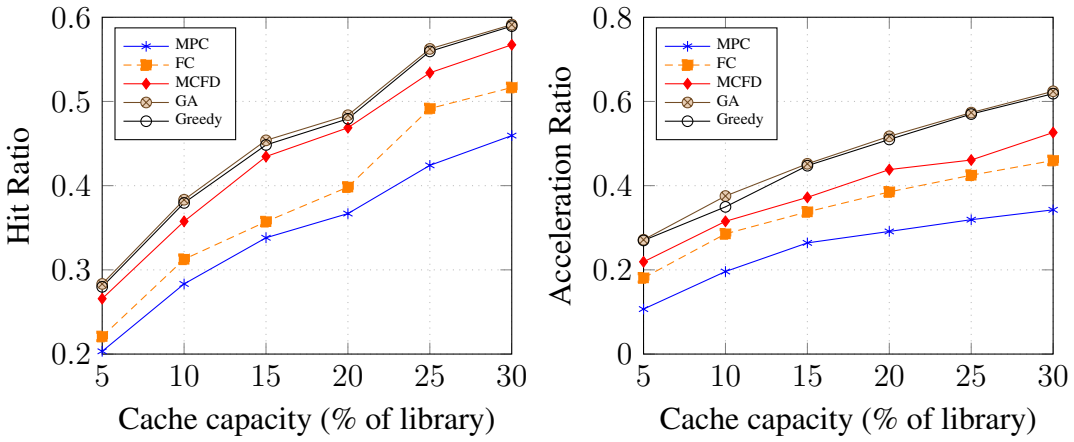


Figure 5.5: Comparison of caching schemes using cache capacity vs (a) cache hit ratio (b) acceleration ratio. When N = 10 %, d = 3 slots and b = 8 Mbps.

5.4.7 Impact of Cache Capacity

This section shows that the impact of cache capacity on acceleration ratio and cache hit ratio shown in Fig. 5.5a and 5.5b. In this simulations, number of MECs is 10, delay deadline is 3 slots, the data transmission rate is 8 Mbps, contact duration is 6 min, and the cache capacity varies from 5% to 30% of the total library size.

In Fig. 5.5a, cache hit ratio for different mechanisms is shown for different cache capacities. It can be observed that the mobility aware caching mechanisms MCFD, GA and Greedy outperform static caching mechanisms Femto caching and MPC. The reason is that mobility aware algorithms take caching decisions based on user mobility. The proposed greedy and GA algorithms outperform all other algorithms because the caching decisions are taken using the contact duration of users in MEC. MCFD scheme considers the user mobility in caching decision. However, MCFD Scheme does not consider the contact time. Most popular content caching (MPC) takes caching decisions based on the global popu-

larity of the content which may not be popular locally. Here, when the cache capacity increases, the cache hit ratio also increases because more content is available when the cache size is large. The gain of the proposed Greedy algorithm has been shown over existing algorithms MPC, FC and MCFD in terms of hit ratio by 10, 6.8 and 1 per cent respectively. Further, the proposed GA outperforms the MPC, FC and MCFD in terms of hit ratio by 11, 8 and 2.4 per cent respectively.

In Fig. 5.5b, acceleration ratio is shown for various cache capacities. It can be observed from Fig. 5.5b that the increase in cache capacity leads to increase in acceleration ratio. The reason is that more content will be cached as the cache capacity increases and this leads to the availability of different contents at MEC. MPC does not perform well as compared to other mechanisms (as shown in Fig. 5.5b) because the content will be cached at MEC based on global popularity. In contrast, Femto caching considers the distributed cache as a single cache due to the cooperation of the MECs. The mobility aware caching mechanisms outperform static caching mechanisms because the cache placement decision is made based on the mobility, which allows informed caching. The proposed caching mechanisms show superiority over the MCFD by allowing more content to be placed at MECs because it considers contact duration and coding.

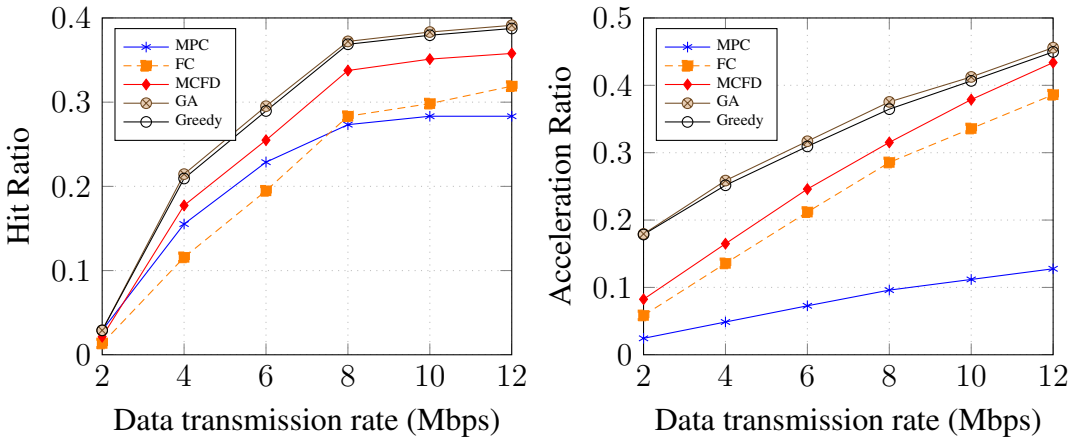


Figure 5.6: Comparison of caching schemes using average data transmission rate vs (a) cache hit ratio (b) acceleration ratio. When C = 10 %, d = 3 slots and N = 10.

5.4.8 Impact of data transmission rate

In this section, the effect of transmission rate on acceleration ratio and cache hit ratio is shown in Fig. 5.6a and 5.6b. In this simulations, the number of MECs is 10, delay deadline is 3 slots, contact duration is 6 minutes, cache capacity is 10% (of the library size), and data transmission rate varies from 2mbps to 12mbps with step size 2.

The impact of the data transmission rate on the hit ratio is shown in Fig. 5.6a. The hit ratio grows faster with low data transmission rate and then grows slowly for larger transmission rates. The contact duration based mechanisms outperform MCFD because the cache placement decision is made based on the contact duration and this allows MECs to cache different content. The gain of the proposed Greedy algorithm has been shown over existing algorithms MPC, FC and MCFD by 5.7, 6.2 and 1.6 per cent respectively. Further, the proposed GA outperforms the MPC, FC and MCFD by 7.2, 7.7 and 3 per cent respectively.

In Fig. 5.6b, acceleration ratio is shown for various data transmission rates. MPC mechanism may not fulfill the user demands because MECs are cached with the globally popular content. FC caches content cooperatively which allows different content to be cached at MECs. MCFD considers a fixed data transmission rate. The contact duration aware caching mechanisms performs well compared to MCFD because MCFD (delivers the fixed amount of content even if contact time is more) does not consider users' contact duration.

5.4.9 Impact of contact duration

In this section, the effect of contact duration on cache hit ratio is shown in Fig. 5.7a and 5.7b. In this simulations, the number of MECs is 10, delay deadline is 3 slots, cache capacity is 10% (of the library size), the data transmission rate is 8 Mbps, and contact duration varies from 10 to 60 with step size 10.

The impact of the contact time on the hit ratio is shown in Fig. 5.7a. The hit ratio increases with growth in the contact time. The contact duration aware caching strategies outperform other caching mechanisms. The reason is that the MPC caches popular content

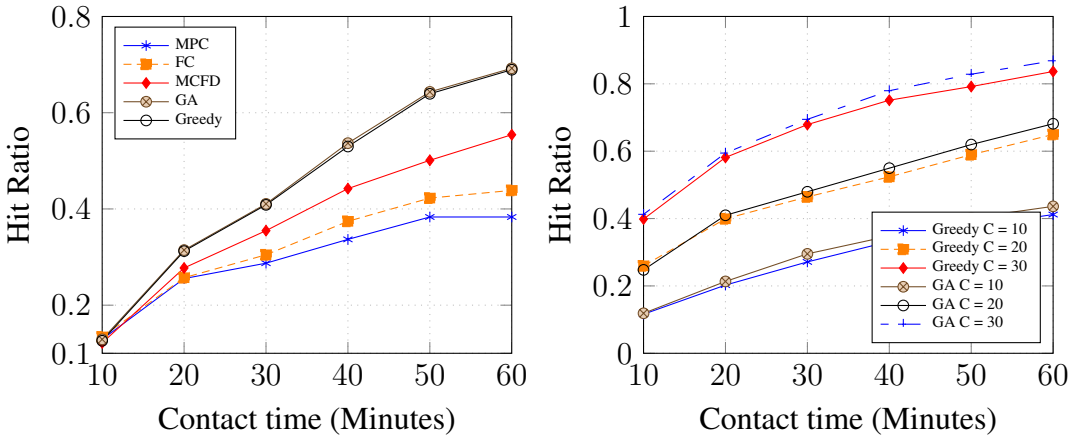


Figure 5.7: Comparison of caching schemes using contact time vs (a) cache hit ratio, when C = 10 %, d = 3 slots, b = 8 Mbps and N = 10. (b) hit ratio for mobile user with different contact time, where d =3 slots, b = 8 Mbps and N = 10.

by not considering mobility. FC mechanism considers the cooperation among MECs, but mobility is not considered. However, MCFD considers mobility, and it does not take a coding scheme into consideration. Hence, the contact duration aware caching mechanisms perform better than other schemes. Fig. 5.7a shows that with short contact time cache hit ratio is low because users moving with high speed encounters more MECs. The hit ratio is improved for slowly moving users because the contact time with a MEC increases. The gain of the proposed Greedy algorithm has been shown over existing algorithms MPC, FC and MCFD by 12.9, 10.3 and 4.5 per cent respectively. Further, the proposed GA outperforms the MPC, FC and MCFD by 16, 13.4 and 8 per cent respectively.

The impact of the mobility speed (i.e., with variable contact time) on hit ratio is shown in Fig. 5.7b. This simulation, considers the hit ratio of mobile users with different contact time with MECs. The contact time is inversely proportional to user mobility. The hit ratio decreases with high mobility of the user because the user cannot receive the requested content fully when the user mobility is high. With the low mobility speed, the user can collect the requested content successfully from MECs.

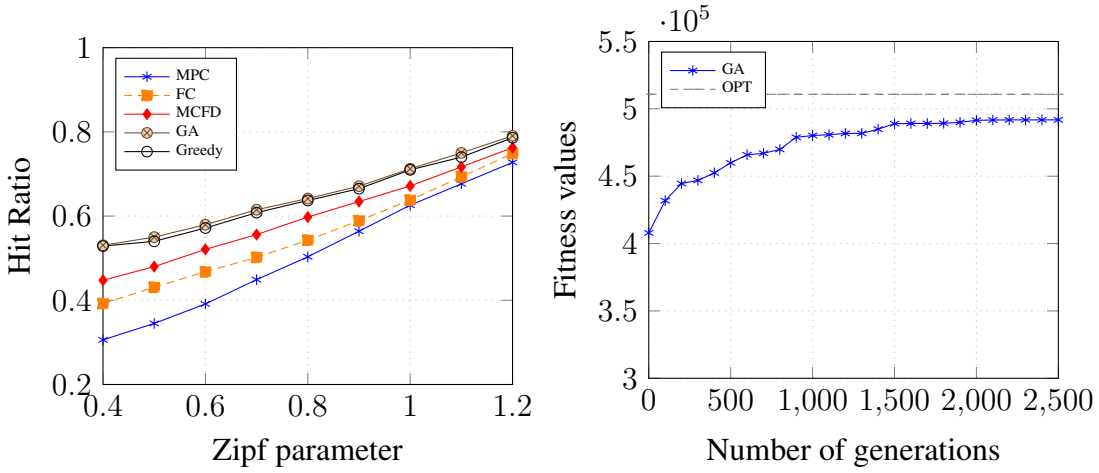


Figure 5.8: (a) Comparison of different caching mechanisms with content popularity profile (Zipf parameter) γ where C = 10%, d = 3 slots and b = 8 Mbps (b) Convergence behavior of saved delay maximization with N_{pop} = 150, c_p = 0.95 and m_p =0.05.

5.4.10 Impact of content popularity

The effect of content popularity on the cache hit ratio is shown in Fig. 5.8a. It can be observed from Fig. 5.8a that the cache hit ratio increases as the Zipf parameter γ increases. The reason is that as γ increases, fewer popular content attracts more user requests. The contact duration aware caching strategies outperform over existing caching schemes. The proposed algorithms perform better compared to other caching schemes because the proposed algorithms use the contact information.

Fig. 5.8b shows that as the number of generations increases the gap between the optimal solution (OPT) and genetic algorithm-based solution decreases. From Fig. 5.8b, it can be noticed that the fitness value raises sharply with a few generations. It converges to an suboptimal solution when the number of generations become approximately 1000.

5.5 Summary

This chapter analyses the influence of user mobility and contact duration on cache placement in mobile edge networks, aiming to maximize the saved delay by considering the capacity constraint. The user mobility has been considered as a Markov renewal process to

predict the contact duration and the moving path in the problem formulation. An effective greedy algorithm has been designed to solve the formulated problem. Further, a heuristic search mechanism based on a genetic algorithm has been proposed to solve a large scale problem. Simulation results based on real-world traces of user mobility and requests for content demonstrate that the proposed contact duration aware caching mechanisms outperform three caching strategies (such as most popular caching, femto caching and mobility aware caching). From the simulation results, it can be observed that the proposed greedy and genetic algorithms provide improvement of up to 13 and 16 per cent on hit ratio compared with MPC, FC and MCFD, respectively. The next chapter presents a cooperative cache replacement mechanism using recurrent multi-agent deep reinforcement learning in the obscene of content popularity information.

Chapter 6

Cooperative Cache Replacement using Recurrent Multi-Agent Deep Reinforcement Learning for Mobile Edge Networks

The content needs to be fetched from the far distant content server in the peak time when the requested content is not cached at the edge node and this leads to high delay, backhaul load and congestion, which is known as reactive caching. In this chapter, a reactive caching mechanism has been presented in a multi-cell scenario. Earlier works presented proactive caching schemes by considering the content popularity which is known in advance [14] or predicted [15, 16]. In some of the realistic scenarios, the content popularity is time-varying, so the above assumption (known in advance) may not be feasible. Considering the dynamic nature of the content popularity, high dimensional parameters, and for an intelligent caching decision, the conventional optimization methods will not be suitable [30]. The recent success in Reinforcement Learning (RL) [31], strong characteristic representation capability of Deep Reinforcement Learning (DRL) [16, 32] to tackle the changing nature and complex systems has encouraged this research work to use these learning mechanisms to solve the above problem.

This chapter aims to maximize the saved delay by considering the dynamic nature of content popularity along with the capacity and deadline constraints for accessing a large volume of data. A DRL based cooperative caching mechanism has been proposed using the actor-critic framework. Since each edge node observes its local state, the cooperative cache replacement problem is modelled as a Partially Observable Markov Decision Process (POMDP) [33]. The modelled multi-agent decision problem optimizes the latency of obtaining content from local MEC, neighbouring MEC and content server. To manage nodes to coordinate the caching decisions, a multi-agent actor-critic framework has been adopted. The contributions of this chapter are as follows:

- Design an integer linear programming problem for content caching problem: maximization of saved download delay in the absence of content popularity information with deadline and capacity constraint.
- Formulating the cooperative cache update problem as a POMDP based on a multi-agent decision problem to maximize the cumulative reward by ensuring the coordination of the MEC servers.
- Design a multi-agent recurrent deep reinforcement learning-based cooperative caching algorithm by devising the multi-agent actor-critic framework to solve the given problem (i.e., MARDDPG algorithm).
- Extensive simulations have been performed to show the efficacy of the proposed recurrent multi-agent cooperative caching algorithm by considering acceleration ratio, hit ratio and caching reward.

The rest of the chapter is organized as follows. In Section 6.1, system model and formulation of the content replacement problem are presented. A multi-agent DRL model for the proposed problem is presented in Section 6.2. Then, a multi agent actor-critic framework is presented in 6.3.1 and the multi-agent recurrent DRL algorithm is presented in Section 6.3.2. Simulation environment and results are presented in Section 6.4. A summary of this chapter is mentioned in section 6.5.

6.1 System Model and Problem Formulation

In this section, the network model and problem formulation are presented in detail.

6.1.1 Network Model

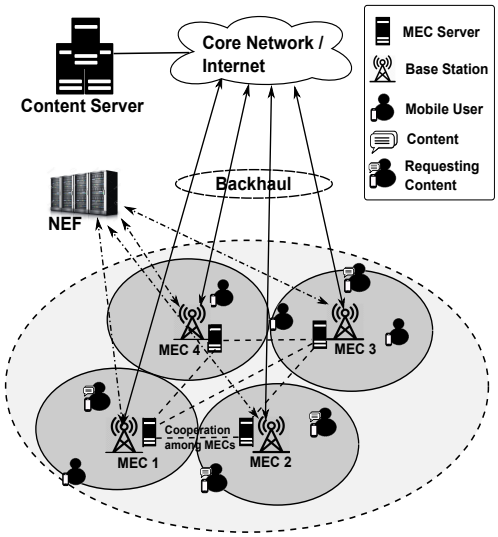


Figure 6.1: Illustration of the proposed system model.

Mobile edge computing improves users' capabilities by providing cache capacity (i.e., storage), network resources and computing near to the users. Consider a mobile edge network containing a set $\mathcal{M}$ of M small base stations (BSs) equipped with a MEC server, a set $\mathcal{U}$ of U mobile users, a content server and a central coordinator NEF as shown in Fig. 6.1. Each MEC $i \in \mathcal{M}$ has a limited cache S_i called local storage. The storage of each MEC is used for content caching. The MECs are connected and also to the core network through the backhaul link. The content server acts as an origin server that stores all contents. Network Exposure Function (NEF) serves as a coordinator (and it is a crucial network element in 5G networks) [151]. NEF has a global view and it maintains the content cached at individual MECs and monitors users' content requests at each MEC [151]. A user directly connected to a base station (BS) and the user may be in the communication range of more than one BS at any point in time. However, any user can communicate with only one BS at a particular time. Mobile users are attached to the base stations according to a cellular network protocol. The connected base stations are accountable for serving user

requests. Each BS receives content requests from multiple users in the communication range without knowing its popularities. The MEC can serve the requests in three ways: 1) local MEC, 2) neighbour MEC, and 3) content (central) server.

Consider a set $\mathcal{F}$ of F contents in the content library located in the content server. Each content f is determined with two features S_f denotes the size of the content and dl_f denotes maximum allowed access latency to get content f. The time split into slots and each time slot is denoted by $t \in T$. Assume that the content requests are independent. The user can request only one content in time t and user location cannot change in any given time slot. The requests generated by user u at time t is represented by a binary vector $W_{i,u}^t = \{w_{i,u,1}^t, w_{i,u,2}^t, \ldots, w_{i,u,F}^t\}$. $w_{i,u,f}^t = 1$ means the user u requests content f in MEC i at time t, $w_{i,u,f}^t = 0$ otherwise. Assume that the content popularity is unknown. The frequency of content is indicated as $\rho_{i,u}^t = \{\rho_{i,u,1}^t, \rho_{i,u,2}^t, \ldots, \rho_{i,u,F}^t\}$, where $\rho_{i,u,f}^t$ denotes the cumulative requests for content f in MEC i at time t. Also, assume that in each time slot, the MECs storage is filled with contents.

Table 6.1: List of Notations

Term	**Definition**
$\mathcal{M}, \mathcal{F}, \mathcal{U}$	Set of base stations, contents and users
S_i	The cache capacity of i-th MEC
S_f, dl_f	The size and deadline of f-th content
$t \in T$	Time slot
$W_{i,u}^t$	Requests generated by user u in MEC i at t
$\rho_{i,u,f}^t$	Cumulative requests for content f in MEC i at t
$d_{i,u}, d_{j,u}, d_{h,u}$	Delay from local, neighbour and central server to user u
$x_{f,i}^t$	The content f cached in MEC i at time t
D^t	The expected saved delay
s_i^t, a_i^t, R_i^t	System state, action spaces and reward at MEC i in t
K_i^t, N_i^t, B_i^t	Set of user requests, cache state and deadline of MEC i
$\psi_{f,i}^{t,l}, \psi_{f,i}^{t,a}, \psi_{f,i}^{t,h}$	Low, average and high priority of f in MEC i at t
$c_i, c_{i,j}, c_{i,h}$	Cost of serving f from local, nearby and central server
$r_{i,l}^t, r_{i,j}^t, r_{i,h}^t$	l contents fetched from local, nearby and central server
$\alpha_i, \delta_{f_+,i}^t$	Cost of replacement and number of contents replaced at i
h_a^t, h_c^t	Actor and critic network historical information
$y_{i,j}^t$	target network
$\emptyset, \theta$	actor and critic network weight parameters

6.1.2 Problem Formulation

The cooperative content replacement problem formed as an optimization problem to maximize the saved download delay. A binary indicator $X^t \in \{0, 1\}$ denotes the cooperative cache replacement scheme in MEC, $x^t_{f,i} \in X^t$ is 1 if the content f is stored in cache of MEC i at time t, 0 otherwise. The download delay is a typical metric to evaluate the performance in mobile edge networks. First, find the expected saved delay, then formulate the maximizing the saved delay subjective to capacity and deadline constraints.

The delay for getting content f from MEC i to user u is denoted as $d_{i,u}$. The content requested by the user retrieved from the local storage of the corresponding MEC, then the delay is considered as 0. In case of the content is not available at corresponding MEC i then i forward the request to neighbouring MECs as per the NEF direction. The delay is considered as the number of hops between user u and MEC j (j is the neighbouring node of MEC i) as $d_{j,u}$. If the requested content is unavailable at any of the MECs, the user fetches the content from the central server $d_{h,u}$ and $d_{i,u} < d_{j,u} < d_{h,u}, \forall j \neq i, j \in \mathcal{M}$.

Definition 6.1.1 (Saved delay)**.** *The difference in delay from the content server and MEC node is defined as the saved delay.*

The saved delay depends on the frequency of cached content requests. The saved delay is split into two parts in the proposed model, intra MEC saved delay (local caching) and inter MEC saved delay (cooperative caching). Intra MEC saved delay is attained by locally cached contents.

$$D^t_{i,0} = \sum_{f=1}^{F} \sum_{u=1}^{U} w^t_{i,u,f} \cdot \rho^t_{i,u,f} \cdot S_f \cdot x^t_{f,i} \cdot (d_{h,u} - d_{i,u}) \tag{6.1}$$

Inter MEC saved delay is attained by the neighbouring MECs sharing the cached contents.

$$D^t_{i,1} = \sum_{f=1}^{F} \sum_{u=1}^{U} w^t_{i,u,f} \cdot \rho^t_{i,u,f} \cdot S_f \cdot (1 - x^t_{f,i}) \cdot z^t_i \cdot (d_{h,u} - d_{j,u}) \tag{6.2}$$

where $z_i^t = (1 - \prod_{j=1, j\neq i}^{M}(1 - x_{f,j}^t))$. The expected saved delay is

$$D^t = \sum_{i=1}^{M} D_{i,0}^t + D_{i,1}^t \tag{6.3}$$

Aim of this work is to maximize the saved delay by replacing the requested contents at each MEC subjective on deadline and capacity constraints. Hence, the problem is formulated as:

$$\text{Max } \frac{1}{T}\sum_{t=1}^{T} D^t \tag{6.4}$$

s. t.

$$\sum_{f=1}^{F} S_f \cdot x_{f,i}^t \leq S_i, \qquad \forall i \in \mathcal{M} \tag{6.5}$$

$$\sum_{f=1}^{F} x_{f,i}^t \leq 1, \qquad \forall i \in \mathcal{M} \tag{6.6}$$

$$\sum_{i=1}^{V} x_{f,i}^t \leq M, \qquad \forall f \in \mathcal{F}, 1 \leq i \leq M \tag{6.7}$$

$$D^t \leq dl_f, \qquad \forall f \in \mathcal{F}, \forall i \in \mathcal{M} \tag{6.8}$$

$$x_{f,i}^t \in \{0, 1\}, \qquad \forall f \in \mathcal{F}, i \in \mathcal{M} \tag{6.9}$$

The objective (6.4) is the total saved delay of the overall network. Constraint (6.6) and (6.7) guarantees that the MEC node is not allowed to cache duplicate content. Constraints (6.5) provides the finite capacity of each BS. Constraints (6.8) is the deadline constraint, which ensures that the maximum allowable delay for the response to a request. Thus, the BS can satisfy the users' QoS requirements. Finally, constraint (6.9) is the non-negativity and integrality of the decision variables.

The cooperative content replacement problem presented in Eq. (6.4) is an integer linear programming (ILP) problem. The proposed problem can be shown as NP-hard by transforming the knapsack problem (known as NP-hard) into our problem. The problem presented in Eq. (6.4) can be addressed by finding the optimal decision variables $\{X^t\}$ in the present time slot. Nevertheless, the decision variable present in Eq. (6.4) is a binary variable and changing dynamically. Addressing the proposed problem requires to gather a

huge quantity of network state information. Besides, it is considered the practical scenario with an unknown content request pattern in advance. The conventional optimisation methods cannot be adopted because of the changing nature of the content popularity and to take an intelligent caching decision [30]. With the recent success in Reinforcement Learning (RL) [31], strong characteristic representation capability of Deep Neural Network (DNN) and Deep Learning (DL) [174] has encouraged the adoption of learning in wireless networks. The learning based mechanism allows an end-to-end solution from predicting the content requests to cache decision. As a branch of AI, reinforcement learning is extensively adapted in several fields (self-driving, robot control, etc.) to solve decision optimization problems. The multi-agent DRL (MADRL) [175] for cooperative content replacement has been adopted in mobile edge network.

The content placement is determined mainly based on the present state and the caching decision, which does not depend on the previous states. Therefore, the system states evolution can be modelled using a Markov process. The MEC has its cache state and current request information while taking the caching decision at time t. In the multiple MEC scenario, each MEC takes the caching decision based on its local cache state and each MEC does not have the caching state information of other (neighbouring) MEC nodes. In reality, MEC can not observe the complete system information regarding the caching states and content request distribution to take the cache decisions, which motivates to represent the problem as a partially observable Markov decision process (POMDP) [33]. Then a cooperative content replacement strategy has been developed using multi-agent reinforcement learning.

6.2 Multi-Agent Deep Reinforcement Learning Model for Cooperative Caching

In a multi-agent environment, individual agent can observe its local state, which is partial information about the environment. Therefore, in multi agent decision problems are modelled by POMDP. A POMDP is defined by a tuple $\{\mathcal{S}, \mathcal{A}, R, P, \Omega, \mathcal{O}\}$. $\mathcal{S}$ is a set of states,

A set of actions, R is the reward function and P is the transition probability from state s to s', which is defined in the MDP model. Ω is the set of observations and $\mathcal{O}$ is the observation probabilities. The observation, state, action and reward are defined as follows.

6.2.1 Observation and State Space

Let S is the set of system state space where $S = \{s_i | s_i = (N_i^t, K_i^t, B_i^t)\}$. In each time slot t the state s_i^t contains the set of user requests K_i^t, MEC i cache state N_i^t and B_i^t content delivery deadline of MEC i. Where $K_i^t = \{k_{i,1}^t, k_{i,2}^t, \ldots, k_{i,U}^t\}$, $k_{i,u}^t$ is the contents requested by user u at i in time t, $B_i^t = \{b_{i,1}^t, b_{i,2}^t, \ldots, b_{i,F}^t\}$, $b_{i,f}^t$ is the content delivery deadlines of MEC i for accessing the requested content f in time t. Since the content popularity is not available, the caching decisions are derived depend on the content already cached in MEC and the currently requested content. Therefore, the priority of the content cached in MEC in time t represented as $\psi_i^t = \{\psi_{f,i}^{t,l}, \psi_{f,i}^{t,a}, \psi_{f,i}^{t,h}\}$. Where $\psi_{f,i}^{t,l} = \sum_{t-\tau_l}^{t} \sum_{u=1}^{U} \rho_{i,u,f}^t$ is low priority, $\psi_{f,i}^{t,a} = \sum_{t-\tau_a}^{t} \sum_{u=1}^{U} \rho_{i,u,f}^t$ is average priority and $\psi_{f,i}^{t,h} = \sum_{t-\tau_h}^{t} \sum_{u=1}^{U} \rho_{i,u,f}^t$ is high priority. The system state space is denoted as

$$s_i^t = (N_i^t, K_i^t, B_i^t, \psi_i^t) \tag{6.10}$$

6.2.2 Action Space

Let A is the set of actions. Each MEC determines whether to keep or replace the content. The challenge in the MADRL is that multiple contents need to be replaced in each time slot. Since the system environment is multi-agent, each MEC serves multiple users. Therefore, different MECs get a different number of content requests.

If some user requests are missing from corresponding MEC and neighbouring MECs, then replace the missed content with appropriate content by fetching it from the content server. Otherwise (all the user requests miss), MEC replaces contents comprises of newly obtained content from the server with the suitable MECs and its cache. $A = \bigcup_{t \in T} a_i^t, \forall s_i \in S$ represents the action space analogous to the state space S. Upon receiving the content requests, MEC calculates each content's priority and determines whether the content to

keep or replace based on priority by satisfying constraints (6.5) and (6.8).

6.2.3 Reward Function

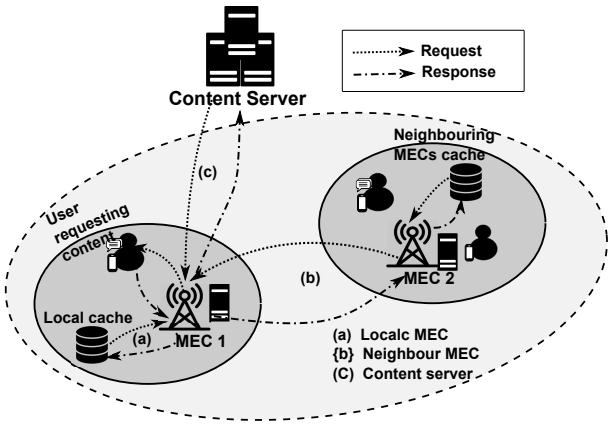


Figure 6.2: Illustration of requests served by MEC.

This work aims to maximize the saved transmission delay by obtaining the desired content at a low transmission delay within the fetching deadline. Each MEC node replaces the cached content at local storage cooperatively. In the multi-agent cooperative environment, based on the availability of the content, either neighbouring MECs or the central server, serves the local MECs requests. The MEC associated with the user called local MEC, the nearby MECs, is called neighbouring MECs.

1. Suppose the content requested by user is available at the local MEC, the content can be delivered immediately with low latency. The cost of delivering content from local MEC is denoted as c_i. Let's consider that the MEC i fetches l contents from its local storage in time t is indicated as $r^t_{i,l}$. Therefore, the cost of the local MEC service is represented as $c_i r^t_{i,l}$.

2. Suppose some of the contents requested by the user are not served by the corresponding MEC i. Consider that the content requested by user is available at neighbour MEC j, and the content is served by j to the user via MEC i. The cost of fetching content from j to i is denoted as $c_{i,j}$. Let l contents are fetched from j to i in time t is denoted as $r^t_{i,j}$. Therefore, the cost of neighbouring MEC service is represented as $\sum_{j \in \mathcal{M}, j \neq i} c_{i,j} r^t_{i,j}$.

3. Suppose the content requested by the user is unavailable at any of the MECs. The corresponding MEC obtains the content from the content server. Let's consider the cost to get the content from the content server to the user via MEC i denoted as $c_{i,h}$. Let l contents are fetched from content server h to i in time t is denoted as $r^t_{i,h}$. Therefore, the cost of content server service is represented as $c_{i,h}r^t_{i,h}$.

The overall cost of the service in time t is represented as

$$c_i r^t_{i,l} + \sum_{j \in \mathcal{M}, j \neq i} c_{i,j} r^t_{i,j} + c_{i,h} r^t_{i,h} \tag{6.11}$$

The content server serves the content miss at local MEC and neighbouring MEC. Hence, the MEC replace the newly fetched content with less popular content. Therefore, the cost should contain the replacement cost along with the delivery cost. Let the cost of replacing content at MEC i is denoted by α_i. The number of contents replaced by MEC i at time t is indicated as $\delta^t_{f_+,i} = f_i - (x^t_{f,i} \cap x^{t-1}_{f,i})$ where $x^t_{f,i}$ indicates content f cached in MEC i in time t, $x^{t-1}_{f,i}$ indicates the content f cached in MEC i at time $t-1$ and f_i indicates the content requests at MEC i. Therefore, the replacement cost is defined as

$$\sum_{f \in \mathcal{F}} \alpha_i \delta^t_{f_+,i} \tag{6.12}$$

The total cost is represented as sum of (6.11) and (6.12). That is

$$c_i r^t_{i,l} + \sum_{j \in \mathcal{M}, j \neq i} d_{i,j} r^t_{i,j} + c_{i,h} r^t_{i,h} + \sum_{f \in \mathcal{F}} \alpha_i \delta^t_{f_+,i} \tag{6.13}$$

It is considered that each content should be satisfied within the specified deadline of the content. If the content does not get within the deadline, the penalty cost should be included in the reward. The penalty cost of the system is represented as $\rho^t_{i,f} b^t_{i,f}$, where $b^t_{i,f}$ is the deadline of content f in MEC i and $\rho^t_{i,f}$ is the content frequency.

The cost of utilizing the local MECs cache is higher than without the local cache. Therefore, the saved cost need to be maximized for an effective caching scheme. The reward

function of MEC i is denoted as

$$R_i^t = (c_{i,h} - c_i) r_{i,l}^t + \sum_{j \in \mathcal{M}, j \neq i} (c_{i,h} - c_{i,j}) r_{i,j}^t - \sum_{f \in \mathcal{F}} (\alpha_i \delta_{f_+,i}^t + \rho_{i,f}^t b_{i,f}^t) \tag{6.14}$$

Maximizing the reward is maximizing the cost of saved delay. The term $r_{i,j}^t$ depends on the local cache and neighbouring cache. The instant reward of the system is defined as

$$R^t = \sum_{i \in \mathcal{M}} R_i^t \tag{6.15}$$

In a multi-agent system each MEC is considered as an agent. Based on the systems sates, each agent determines its cache placement. $\pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ indicate the set of all caching strategies, $\pi : S \rightarrow A$ is a caching policy, which associates the current system state s to a permissible action a. The optimal caching policy π^* maximizes the long-term reward in the multi-agent system. To maximize the system's long-term reward, each agent needs to work cooperatively because the immediate and long-term rewards impact agent actions. Hence, the cooperative content replacement problem expressed to maximize the cumulative discounted reward. The value function $V^\pi(S)$: is defined as

$$\mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R^t | s(0) = s, \pi\right] \tag{6.16}$$

where $0 \leq \gamma < 1$ is the discount faction, γ decides the future reward's effectiveness to the present decision. Lower γ values give more weight to the immediate reward. Finding the optimal caching policy π^* needs to follows Bellman's functions

$$V^{\pi^*}(s) = R(s, \pi^*(s)) + \gamma \sum_{s' \in S} P_{s's} V^{\pi^*}(s') \tag{6.17}$$

where $P_{s's}$ is the state transition probability. Bellman's functions usually solved by either value or policy iteration methods. However, Bellman's function presented in Eq. (6.17) is challenging because of the following points.

1. The state transition probability $P_{s's}$ is not known in advance without any prior knowl-

edge. It is difficult to estimate $P_{s's}$ in real environment.

2. The time complexity is high with the traditional value or policy iteration methods because of the vast state and action space, restricting practical cache systems' applicability.

3. Due to the cooperation among the MEC nodes, each MEC node should not cache the cached content at neighbouring MEC nodes. Each MEC can only know its local information and not aware of the full system sates and actions of the other MECs.

Hence, to address these issues, a MADRL based cooperative caching mechanism has been presented. The following section presents a multi-agent DRL mechanism to handle the proposed caching problem.

6.3 Multi-agent Recurrent DRL for cooperative Content Caching

In real world, the environment has challenging conditions for multi-agent system that demands the cooperation among agents, such as partial observable nature of agents and non-stationary nature. Therefore, the multi-agent DRL framework has been presented for cooperative content replacement in MEN.

6.3.1 Multi-Agent Actor-Critic Framework

Usually, there are two approaches to develop caching decisions, namely decentralized and centralized. In the centralized caching mechanism, the centralized server determines the caching decisions depend on the caching states' global view and assign them to edge nodes. Each edge node is responsible for executing the caching decisions and data storage. In the decentralized caching mechanism, the caching decisions determined by the edge nodes. Each edge node determines its caching decision based on other nodes cache state information received from the central server in this mechanism. The central server is responsible for synchronization and cache state information interaction. However, both approaches

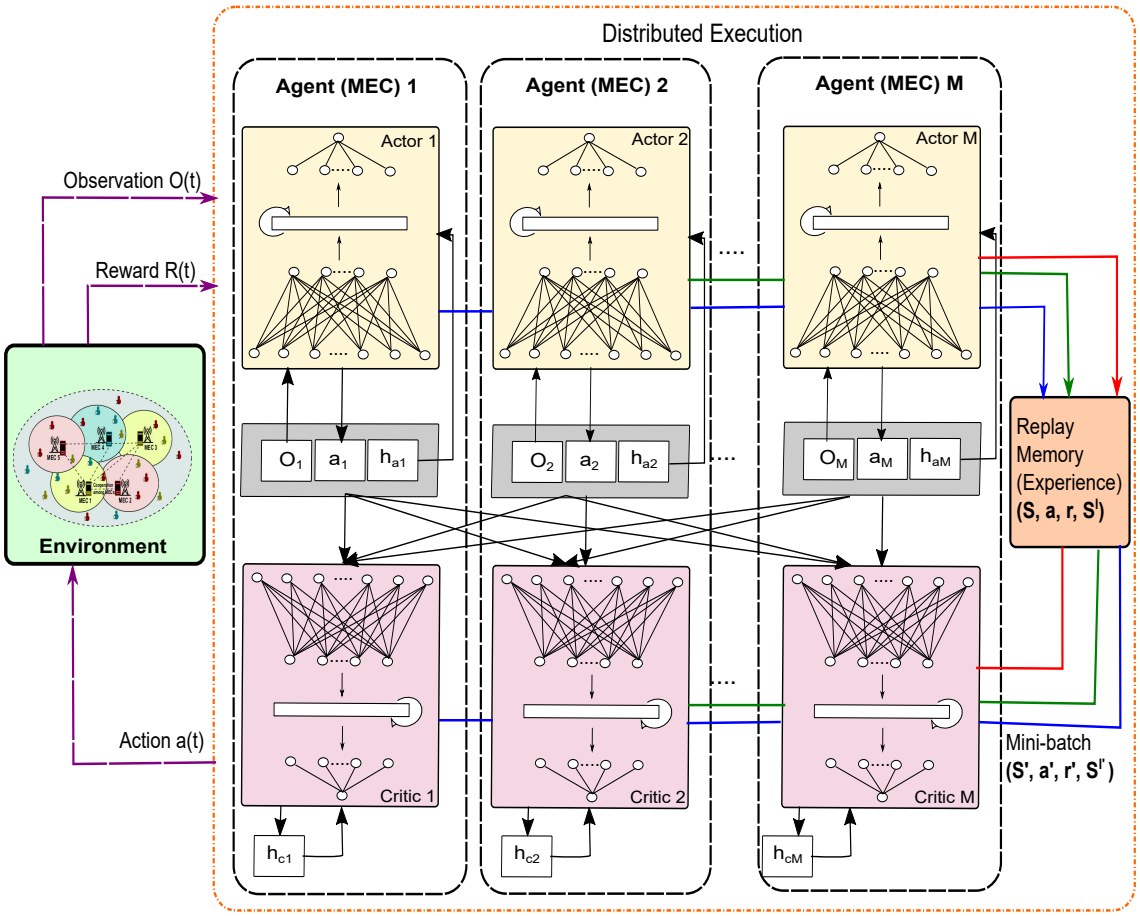


Figure 6.3: Multi-agent recurrent DRL framework for cooperative caching. Here O_i, a_i represents the observation and actions of agent i and h_a, h_c represents the history of actor and critic.

induce new problems. A centralized mechanism causes additional delay since the central server determines the cache decisions. Decentralized mechanism suffers from the cache state information exchanging problem, and it has a severe effect on cooperation among edge nodes.

A cooperative caching has been proposed by adopting centralized training with distributed execution framework to alleviate the problems mentioned earlier, providing the policies to utilize additional knowledge to simplify the training. In general, Q-learning cannot carry various information at the training and testing phase; therefore, making additional assumptions about the environment's structure is unnatural. Hence, an Multi-Agent Recurrent Deep Deterministic Policy Gradient (MARDDPG) algorithm has been proposed to decide whether to evict or retain the requested content inspired by [176, 175]. The

MARDDPG is a simple addition of the DDPG mechanism for a multi-agent system, where the actor can only obtain its local information and the critic augmented with additional information about other agents' policies. An individual agent is unaware of the other agents' policies in a multi-agent system, leading to a non-stationary problem. Each agent estimates the other agent policies by leveraging the actions and global state through the training phase to avoid the environment from the non-stationary problem. Hence, each agent attains the global optimal strategy by altering its policy depending on the other agents' estimated policy. Each agent consists of its own actor and critic networks in the proposed mechanism considering agent independently learns a distinct policy because of different locations.

Actor network: The actor-network described as a function that learns a caching policy $\pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ that maps the state to a permissible action taken from the action space $\mathcal{A}$. The state comprises the global state g and local state s_i observed by agent i. The agent i chooses an action a_i^t depends on its state s_i^t and policy $\pi_i^{\emptyset_i}$ throughout the testing phase without critic.

$$a_i^t = \pi_i^{\emptyset_i}(s_i^t) \tag{6.18}$$

Critic network: The critic network adopted for approximating the action-value function $V(S)$ provides the overall reward while taking action a_i^t based on the state s_i^t and the global state g at time t in the training phase. Each agent executes the actions in the environment and sends the present state information s_i^t and response from the environment to the critic network after the actor-network in time t chooses the actions $a^t = \{a_1^t, a_2^t, \ldots, a_M^t\}$. The feedback consists of the next time instant state information g^{t+1} and reward R^t. Hence, Q-function defined as $Q_i^{\theta_i}(s_1^t, s_2^t, \ldots, s_M^t, a_1^t, a_2^t, \ldots, a_M^t)$ for each agent, which solves the problem caused by a not-stationary environment. Consider M agents with policy of agent i is π_i, then

$$\begin{aligned} P(s_i^{t+1}|s_i^t; Env) =& P(s_i^{t+1}|s_i^t; s_1^t, s_2^t, \ldots, s_M^t, a_1^t, a_2^t, \ldots, a_M^t, \\ & \pi_1, \pi_2, \ldots, \pi_M) \\ =& P(s_i^{t+1}|s_i^t; s_1^t, s_2^t, \ldots, s_M^t, a_1^t, a_2^t, \ldots, a_M^t, \\ & \pi_1', \pi_2', \ldots, \pi_M') \end{aligned} \tag{6.19}$$

By minimizing the loss function $\mathcal{L}(\theta_i)$, each critic updates its network and the loss function $\mathcal{L}(\theta_i)$ prarameterized by $\theta = \{\theta_1, \theta_2, \ldots, \theta_M\}$ defined as

$$\mathcal{L}(\theta_i) = \mathrm{E}_{s^t, a^t}\Big[\big(Q_i^{\theta_i}(s^t, a^t) - y^t\big)^2\Big] \tag{6.20}$$

where $s^t = \{s_1^t, s_2^t, \ldots, s_M^t\}$, $a^t = \{a_1^t, a_2^t, \ldots, a_M^t\}$ and

$$y^t = R^t + \gamma Q_i^{\theta_i}(g^{t+1}, a^t)|_{a_i^t = \pi_i^{\emptyset_i'}(s_i^{t+1})} \tag{6.21}$$

here $0 \leq \gamma < 1$ is the discount factor.

Update: Individual agent maximizes the reward by optimizing the policy directly where the policy parameterized by $\emptyset$. Therefore, the objective is to maximize the cumulative reward function.

$$J(\emptyset_i) = \mathrm{E}_{s^t, a^t}\Big[Q_i^{\theta_i}(s^t, a^t)|_{a_{s_i}^t = \pi_i^{\emptyset_i}(i)}\Big] \tag{6.22}$$

In the MARDDPG algorithm, the recurrent neural network LSTM added to the actor-network and critic network. Since the agents cannot communicate, the model takes a single frame in each time slot. Adding the LSTM enables a way to remember the last communication (the effect of the actions on reward) received from other agents. The actor-network and critic network historical information denoted by h_a^t and h_c^t. The individual agent chooses the action based on previous state h_i^t, i.e., $a_i^t = \pi_i^{\emptyset_i}(h_i^t)$, then the Q-function turn into $Q_i^{\theta_i}(h_c^t, a^t)$ where $h_c^t = \{h_{c,1}^t, h_{c,2}^t, \ldots, h_{c,M}^t\}$. Likewise, loss function $\mathcal{L}(\theta_i)$ in the critic network is

$$\mathcal{L}(\theta_i) = \mathrm{E}_{h_c^t, a^t}\Big[\Big(Q_i^{\theta_i}(h_c^t, a^t) - y_i^t\Big)^2\Big] \tag{6.23}$$

where

$$y_i^t = r_i^t + \gamma Q_i^{\theta_i'}\Big(h_c^{t+1}, \pi_1^{\emptyset_1'}(h_{a,1}^{t+1}), \ldots, \pi_M^{\emptyset_M'}(h_{a,M}^{t+1})\Big) \tag{6.24}$$

The objective function is denoted as

$$J(\emptyset_i) = \mathrm{E}_{h_c^t, a^t}\Big[Q_i^{\theta_i}\Big(h_c^{t+1}, \pi_1^{\emptyset_1}(h_{a,1}^{t+1}), \ldots, \pi_M^{\emptyset_M}(h_{a,M}^{t+1})\Big)\Big|_{a_i = \pi_i^{\emptyset_i}(h_{a,i}^t)}\Big] \tag{6.25}$$

The replay buffer stores the experience information in the training phase. The critic and actor networks updated by randomly sampled episodes from the replay buffer in each training step. The target critic and actor network parameters are denoted by θ' and $\emptyset'$ respectively. The target network is updated using soft updates.

6.3.2 Multi-Agent Recurrent DRL based Cooperative Caching Algorithm

The MEC node receives the user requests and obtains their features. It supplies the current request and caching state to the actor-network to get the caching actions. Following the action executing based on the policy, each agent receives the reward and the next state. Store the information received from the environment as history using LSTM.

$$\begin{aligned} h_a^{t+1} &= \text{LSTM}(h_a^t, s^{t+1}) \\ h_c^{t+1} &= \text{LSTM}(h_c^t, s^{t+1}, a^t) \end{aligned} \tag{6.26}$$

Then actor and critic network stores the experience in replay memory. To train the critic and actor-network, randomly select a mini-batch of the transitions from replay memory. For an individual sample, set the target critic network

$$y_{i,j}^t = r_{i,j}^t + \gamma Q_i^{\theta_i'}\left(h_{c,1}^{t+1,j}, \ldots, h_{c,M}^{t+1,j}, \pi_1^{\emptyset_1'}\left(h_{a,1}^{t+1,j}\right), \ldots, \pi_N^{\emptyset_N'}\left(h_{a,N}^{t+1,j}\right)\right) \tag{6.27}$$

and updates its parameter θ by reducing the loss function over mini batch

$$\mathcal{L}(\theta_i) = \frac{1}{S}\sum_{j \in S}\left(Q_i^{\theta_i}\left(h_{c,1}^{t,j}, \ldots, h_{c,M}^{t,j}, a_{1,j}^t, \ldots, a_{M,j}^t\right) - y_{i,j}^t\right)^2 \tag{6.28}$$

Furthermore, the actor computes the policy gradient leveraging the loss function and the parameter $\emptyset$ updated using the gradient over mini batch.

$$\begin{aligned} \nabla_{\emptyset_i} J(\emptyset_i) \approx \frac{1}{S}\sum_{j \in S} \nabla_{\emptyset_{i,\pi_i^{\emptyset_i}}}\left(h_{a,i}^{t,j}\right), \nabla_{a_i} Q_i^{\theta_i}\left(h_{c,1}^{t,j}, \ldots, h_{c,M}^{t,j}, \pi_1^{\emptyset_1}\right. \\ \left.\left(h_{a,1}^{t,j}\right), \ldots, \pi_N^{\emptyset_N}\left(h_{a,N}^{t,j}\right)\right) \end{aligned} \tag{6.29}$$

Update the critic and actor parameters of the target network

$$\begin{aligned} \theta'_i &\leftarrow \tau\theta_i + (1-\tau)\theta' \\ \emptyset'_i &\leftarrow \tau\emptyset_i + (1-\tau)\emptyset' \end{aligned} \tag{6.30}$$

Algorithm 1 summarizes the cooperative content caching mechanism based on MARDDPG. First, randomly initialize the critic network parameter θ and actor-network parameter $\emptyset$. Initialize the replay memory G and the target network with weights θ' and $\emptyset'$. In each episode, initialize the empty history $h^t_{a,i}, h^t_{c,i}$ and a random process for exploration. Lines 5 to 13 shows that MEC receives the requests and observe the state. The individual agent selects an action a^t depend on the policy $\pi_i^{\emptyset_i}(h^t_{a,i})$. After performing an action a^t, the agent gets the following state s^{t+1} information, and the reward R^t then stores the information collected from the environment in history using the LSTM network. Save the individual agent's experiences $(s^t_i, a^t_i, r^t_i | t = \{1, \ldots, T\})$ in the replay memory G to train actor and critic networks. Line 14 shows that each agent randomly samples a mini batch of S transitions $\{s^1_{i,j}, a^1_{i,j}, r^1_{i,j}, s^2_{i,j}, a^2_{i,j}, r^2_{i,j}, \ldots\}$ from the replay memory G to train the critic and actor-network. Lines 15 to 19 show that for each agent, the critic network estimates the Q-approximation for each sample $j \in S$, then compute the temporal difference-error and update its weights by minimizing the loss function $\mathcal{L}(\theta_i)$ over the target network. Further, the actor network computes the policy gradient $\nabla_{\emptyset_i} J(\emptyset_i)$ leveraging the loss function and update its weights by the average policy gradient over the target network. Then update the target network weights, content properties and cache state in lines 21 and 22.

6.4 Performance Evaluation

This section, validates the performance of the proposed MARDDPG based cooperative caching mechanism using simulations. Particularly, first, the simulation environment is mentioned along with performance metrics and reference algorithms. Furthermore, the performance of the MARDDPG mechanism is compared to the reference methods in terms of system parameters, and the simulation results are analyzed in detail.

The real-world Dataset MovieLens 1M Dataset [37] has been used in these simulations

Algorithm 6.1 MARDDPG based Content Caching Algorithm

Initialize the actor network $\pi_i^{\emptyset_i}(h_{a,i}^t)$ with random weights $\emptyset$ and the critic network $Q_i^{\theta_i}(h_c^t, a^t)|h_c^t = \{h_{c,1}^t, h_{c,2}^t, \ldots, h_{c,M}^t\}, a^t = \{a_1^t, a_2^t, \ldots, a_M^t\}$ with random weights θ

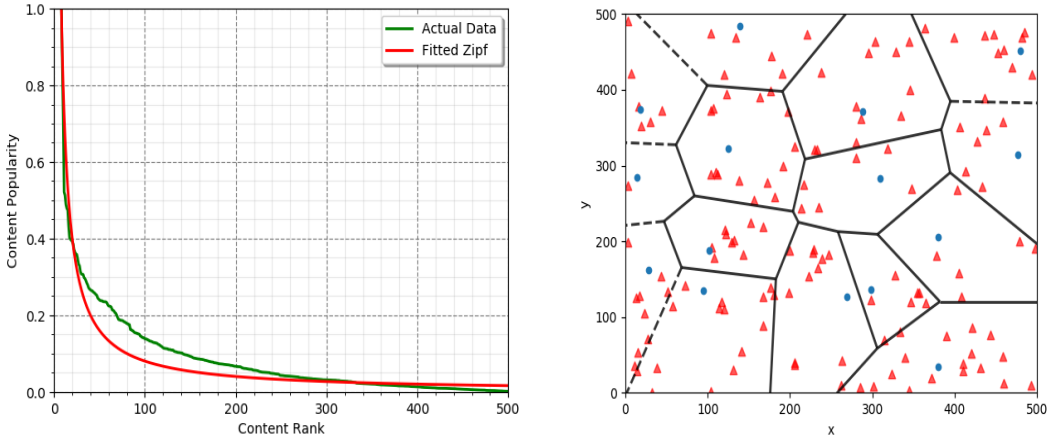
Initialize the target network $\pi_i^{\emptyset_i'}$ $Q_i^{\theta_i'}$ with weights $\emptyset'$ and θ'

Initialize the replay memory G of each agent to capacity F

Output: X: Content Placement.

for all episode **do**
  Initialize t = 1;
  Initialize a random process $\mathcal{M}$ for action exploration;
  Initialize empty history $h_{a,i}^0, h_{c,i}^0$;
  for $t \in T$ and $o^t \neq$ terminal **do**
    The MEC receives user requests W^t;
    Observe the cache state s_i^t of each agent i;
    For each agent i select an action $a_i^t = \pi_i^{\emptyset_i}(h_{a,i}^t)$ with respect to current policy and exploration noise;
    Execute action a^t, store the received reward r^t and new state s^{t+1} information in history using Eq. (6.26);
    $t = t + 1$;
  end for
  Store episode $(s_i^t, a_i^t, r_i^t | t = \{1, \ldots, T\})$ for all agents;
  for all $i \in \mathcal{M}$ **do**
    Randomly sample a mini batch of S episodes from replay memory $\{s_{i,j}^1, a_{i,j}^1, r_{i,j}^1, s_{i,j}^2, a_{i,j}^2, r_{i,j}^2, \ldots\}$ episodes from replay memory G;
    for $t = Tto1$ **do**
      Set target network Eq. (6.27);
      Minimizing the loss using Eq. (6.28) and update the critic network;
      Update the actor network policy using the sampled policy gradient Eq. (6.29);
    end for
  end for
  Update the target networks using Eq. (6.30);
  Update the content properties ψ_i and cache state s_i ;
end for

to investigate for requesting content. The MovieLens dataset consists of 3952 movies, 1000209 user ratings that take integer values [1 (worst), 5 (best)] and 6040 users. Each row of the dataset consists of userid, movieid, rating and timestamp. The rating information is considered the content request since the user rates a movie followed by watching it [131]. The rating information is considered as the frequency of movies requested by a user. Also, it is assumed that the number of requests for a movie within 10, 100 and 1000 requests as the features. Therefore, the top 600 popular content requested by users and the 100 most active users have been selected to analyze the user request statistics. More than 90%



(a) (b)

Figure 6.4: (a) Comparison of content popularity vs content rank Content popularity of Movielens dataset (b) Voronoi cell diagram with size $500m \times 500m$ where blue circle indicates the BSs and red triangles are mobile users.

of the ratings are from the first year in the dataset, so only the first year ratings has been considered for simulation. The skewness parameter $\alpha = 0.8$ is obtained by fitting the actual data from the dataset with the Zipf distribution as shown in Fig 6.4a.

This section presents the system setting to evaluate the performance of the proposed caching algorithm. A square region with an area of $500m \times 500m$ is considered. In the given simulation area, consider the Poisson point process (PPP) for base stations. The users are distributed within each base station coverage based on PPP shown in Fig. 6.4b. Six hundred contents with size determined uniformly at random from the range of [10MB to 100MB], 15 BSs and 90 users has been considered. Each content has a deadline picked randomly from [5 to 30s]. Each MEC can cache 10% of the total files. The latency to fetch content from the base station to the user is specified using uniform distribution ranges from [10 to 30s]. To obtain the file from the central server to MEC is taken as the 80s.

Python with the TensorFlow platform has been considered for implementing the proposed MARDDPG caching mechanism and implemented it on the open-source R-MADDPG package. The neural network model composes the evaluated actor and cretic network and the target actor and critic network for each agent. The evaluated critic and actor networks are similar to that of the target networks. The networks have three hidden layers with 64

neurons in each layer. The middle layer is an LSTM layer, and the other two layers are fully connected, where the first layer has a ReLU activation function. The target network is updated using the Adam optimizer, where the critic and actor networks learning rates are 0.001 and 0.0005, respectively, and the discount reward is 0.9. The capacity of the replay memory and the mini-batch size is considered as 10^5 and 256, respectively.

Table 6.2: Simulation Parameters

Parameters	**Values**
Simulation area	$500/m \times 500/m$
Number of users	90
Number of contents	600
Number of base stations	15
Content size	(10, 100] MB
The delay between BS and user	(5,25]s
The delay between BSs	20s
The delay between content server and BS	80s
The deadline of the content	(10,30] s
Actor and critic learning rate	0.001, 0.0005
Network update rate	0.01
Discount	0.9
Mini batch size	256
Replay memory capacity	10^5
Number of episodes	1500
Number of steps in each episode	100

6.4.1 Performance Metrics

To compare the performance of cache replacement schemes, the following metrics are considered:

1. *Cache Hit Ratio*: The fraction of requests served over the total requests.

2. *Acceleration ratio*: The fraction of saved delay and overall delay (from the controller)

3. *Caching Reward*: The reward measures the cumulative long-term reward collected from caching (i.e., Sum of the intermediate reward of all MECs) using Eq. (6.15).

4. *Local Hit*: The fraction of requests served within the MEC.

5. *Neighbouring Hit*: The fraction of requests served within the network and not within the MEC.

6.4.2 Reference Algorithms

In this section, the proposed algorithms is compared with the following caching mechanisms: Least Recently Used (LRU) [177], First In First Out (FIFO) [178], Least Frequently Used (LFU) [179], Multi-Agent Actor-Critic (MAC) [32] and Deep Reinforcement Learning (DRL) [16].

1. LRU (Least Recently Used): It is a recency based mechanism where the least recently requested file is updated with the fetched file when the cache is already full.

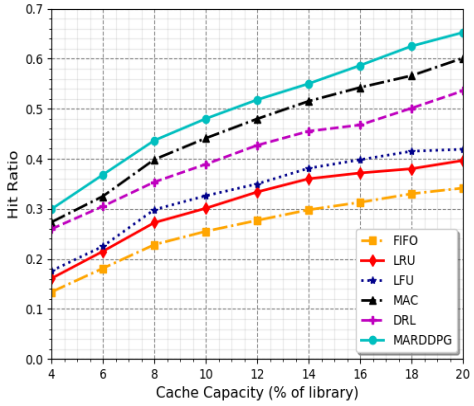
2. LFU (Least Frequently Used): It is a frequency-based mechanism where the least number of times requested file is updated with the fetched file when the cache is already full.

3. FIFO (First In First Out): FIFO is an arrival based mechanism where the earliest cached file is updated with the fetched file when the cache is already full.

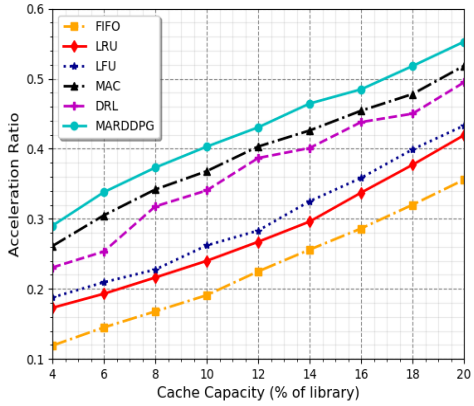
4. DRL (Deep Reinforcement Learning): DRL is a cache replacement decision mechanism where individual MEC performs the caching decisions individually with the help of local observations without considering the impact of other MECs.

5. MAC (Multi-Agent Actor-Critic): MAC is a cache replacement decision mechanism where the actor takes caching decisions and critic evaluates the policy. In this mechanism, communication between the agents is not considered; hence, there is no global state to process for actor networks.

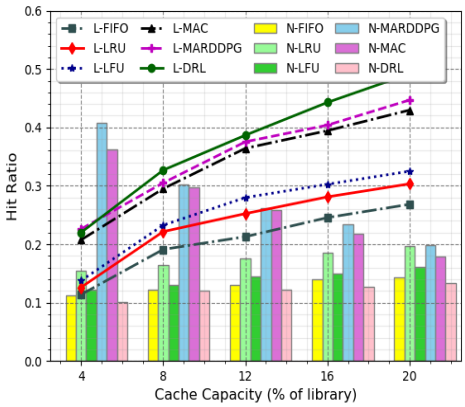
The first three cache replacement strategies update the content individually based on arrival, frequency and recency, whereas the other strategies consider deep reinforcement learning to place the contents. The fourth cache replacement strategy (DRL) is different from the fifth strategy (MAC) because, in DRL, the individual node is not aware of the other nodes'



(a)



(b)



(c)

Figure 6.5: Comparison of caching schemes using cache capacity vs (a) Cache Hit Ratio (b) Acceleration Ratio (c) Local and Neighbour Cache Hit Ratio.

information, and the former is the value-based RL, and the latter is policy-based RL. Moreover, a multi agent deep deterministic policy gradient algorithm (i.e., MADDPG algorithm [176]) has been considered for comparison.

6.4.3 Impact of Cache Size

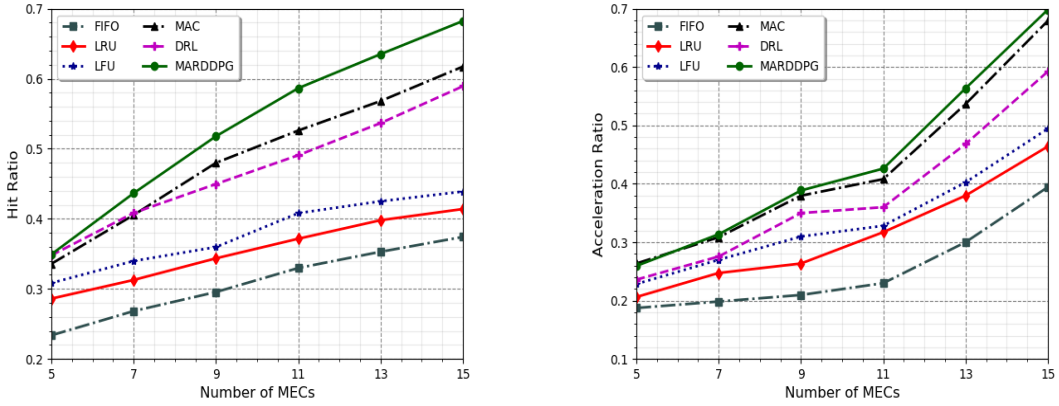
The impact of cache size on acceleration ratio and cache hit ratio is shown in Figure 6.5. In this simulations, the number of MECs is 15, skewness parameter is 0.8 and the cache capacity varies from 4% to 20% total library size with step size 2.

In Fig. 6.5a, the impact of cache size on the cache hit ratio is presented. The curves

indicate an upward trend with the rise in cache size since the large cache size allows MECs to cache more content, allowing them to satisfy more user requests from local or neighbouring MECs. The learning-based mechanisms show superiority over conventional rule-based replacement mechanisms since the learning-based mechanisms capture the user request features from the historical data. MARDDPG has better performance than MAC and DRL. The DRL mechanism does not consider the cooperation among the nodes where each node tries to maximize its reward without concern about other nodes. MAC considers no cooperation between agents even though it considers the multi-agent framework. The proposed MARDDPG mechanism provide improvement of up to 24, 19, 17, 9 and 4 % on hit ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

In Fig. 6.5b, the impact of cache size on the acceleration ratio is presented. The proposed MARDDPG mechanism provide improvement of up to 20, 15, 13, 6 and 3 % on acceleration ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

In Fig. 6.5c, the impact of the cache size on the local and neighbouring cache hit is presented. The local hit rate is denoted with 'L' and the neighbouring hit rate denoted with 'N'. The local hit ratio of the learning-based algorithms has superiority over rule-based mechanisms. The reason is that the learning-based mechanisms perform the cache replacement decision depend on the history of the data that enables the nodes to cache more popular content locally to the MEC and moderately popular content cached at neighbouring nodes. The upward trend indicates that the rise in cache size improves the local hit rate. It can be noticed that the DRL has higher local hit ratio than MAC and MARDDPG and lower neighbour hit ratio because each agent in DRL cache based on local cache information leading to redundant content at each agent. It can also be noticed that MAC and the proposed mechanisms have more neighbouring hit rate with less capacity and decreases as the capacity increases. The reason is that both the MAC and proposed mechanisms considers the cooperation among the agents leading to higher neighbour hit ratio. Overall, both MAC and MARDDPG have a better cache hit ratio than DRL. Further, it can be observed that the proposed mechanism may sacrifice the local hit rate, but it satisfies more user requests with a small delay than all other baseline algorithms.



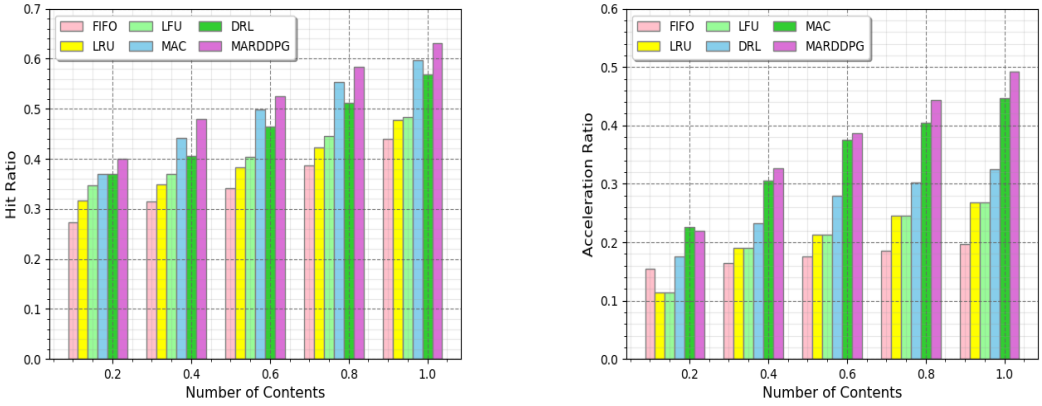
(a) (b)

Figure 6.6: Comparison of caching schemes using number of MECs vs (a) Cache Hit Ratio (b) Acceleration Ratio.

6.4.4 Impact of Number of MECs

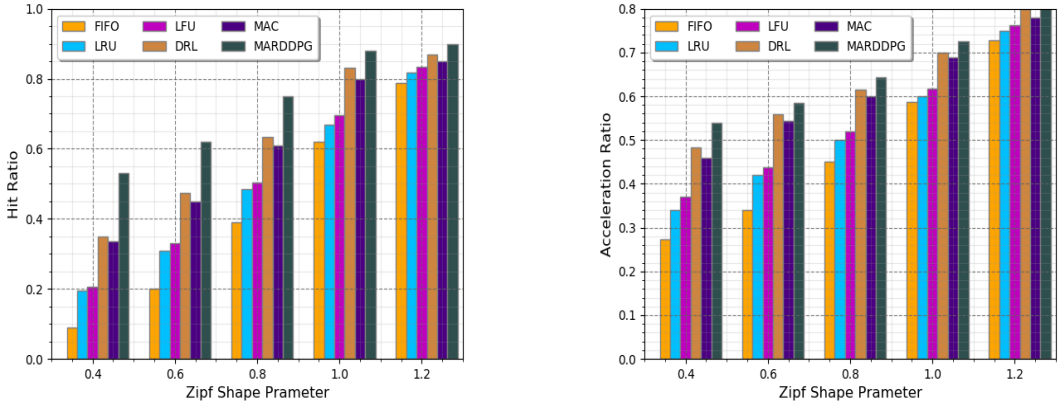
The impact of the number of MECs on the cache hit ratio and acceleration ratio is shown in Figure 6.6. In this simulations, the cache capacity is 10%, skewness parameter is 0.8 and the number of MECs varies from 5 to 15 with step size 2.

It can be seen the effect of the cache hit ratio with a varying number of MECs in Figure 6.6a. It can be noticed that the proposed mechanism shows clear superiority over other baseline mechanisms since it uses the cooperative mechanism and learning the user request pattern from the history data. The conventional rule-based mechanism is performing less than the learning-based mechanisms. The proposed MARDDPG mechanism provide improvement of up to 22.5, 18, 15, 6 and 4.5 % on hit ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively. From Fig. 6.6b, the acceleration ratio grows slowly with less number of MECs is less, and rapidly increases as MECs increases. The propose mechanism outperforms other mechanisms. The proposed MARDDPG mechanism provide improvement of up to 19, 13,10, 6 and 1 % on acceleration ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.



(a) (b)

Figure 6.7: Comparison of caching schemes using number of contents vs (c) Cache Hit Ratio (d) Acceleration Ratio.



(a) (b)

Figure 6.8: Comparison of caching schemes using Zipf shape parameter vs (a) Cache Hit Ratio (b) Acceleration Ratio.

6.4.5 Impact of Number of Contents

The impact of contents on cache hit ratio and acceleration ratio is shown in Figure 6.7. In this simulations, the number of MECs is 15, skewness parameter is 0.8, cache capacity is 10% total library size and number of contents varies from 0.2 to 1.0 with step size 0.2.

In Fig. 6.7a, the impact of number of contents on the cache hit ratio is presented. The bars indicate an upward trend as contents rise. More popular content need to be cached at

MECs, leading to frequent cache replacement due to limited cache capacity. It can be seen that MARDDPG has better performance than other reference algorithms. The DRL mechanism does not consider the cooperation among the nodes where each node tries to maximize its reward without concern about other nodes leading to less cache hit ratio than MAC and MARDDPG. MAC considers no cooperation between agents even though it considers the multi-agent framework. The proposed MARDDPG mechanism provide improvement of up to 20, 17, 17, 11 and 2.2 per cent on hit ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively. It can be noticed from Fig. 6.7b, that the conventional replacement mechanism grows slowly compared to the learning-based mechanisms. Because the learning-based mechanism captures the content popularity, enabling the MEC to replace the less popular content, leading to more saved delay. The proposed mechanism outperforms other baseline algorithms. The proposed MARDDPG mechanism provide improvement of up to 17, 13, 11, 6 and 3 % on acceleration ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

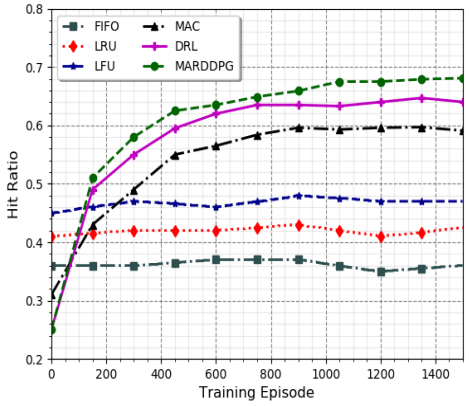
6.4.6 Impact of Zipf parameter

Them impact of Zipf parameter on cache hit ratio and accelerated ratio is presented in Figure 6.8a and 6.8b respectively. The upward trend of all the algorithms indicates that more requests are for few contents as the Zipf skewness parameter rises; this leads to an increase in the hit ratio and allows access to the requested content within a smaller delay improves the acceleration ratio.

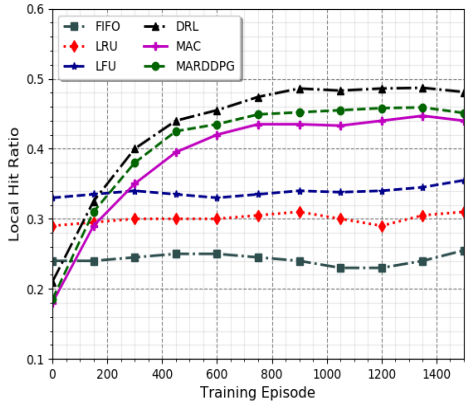
6.4.7 Performance evaluation with training episode

In Fig. 6.9, the performance comparison of cache hit ratio, local and neighbour cache hit ratio is presented. In this simulations, the number of MECs are 15, skeness parameter is 0.8, cache capacity is 10%.

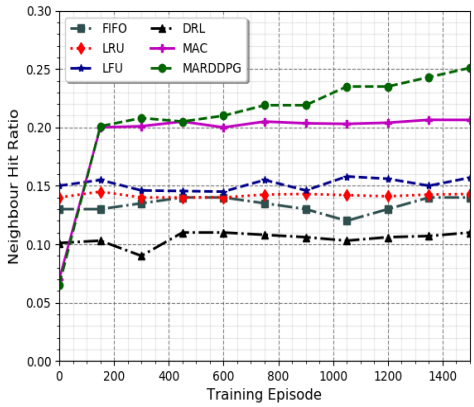
From Fig. 6.9a, it can be seen that the rule-based cache replacement mechanism shows a relatively stable hit ratio since they have not considered real-time continuous learning from the environment. The learning-based algorithms curves indicate an upward trend and



(a)



(b)



(c)

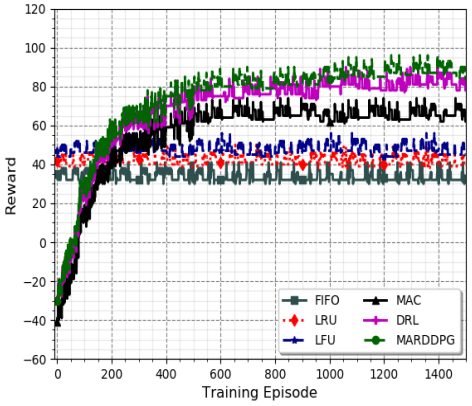
Figure 6.9: Comparison of caching schemes using training episode vs (a) Cache Hit Ratio (b) Local Cache Hit Ratio (c) Neighbour Cache Hit Ratio.

stabilize after that. The DRL has a higher hit ratio than MAC and MARDDPG initially, but as the episodes increase, it slowly diminishes. That is because the DRL is a non-cooperative cache replacement mechanism where each agent performs the cache replacement based on local information, not considering the other agents' information in caching decision. Therefore, each agent may cache content redundantly leads to obtain more content from the content server. In MAC, the agents cache the content based on the central controller, which cooperates with communication overhead. The proposed MARDDPG outperforms the other mechanisms since it uses the LSTM to learn the better policy to cache more popular content.

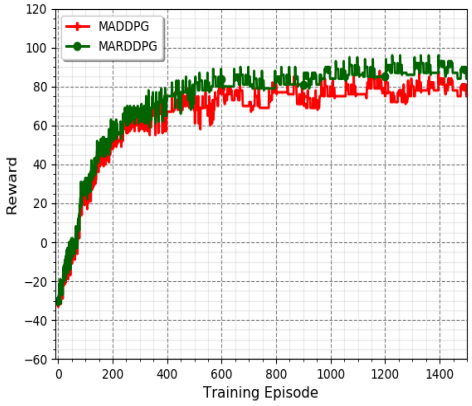
Fig. 6.9b and 6.9c show that the rule-based mechanisms have more local hit ratios and fewer neighbour-hit ratios. The reason is that the rule-based mechanism not consider the learning from the environment; therefore, each MEC caches the redundant content leads to fetch and replace more content compared to other mechanisms. It can also noticed that the DRL mechanism has a lower neighbour hit ratio and higher local cache hit ratio since each agent in DRL caches based on the local cache state. The MAC and MARDDPG have lower local hit rate than DRL, but both mechanisms have better neighbour hit ratio. Overall, both the MAC and MARDDPG have a better cache hit ratio than DRL, even though they sacrifice a little local cache hit ratio. It can be observed that the proposed MARDDPG has an improved neighbour hit ratio than all baseline algorithms since it learns the better policy by using LSTM leads to cache more popular content near users instead of fetching from the distant content server.

6.4.8 The convergence performance

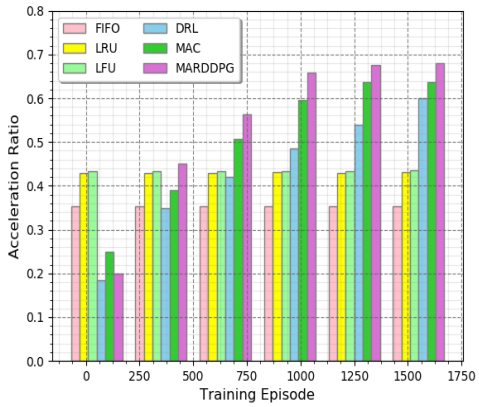
From Fig. 6.10a, it can be noticed that the rule-based replacement mechanisms like FIFO, LRU and LFU fluctuate around 33, 42 and 46, respectively, and do not increase training episodes. The reason is that the rule-based mechanism cannot learn from the environment. It can be seen that the DRL has an upward trend in the first 600 episodes and fluctuates around reward 62. In DRL, each agent learns from its local cache state, leading to more difficulty learning optimal strategy. The MAC raise slowly till 400 episodes and constantly fluctuates around reward 80 since the centralized controller simultaneously controls the multiple agents in the environment leading to slow convergence. It can be noticed that the proposed MARDDPG curve increases quickly and fluctuates around reward 87, which outperforms all other references algorithms. Thus, the MARDDPG finds the best strategy quickly compared to other reference mechanisms in maximizing saved delay. Compared to DRL and MAC, MARDDPG has a 12 and 15 per cent increase in reward. From Fig. 6.10b, It can be observed that both the curves have a similar trend. Both the curves raise quickly till episode 400 and then stabilizes slowly. It can be noticed that even though both curves have a similar trend, the proposed MARDDPG mechanism has a higher reward and



(a)



(b)



(c)

Figure 6.10: (a) Reward of all schemes vs Training episode (b) Reward of proposed and MADDPG schemes during Training episodes (c) Training episode vs Acceleration Ratio.

more stability than the MADDPG. The reason is that the inclusion of LSTM enables the agents in the MARDDPG algorithm to learn a better policy compared to MADDPG, where the LSTM has not considered. Overall it can be noticed that the proposed MARDDPG outperforms all other reference algorithms by considering the LSTM for learning a better strategy to maximize the reward cooperatively among the agents.

In Fig. 6.10c, the impact of training episodes on the acceleration ratio is presented. It can be seen that the rule-base mechanism does not increase as the training episode increase. The proposed mechanism raises quickly and stabilizes after episode 1000. That indicates that the MARDDPG learns a better policy quickly. That means the MARDDPG caches

the more popular content cooperatively with fewer replacements leads to more saved delay. The DRL and MAC have less acceleration ratio since in DRL agent updates policy based on local cache state without cooperation. As in MAC, a centralized coordinator updates the agents' policy simultaneously.

6.5 Summary

This chapter considers caching in the multi-cell scenario. Specifically, a MARDDPG algorithm has been designed to maximize the saved delay in the cooperative mobile edge networks. The LSTM model is integrated in MADDPG to design the cooperative caching algorithm for multi-cell scenarios and discussed the network update in detail. Extensive simulations are performed to determine the performance of the proposed algorithm over existing algorithms. The proposed cooperative cache update algorithm outperforms the existing algorithms by considering performance metrics such as the cache hit ratio, acceleration ratio and reward. The proposed mechanism is shown an improvement over other learning-based and non-learning (rule-based) based algorithms.

Chapter 7

Conclusion and Future Directions

This thesis investigates the design and development of caching algorithms, which reduce the load on backhaul links and congestion in the mobile edge networks by placing appropriate content near the users. Different caching mechanisms that maximize the saved delay when placing the content at the edge nodes in MEN are presented. The proposed caching mechanisms achieve better performance in terms of the cache hit ratio, acceleration ratio, and cache utilization. Performance evaluations have been done to show the efficacy of the proposed algorithms. A comparative study of the proposed protocols has been presented and discussed through several experiments in order to demonstrate their merits and capabilities.

This thesis addresses the main challenges of mobile edge networks (MENs), such as uneven distribution of users, heterogeneity of user preferences, delay sensitivity, mobility, randomness of contact duration, cache utilization improvement, and dynamic content popularity information. The rapid growth in time-critical and delay-sensitive applications like video streaming, Internet of Things (IoT), and financial applications need a response within a deadline. If a request is not served within the deadline, the quality of service would be affected and this affects the user QoE. Hence, to improve the user QoE, the request deadlines must be satisfied. Furthermore, the user mobility and dynamic content popularity imposed additional challenges. In this thesis, contributions have been made by considering the main challenges, such as the deadline of the content, heterogeneity of user preferences, user mobility, and dynamic content popularity in making efficient caching decisions in MEN.

7.1 The Major Contributions of the Thesis

A cache placement problem in mobile edge networks aiming to maximize saved delay by considering the capacity and deadline constraints has been addressed in Chapter 3. The proposed approach improves the acceleration ratio, cache hit ratio and cache utilization. The echo state network is applied to predict content request distribution and a fuzzy caching mechanism is designed based on the predicted content popularity, benefit and deadline.

The clustered cooperative cache placement has been analyzed in large-scale mobile edge networks, aiming to maximize the saved delay by considering the heterogeneity of user preferences, activity level, and uneven user distribution in Chapter 4. The dynamic user behaviour is learned using LSTM model. The users are clustered based on the content based clustering mechanism to cache appropriate content near to users. An efficient greedy mechanism is designed to solve the cache placement problem. The relation between user preferences and local and global content popularity has been analyzed. The proposed mechanism improves the cache hit ratio, acceleration ratio and cache utilization.

In Chapter 5, the impact of user mobility and contact duration on cache placement in mobile edge networks aiming to maximize the saved delay by considering the capacity constraint have been analyzed. The user mobility is modeled as a Markov renewal process to predict the contact duration and the moving path. An effective greedy algorithm is designed to solve the formulated problem. Further, a heuristic search mechanism based on a genetic algorithm is proposed to solve a large scale problem. The proposed mechanism shows improvement in terms of the cache hit ratio and acceleration ration in a mobility based scenario.

In Chapter 6, an efficient deep reinforcement learning algorithm has been designed for cooperative mobile edge networks in the absence of content popularity. In the proposed mechanism LSTM is applied to remember the last communication received from other agents. The proposed mechanism is shown an improvement over other learning-based and non-learning (rule-based) based algorithms. The delay is minimized by replacing the fetched content with the appropriate content at the base station.

7.2 Future Directions

Although the proposed caching algorithms show promising performance improvements as compared to the existing relevant mechanisms available in the literature, there are other aspects and scenarios which could be considered. Some of the potential extensions of our research work presented in this thesis are listed as follows:

Caching and computation offloading play a vital role in improving the user quality of experience. However, providing an efficient method for joint caching and computation offloading decisions to improve network resource utilization and performance for delay-sensitive applications are challenging in large scale mobile edge networks. Further, designing an efficient cache strategy by considering the user mobility into the joint caching and computing in mobile edge networks is more practical.

Research can be extended to investigate the device-to-device communication based incentive-based cooperative caching strategy in mobile edge networks. In Chapter 4, uneven distribution of users and heterogeneous user preferences are considered in the static network setting. However, users may move across the base stations at varying speeds. Research may be extended to investigate caching strategy based on instant cell load information and varied channel conditions in large scale mobile edge networks. In Chapter 5, the contact duration aware cooperative caching mechanism is presented. The problem presented in Chapter 5 can be further investigated by analyzing the mobility aware device-to-device communication in hybrid mobile networks. The dynamic nature of the mobile networks, content popularity, user position and user preferences play a vital role in decision making. Research can be extended by providing an efficient learning-based cooperative caching scheme for big data-driven applications to improve the cache performance.

Appendix

Proof of Lemma 1

Then the benefit of adding $n^r_{f,k}$ to A is

$$g(A \cup n^r_{f,k}) = \frac{1}{U} \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{V}} p_f P^v_u \sum_{f \in \mathcal{F}} \min\{x^r_f + 1, \delta^r_u \frac{\beta_r}{B_f}\} + \sum_{j \in R, j \neq r} \min\{x^j_f, \delta^j_u \frac{\beta_j}{B_f}\}(d^m_u - d^r_u) \quad (7.1)$$

Here, $\min\{x^r_f + 1, \delta^r_u \frac{\beta_r}{B_f}\} \geq \min\{x^r_f, \delta^r_u \frac{\beta_r}{B_f}\}$ shows that the marginal benefit gained by adding an element $n^r_{f,k}$ to the caching scheme A is non-negative. The monotone submodularity of function $g(A)$ is proved by satisfying the submodular property (5.14). Therefore, $g(A \cup n^r_{f,k}) - g(A) \geq g(B \cup n^r_{f,k}) - g(B) \geq 0$. We can obtain this $x^r_{f,k} < (x^r_{f,k} + 1) \leq b^r_{f,k} < (x^r_{f,k} + 1)$ since $A \subset B \subset N$ then $x^r_f \subset b^r_f$. The difference of marginal values of $(g(A \cup n^r_{f,k}) - g(A)) \; - \; (g(B \cup n^r_{f,k}) - g(B))$ is given as

$$g(A \cup n^r_{f,k}) - g(A) = \frac{1}{U} \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{V}} p_f P^v_u D \sum_{f \in \mathcal{F}} \times \Big[\min\{x^r_f + 1, \omega\} + \; \Delta(A) - \min\{x^r_f, \omega\} + \; \Delta(A)\Big] \quad (7.2)$$

where $\Delta(A) = \sum_{j \in R, j \neq r} \min\{x^j_f, \delta^j_u \frac{\beta_j}{B_f}\}, D = (d^m_u - d^r_u)$ and $\omega = \delta^r_u \frac{\beta_r}{B_f}$. The difference of marginal values of $g(A \cup n^r_{f,k}) - g(A) \; - \; g(B \cup n^r_{f,k}) - g(B)$ is given as

$$\frac{1}{U} \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{V}} p_f P^v_u D \sum_{f \in \mathcal{F}} \times \Big[\min\{x^r_f + 1, \omega\} + \; \Delta(A) - \min\{x^r_f, \omega\} + \; \Delta(A)\Big] - \Big[\min\{b^r_f + 1, \omega\} + \; \Delta(B) + \min\{b^r_f, \omega\} + \; \Delta(B)\Big] \quad (7.3)$$

this can be written as

$$\frac{1}{U}\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{V}} p_f P_u^v D \sum_{f\in\mathcal{F}} \alpha \quad (7.4)$$

where $\alpha = \Big[\min\{x_f^r+1,\omega\} + \Delta(A) - \min\{x_f^r,\omega\} + \Delta(A)\Big]$
$-\Big[\min\{b_f^r+1,\omega\} + \Delta(B) + \min\{b_f^r,\omega\} + \Delta(B)\Big]$ Based on the content cached and range of the MECs there exist five cases. Case I $= (\omega \le x_{f,k}^r)$, Case II $= (x_{f,k}^r < \omega \le x_{f,k}^r+1)$, Case III $= (x_{f,k}^r+1 < \omega \le b_{f,k}^r)$, Case IV $= (b_{f,k}^r < \omega \le b_{f,k}^r+1)$ and Case V $= (b_{f,k}^r+1 \le \omega)$

$$\begin{cases} \Big[\{\omega+\Delta(A)\} - \{\omega+\Delta(A)\} - \{\omega+\Delta(B)\} + \{\omega+\Delta(B)\}\Big] = 0, & \text{I} \\ \Big[\{\omega+\Delta(A)\} - \{x_{f,k}^r+\Delta(A)\} - \{\omega+\Delta(B)\} + \{\omega+\Delta(B)\}\Big] = \omega - x_{f,k}^r, & \text{II} \\ \Big[\{x_{f,k}^r+1+\Delta(A)\} - \{x_{f,k}^r+\Delta(A)\} - \{\omega+\Delta(B)\} + \{\omega+\Delta(B)\}\Big] & \\ = x_{f,k}^r+1+\Delta(A) - (x_{f,k}^r+\Delta(A)) = 1, & \text{III} \\ \Big[\{x_{f,k}^r+1+\Delta(A)\} - \{x_{f,k}^r+\Delta(A)\} - \{\omega+\Delta(B)\} + \{b_{f,k}^r+\Delta(B)\}\Big] & \\ = \omega - b_{f,k}^r + 1, & \text{IV} \\ \Big[\{x_{f,k}^r+1+\Delta(A)\} - \{x_{f,k}^r+\Delta(A)\} - \{b_{f,k}^r+1+\Delta(B)\} & \\ +\{b_{f,k}^r+\Delta(B)\}\Big] = 0, & \text{V} \end{cases} \quad (7.5)$$

From the all cases we can observe that the $\frac{1}{U}\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{V}} p_f P_u^v D \sum_{f\in\mathcal{F}} \alpha \ge 0$ since $\alpha \ge 0$. Therefore, $g(A\cup n_{f,k}^r) - g(A) \ge g(B\cup n_{f,k}^r) - g(B)$. with this the monotone submodularity of (5.16) is proved.

Bibliography

[1] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, 2019.

[2] Cisco Systems Inc. Cisco visual networking index: Global mobile data traffic forecast update, 2017-–2022. *White Paper*, Feb 2019.

[3] Jingjing Yao, Tao Han, and Nirwan Ansari. On mobile edge caching. *IEEE Communications Surveys & Tutorials*, 21(3):2525–2553, 2019.

[4] Li Qiu and Guohong Cao. Popularity-aware caching increases the capacity of wireless networks. *IEEE Transactions on Mobile Computing*, 19(1):173–187, 2019.

[5] Wenlu Hu, Ying Gao, Kiryong Ha, Junjue Wang, Brandon Amos, Zhuo Chen, Padmanabhan Pillai, and Mahadev Satyanarayanan. Quantifying the impact of edge computing on mobile applications. In *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*, pages 1–8, 2016.

[6] Amardeep Mehta, William Tärneberg, Cristian Klein, Johan Tordsson, Maria Kihl, and Erik Elmroth. How beneficial are intermediate layer data centers in mobile edge networks? In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, pages 222–229. IEEE, 2016.

[7] Meng Zhang, Hongbin Luo, and Hongke Zhang. A survey of caching mechanisms in information-centric networking. *IEEE Communications Surveys & Tutorials*, 17(3):1473–1499, 2015.

[8] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5):1728–1739, 2016.

[9] Swaroop Nunna, Apostolos Kousaridas, Mohamed Ibrahim, Markus Dillinger, Christoph Thuemmler, Hubertus Feussner, and Armin Schneider. Enabling real-time context-aware collaboration through 5g and mobile edge computing. In *2015 12th International Conference on Information Technology-New Generations*, pages 601–605. IEEE, 2015.

[10] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, volume 1, pages 126–134. IEEE, 1999.

[11] Tuyen X Tran, Duc V Le, Guosen Yue, and Dario Pompili. Cooperative hierarchical caching and request scheduling in a cloud radio access network. *IEEE Transactions on Mobile Computing*, 17(12):2729–2743, 2018.

[12] Sampa Sahoo, Bibhudatta Sahoo, and Ashok Kumar Turuk. A learning automata-based scheduling for deadline sensitive task in the cloud. *IEEE Transactions on Services Computing*, 2019.

[13] Surbhi Saraswat, Hari Prabhat Gupta, Tanima Dutta, and Sajal K Das. Energy efficient data forwarding scheme in fog based ubiquitous system with deadline constraints. *IEEE Transactions on Network and Service Management*, 2019.

[14] Mohammed S ElBamby, Mehdi Bennis, Walid Saad, and Matti Latva-Aho. Content-aware user clustering and caching in wireless small cell networks. In *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 945–949. IEEE, 2014.

[15] Mingzhe Chen, Walid Saad, Changchuan Yin, and Mérouane Debbah. Echo state networks for proactive caching in cloud-based radio access networks with mobile users. *IEEE Transactions on Wireless Communications*, 16(6):3520–3535, 2017.

[16] Hao Zhu, Yang Cao, Wei Wang, Tao Jiang, and Shi Jin. Deep reinforcement learning for mobile edge caching: Review, new features, and open issues. *IEEE Network*, 32(6):50–57, 2018.

[17] David B Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical programming*, 62(1-3):461–474, 1993.

[18] Zheng Chen, Jemin Lee, Tony QS Quek, and Marios Kountouris. Cooperative caching and transmission design in cluster-centric small cell networks. *IEEE Transactions on Wireless Communications*, 16(5):3401–3415, 2017.

[19] Ming-Chun Lee, Andreas F Molisch, Nishanth Sastry, and Aravindh Raman. Individual preference probability modeling and parameterization for video content in wireless caching networks. *IEEE/ACM Transactions on Networking*, 27(2):676–690, 2019.

[20] Jie Yang, Yuanyuan Qiao, Xinyu Zhang, Haiyang He, Fang Liu, and Gang Cheng. Characterizing user behavior in mobile internet. *IEEE transactions on emerging topics in computing*, 3(1):95–106, 2014.

[21] Konstantinos Poularakis and Leandros Tassiulas. Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks. *IEEE Transactions on Mobile Computing*, 16(3):675–687, 2016.

[22] Rui Wang, Jun Zhang, SH Song, and Khaled B Letaief. Mobility-aware caching in d2d networks. *IEEE Transactions on Wireless Communications*, 16(8):5001–5015, 2017.

[23] Zongqing Lu, Xiao Sun, and Thomas La Porta. Cooperative data offloading in opportunistic mobile networks. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[24] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. Femtocaching: Wireless content delivery through distributed caching helpers. *IEEE Transactions on Information Theory*, 59(12):8402–8413, 2013.

[25] Fengxian Guo, Heli Zhang, Hong Ji, Xi Li, and Victor CM Leung. An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. *IEEE/ACM Transactions on Networking*, 26(6):2651–2664, 2018.

[26] Seungseob Lee, SuKyoung Lee, Kyungsoo Kim, and Yoon Hyuk Kim. Base station placement algorithm for large-scale lte heterogeneous networks. *PloS one*, 10(10):e0139190, 2015.

[27] Zhe Li and Gwendal Simon. In a telco-cdn, pushing content makes sense. *IEEE Transactions on Network and Service Management*, 10(3):300–311, 2013.

[28] Ejder Baştuğ, Marios Kountouris, Mehdi Bennis, and Mérouane Debbah. On the delay of geographical caching methods in two-tiered heterogeneous networks. In *2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2016.

[29] Shuo Wang, Xing Zhang, Yan Zhang, Lin Wang, Juwo Yang, and Wenbo Wang. A survey on mobile edge networks: Convergence of computing, caching and communications. *Ieee Access*, 5:6757–6779, 2017.

[30] Wei Jiang, Gang Feng, and Shuang Qin. Optimal cooperative content caching and delivery policy for heterogeneous cellular networks. *IEEE Transactions on Mobile Computing*, 16(5):1382–1393, 2016.

[31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[32] Chen Zhong, M Cenk Gursoy, and Senem Velipasalar. Deep reinforcement learning-based edge caching in wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, 6(1):48–61, 2020.

[33] Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement Learning*, pages 387–414. Springer, 2012.

[34] Yang Guan, Yao Xiao, Hao Feng, Chien-Chung Shen, and Leonard J Cimini. Mobicacher: Mobility-aware content caching in small-cell networks. In *2014 IEEE Global Communications Conference*, pages 4537–4542. IEEE, 2014.

[35] Tuyen X Tran, Fatemeh Kazemi, Esmaeil Karimi, and Dario Pompili. Mobee: Mobility-aware energy-efficient coded caching in cloud radio access networks. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 461–465. IEEE, 2017.

[36] Marvin McNett and Geoffrey M Voelker. Access and mobility of wireless pda users. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(2):40–55, 2005.

[37] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

[38] Brian McFee, Thierry Bertin-Mahieux, Daniel PW Ellis, and Gert RG Lanckriet. The million song dataset challenge. In *Proceedings of the 21st International Conference on World Wide Web*, pages 909–916, 2012.

[39] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 18(3):1617–1655, 2016.

[40] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7), 2012.

[41] Hamidreza Shariatmadari, Rapeepat Ratasuk, Sassan Iraji, Andrés Laya, Tarik Taleb, Riku Jäntti, and Amitava Ghosh. Machine-type communications: current status and future perspectives toward 5g systems. *IEEE Communications Magazine*, 53(9):10–17, 2015.

[42] Mahadev Satyanarayanan. Mobile computing: the next decade. *ACM SIGMOBILE Mobile Computing and Communications Review*, 15(2):2–10, 2011.

[43] Ying Gao, Wenlu Hu, Kiryong Ha, Brandon Amos, Padmanabhan Pillai, and Mahadev Satyanarayanan. Are cloudlets necessary? *School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-139*, page 8, 2015.

[44] Waleed Ali, Siti Mariyam Shamsuddin, Abdul Samad Ismail, et al. A survey of web caching and prefetching. *Int. J. Advance. Soft Comput. Appl*, 3(1):18–44, 2011.

[45] Xuan Liu, Zhuo Li, Peng Yang, and Yongqiang Dong. Information-centric mobile ad hoc networks and content routing: a survey. *Ad Hoc Networks*, 58:255–268, 2017.

[46] George Xylomenos, Christopher N Ververidis, Vasilios A Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V Katsaros, George C Polyzos, et al. A survey of information-centric networking research. *IEEE Communications Surveys and Tutorials*, 16(2):1024–1049, 2014.

[47] Stefan Podlipnig and Laszlo Böszörmenyi. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, 35(4):374–398, 2003.

[48] Kai Jiang, Huan Zhou, Xin Chen, and Haijun Zhang. Mobile edge computing for ultra-reliable and low latency communications. *IEEE Communications Standards Magazine*, 2021.

[49] Xi Peng, Juei-Chin Shen, Jun Zhang, and Khaled B Letaief. Backhaul-aware caching placement for wireless networks. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[50] Xiaofei Wang, Min Chen, Tarik Taleb, Adlen Ksentini, and Victor CM Leung. Cache in the air: Exploiting content caching and delivery techniques for 5g systems. *IEEE Communications Magazine*, 52(2):131–139, 2014.

[51] Ejder Baştuğ, Mehdi Bennis, Engin Zeydan, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, and Mérouane Debbah. Big data meets telcos: A proactive caching perspective. *Journal of Communications and Networks*, 17(6):549–557, 2015.

[52] Engin Zeydan, Ejder Bastug, Mehdi Bennis, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, and Mérouane Debbah. Big data caching for networking: Moving from cloud to edge. *IEEE Communications Magazine*, 54(9):36–42, 2016.

[53] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. Mobile edge computing: Survey and research outlook. *arXiv preprint arXiv:1701.01090*, 2017.

[54] Mingyue Ji, Giuseppe Caire, and Andreas F Molisch. Fundamental limits of caching in wireless d2d networks. *IEEE Transactions on Information Theory*, 62(2):849–869, 2015.

[55] Abdallah Khreishah and Jacob Chakareski. Collaborative caching for multicell-coordinated systems. In *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 257–262. IEEE, 2015.

[56] Mohammad Ali Maddah-Ali and Urs Niesen. Coding for caching: fundamental limits and practical challenges. *IEEE Communications Magazine*, 54(8):23–29, 2016.

[57] Negin Golrezaei, Andreas F Molisch, Alexandros G Dimakis, and Giuseppe Caire. Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution. *IEEE Communications Magazine*, 51(4):142–149, 2013.

[58] Dong Liu and Chenyang Yang. Energy efficiency of downlink networks with caching at base stations. *IEEE Journal on Selected Areas in Communications*, 34(4):907–922, 2016.

[59] Min Sheng, Weijia Han, Chuan Huang, Jiandong Li, and Shuguang Cui. Video delivery in heterogenous crans: architectures and strategies. *IEEE Wireless Communications*, 22(3):14–21, 2015.

[60] Hasti Ahlehagh and Sujit Dey. Video-aware scheduling and caching in the radio access network. *IEEE/ACM Transactions on Networking (TON)*, 22(5):1444–1462, 2014.

[61] Jingxiong Gu, Wei Wang, Aiping Huang, and Hangguan Shan. Proactive storage at caching-enable base stations in cellular networks. In *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1543–1547. IEEE, 2013.

[62] Syed Ali Raza Zaidi, Mounir Ghogho, and Desmond C McLernon. Information centric modeling for two-tier cache enabled cellular networks. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 80–86. IEEE, 2015.

[63] Zheng Chang, Yunan Gu, Zhu Han, Xianfu Chen, and Tapani Ristaniemi. Context-aware data caching for 5g heterogeneous small cells networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

[64] Pol Blasco and Deniz Gündüz. Learning-based optimization of cache content in a small cell base station. In *2014 IEEE International Conference on Communications (ICC)*, pages 1897–1903. IEEE, 2014.

[65] Konstantinos Poularakis, George Iosifidis, and Leandros Tassiulas. Approximation algorithms for mobile data caching in small cell networks. *IEEE Transactions on Communications*, 62(10):3665–3677, 2014.

[66] Maria Gregori, Jesús Gómez-Vilardebó, Javier Matamoros, and Deniz Gündüz. Wireless content caching for small cell and d2d networks. *IEEE Journal on Selected Areas in Communications*, 34(5):1222–1234, 2016.

[67] Tianyu Wang, Lingyang Song, and Zhu Han. Dynamic femtocaching for mobile users. In *2015 IEEE wireless communications and networking conference (WCNC)*, pages 861–865. IEEE, 2015.

[68] Xiuhua Li, Xiaofei Wang, and Victor CM Leung. Weighted network traffic offloading in cache-enabled heterogeneous networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

[69] Ying Cui and Dongdong Jiang. Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks. *IEEE transactions on Wireless Communications*, 16(1):250–264, 2016.

[70] Bo Bai, Li Wang, Zhu Han, Wei Chen, and Tommy Svensson. Caching based socially-aware d2d communications in wireless content delivery networks: A hypergraph framework. *IEEE Wireless Communications*, 23(4):74–81, 2016.

[71] Yecheng Wu, Sha Yao, Yang Yang, Ting Zhou, Hua Qian, Honglin Hu, and Matti Hamalainen. Challenges of mobile social device caching. *IEEE Access*, 4:8938–8947, 2016.

[72] Binqiang Chen, Chenyang Yang, and Gang Wang. Cooperative device-to-device communications with caching. In *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2016.

[73] Constantinos Psomas, Gan Zheng, and Ioannis Krikidis. Cooperative wireless edge caching with relay selection. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2017.

[74] Slađana Jošilo, Valentino Pacifici, and György Dán. Distributed algorithms for content placement in hierarchical cache networks. *Computer Networks*, 125:160–171, 2017.

[75] Yong Cui, Jian Song, Minming Li, Qingmei Ren, Yangjun Zhang, and Xuejun Cai. Sdn-based big data caching in isp networks. *IEEE Transactions on Big Data*, 4(3):356–367, 2018.

[76] Shashwat Kumar and A Antony Franklin. Consolidated caching with cache splitting and trans-rating in mobile edge computing networks. In *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE, 2017.

[77] Shashwat Kumar, Doddala Sai Vineeth, et al. Edge assisted dash video caching mechanism for multi-access edge computing. In *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE, 2018.

[78] Tadege Mihretu Ayenew, Dionysis Xenakis, Nikos Passas, and Lazaros Merakos. A novel content placement strategy for heterogeneous cellular networks with small cells. *IEEE Networking Letters*, 2019.

[79] Dewang Ren, Xiaolin Gui, Kaiyuan Zhang, and Jie Wu. Hybrid collaborative caching in mobile edge networks: An analytical approach. *Computer Networks*, 158:1–16, 2019.

[80] Lintao Yang, Yanqiu Chen, Luqi Li, and Hao Jiang. Cooperative caching and delivery algorithm based on content access patterns at network edge. In *International Conference on 5G for Future Wireless Networks*, pages 99–123. Springer, 2019.

[81] Xinwei Liu, Jiaxin Zhang, Xing Zhang, and Wenbo Wang. Mobility-aware coded probabilistic caching scheme for mec-enabled small cell networks. *IEEE Access*, 5:17824–17833, 2017.

[82] Tuo Liu, Sheng Zhou, and Zhisheng Niu Tsinghua. Mobility-aware coded-caching scheme for small cell network. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[83] Min Chen, Yixue Hao, Long Hu, Kaibin Huang, and Vincent KN Lau. Green and mobility-aware caching in 5g networks. *IEEE Transactions on Wireless Communications*, 16(12):8347–8361, 2017.

[84] Yu Ye, Ming Xiao, and Mikael Skoglund. Mobility-aware content preference learning in decentralized caching networks. *IEEE Transactions on Cognitive Communications and Networking*, 6(1):62–73, 2019.

[85] Emre Ozfatura and Deniz Gündüz. Mobility and popularity-aware coded small-cell caching. *IEEE Communications Letters*, 22(2):288–291, 2017.

[86] Valerio Bioglio, Frederic Gabry, and Ingmar Land. Optimizing mds codes for caching at the edge. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[87] Mohammad Ali Maddah-Ali and Urs Niesen. Fundamental limits of caching. *IEEE Transactions on Information Theory*, 60(5):2856–2867, 2014.

[88] Mohammad Ali Maddah-Ali and Urs Niesen. Decentralized coded caching attains order-optimal memory-rate tradeoff. *IEEE/ACM Transactions On Networking*, 23(4):1029–1040, 2014.

[89] Ramtin Pedarsani, Mohammad Ali Maddah-Ali, and Urs Niesen. Online coded caching. *IEEE/ACM Transactions on Networking*, 24(2):836–845, 2015.

[90] Nikhil Karamchandani, Urs Niesen, Mohammad Ali Maddah-Ali, and Suhas N Diggavi. Hierarchical coded caching. *IEEE Transactions on Information Theory*, 62(6):3212–3229, 2016.

[91] Urs Niesen and Mohammad Ali Maddah-Ali. Coded caching with nonuniform demands. *IEEE Transactions on Information Theory*, 63(2):1146–1158, 2016.

[92] Jinbei Zhang, Xiaojun Lin, and Xinbing Wang. Coded caching under arbitrary popularity distributions. *IEEE Transactions on Information Theory*, 64(1):349–366, 2017.

[93] Ejder Baştuğ, Mehdi Bennis, and Mérouane Debbah. Social and spatial proactive caching for mobile data offloading. In *2014 IEEE international conference on communications workshops (ICC)*, pages 581–586. IEEE, 2014.

[94] Kyi Thar, Nguyen H Tran, Thant Zin Oo, and Choong Seon Hong. Deepmec: Mobile edge caching using deep learning. *IEEE Access*, 6:78260–78275, 2018.

[95] Chenxi Zhang, Pinyi Ren, and Qinghe Du. Learning-to-rank based strategy for caching in wireless small cell networks. In *International Conference on Internet of Things as a Service*, pages 111–119. Springer, 2018.

[96] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. *Collaborative filtering recommender systems*. Now Publishers Inc, 2011.

[97] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

[98] Livia Elena Chatzieleftheriou, Merkouris Karaliopoulos, and Iordanis Koutsopoulos. Caching-aware recommendations: Nudging user preferences towards better caching performance. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[99] Gao Shen, Li Pei, Pan Zhiwen, Liu Nan, and You Xiaohu. Machine learning based small cell cache strategy for ultra dense networks. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2017.

[100] Binqiang Chen and Chenyang Yang. Caching policy for cache-enabled d2d communications by learning user preference. *IEEE Transactions on Communications*, 66(12):6586–6601, 2018.

[101] Yi Li, Chen Zhong, M Cenk Gursoy, and Senem Velipasalar. Learning-based delay-aware caching in wireless d2d caching networks. *IEEE Access*, 6:77250–77264, 2018.

[102] Yuyang Wang, Yun Chen, Haibo Dai, Yongming Huang, and Luxi Yang. A learning-based approach for proactive caching in wireless communication networks. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2017.

[103] Lu Hou, Lei Lei, Kan Zheng, and Xianbin Wang. A q-learning-based proactive caching strategy for non-safety related services in vehicular networks. *IEEE Internet of Things Journal*, 6(3):4512–4520, 2018.

[104] Chenyang Wang, Shanjia Wang, Ding Li, Xiaofei Wang, Xiuhua Li, and Victor CM Leung. Q-learning based edge caching optimization for d2d enabled hierarchical wireless networks. In *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 55–63. IEEE, 2018.

[105] Amal Feriani and Ekram Hossain. Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 2021.

[106] Alireza Sadeghi, Fatemeh Sheikholeslami, and Georgios B Giannakis. Optimal and scalable caching for 5g using reinforcement learning of space-time popularities. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):180–190, 2017.

[107] Samuel O Somuyiwa, András György, and Deniz Gündüz. A reinforcement-learning approach to proactive caching in wireless networks. *IEEE Journal on Selected Areas in Communications*, 36(6):1331–1344, 2018.

[108] Ying He, Zheng Zhang, F Richard Yu, Nan Zhao, Hongxi Yin, Victor CM Leung, and Yanhua Zhang. Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks. *IEEE Transactions on Vehicular Technology*, 66(11):10433–10445, 2017.

[109] Hao Zhu, Yang Cao, Xiao Wei, Wei Wang, Tao Jiang, and Shi Jin. Caching transient data for internet of things: A deep reinforcement learning approach. *IEEE Internet of Things Journal*, 6(2):2074–2083, 2018.

[110] Wei Jiang, Gang Feng, Shuang Qin, Tak Shing Peter Yum, and Guohong Cao. Multi-agent reinforcement learning for efficient content caching in mobile d2d networks. *IEEE Transactions on Wireless Communications*, 18(3):1610–1622, 2019.

[111] Tong Wu, Pan Zhou, Kai Liu, Yali Yuan, Xiumin Wang, Huawei Huang, and Dapeng Oliver Wu. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(8):8243–8256, 2020.

[112] Jiongjiong Song, Min Sheng, Tony QS Quek, Chao Xu, and Xijun Wang. Learning-based content caching and sharing for wireless networks. *IEEE Transactions on Communications*, 65(10):4309–4324, 2017.

[113] Di Wu, Yuan Zhang, Juan Luo, and Renfa Li. Efficient data dissemination by crowd-sensing in vehicular networks. In *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*, pages 314–319. IEEE, 2014.

[114] David Applegate, Aaron Archer, Vijay Gopalakrishnan, Seungjoon Lee, and KK Ramakrishnan. Optimal content placement for a large-scale vod system. *IEEE/ACM Transactions on Networking*, 24(4):2114–2127, 2016.

[115] Bin Cao, Long Zhang, Yun Li, Daquan Feng, and Wei Cao. Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework. *IEEE Communications Magazine*, 57(3):56–62, 2019.

[116] Bartlomiej Blaszczyszyn and Anastasios Giovanidis. Optimal geographic caching in cellular networks. In *2015 IEEE International Conference on Communications (ICC)*, pages 3358–3363. IEEE, 2015.

[117] Jian Qiao, Yejun He, and Xuemin Sherman Shen. Proactive caching for mobile video streaming in millimeter wave 5g networks. *IEEE Transactions on Wireless Communications*, 15(10):7187–7198, 2016.

[118] J. Tadrous and A. Eryilmaz. On optimal proactive caching for mobile networks with demand uncertainties. *IEEE/ACM Transactions on Networking*, 24(5):2715–2727, October 2016.

[119] Fei Shen, Kenza Hamidouche, Ejder Bastug, and Mérouane Debbah. A stackelberg game for incentive proactive caching mechanisms in wireless networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[120] Zhen Tong, Yuedong Xu, Tao Yang, and Bo Hu. Quality-driven proactive caching of scalable videos over small cell networks. In *2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pages 90–96. IEEE, 2016.

[121] Xiaomin Li and Jiafu Wan. Proactive caching for edge computing-enabled industrial mobile wireless networks. *Future Generation Computer Systems*, 89:89–97, 2018.

[122] Sara A Elsayed, Sherin Abdelhamid, and Hossam S Hassanein. Proactive caching at parked vehicles for social networking. In *2018 IEEE International conference on communications (ICC)*, pages 1–6. IEEE, 2018.

[123] Shashwat Kumar, Sai Vineeth Doddala, A Antony Franklin, and Jiong Jin. Ran-aware adaptive video caching in multi-access edge computing networks. *Journal of Network and Computer Applications*, 168:102737, 2020.

[124] Liying Li, Guodong Zhao, and Rick S Blum. A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies. *IEEE Communications Surveys & Tutorials*, 20(3):1710–1732, 2018.

[125] Tuyen X Tran, Mohammad-Parsa Hosseini, and Dario Pompili. Mobile edge computing: Recent efforts and five key research directions. *IEEE COMSOC MMTC Commun.-Frontiers*, 12(4):29–33, 2017.

[126] Yaping Sun, Zhiyong Chen, and Hui Liu. Delay analysis and optimization in cache-enabled multi-cell cooperative networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2016.

[127] Jingjing Yao and Nirwan Ansari. Joint content placement and storage allocation in c-rans for iot sensing service. *IEEE Internet of Things Journal*, 6(1):1060–1067, 2018.

[128] BN Bharath, Kyatsandra G Nagananda, and H Vincent Poor. A learning-based approach to caching in heterogenous small cell networks. *IEEE Transactions on Communications*, 64(4):1674–1686, 2016.

[129] Sabrina Müller, Onur Atan, Mihaela van der Schaar, and Anja Klein. Context-aware proactive content caching with service differentiation in wireless networks. *IEEE Transactions on Wireless Communications*, 16(2):1024–1036, 2016.

[130] BN Bharath, Kyatsandra G Nagananda, Deniz Gündüz, and H Vincent Poor. Caching with time-varying popularity profiles: A learning-theoretic perspective. *IEEE Transactions on Communications*, 66(9):3837–3847, 2018.

[131] Navneet Garg, Mathini Sellathurai, Vimal Bhatia, BN Bharath, and Tharmalingam Ratnarajah. Online content popularity prediction and learning in wireless edge caching. *IEEE Transactions on Communications*, 68(2):1087–1100, 2019.

[132] Peiyan Yuan, Yunyun Cai, Yihang Liu, Junna Zhang, Yali Wang, and Xiaoyan Zhao. Prorec: a unified content caching and replacement framework for mobile edge computing. *Wireless Networks*, pages 1–13, 2020.

[133] Ronghui Hou, Kaiwen Huang, Huilin Xie, King-Shan Lui, and Hongyan Li. Caching and resource allocation in small cell networks. *Computer Networks*, page 107100, 2020.

[134] Ni Zhang, Songtao Guo, Yifan Dong, and Defang Liu. Joint task offloading and data caching in mobile edge computing networks. *Computer Networks*, page 107446, 2020.

[135] Linpeng Tang, Qi Huang, Amit Puntambekar, Ymir Vigfusson, Wyatt Lloyd, and Kai Li. Popularity prediction of facebook videos for higher quality streaming. In *2017 {USENIX} Annual Technical Conference ({USENIX}{ATC} 17)*, pages 111–123, 2017.

[136] Jun Li, Shuang Hong, Sha Xia, and Shengmei Luo. Neural network based popularity prediction for iptv system. *J. Networks*, 7(12):2051–2056, 2012.

[137] Suoheng Li, Jie Xu, Mihaela Van Der Schaar, and Weiping Li. Popularity-driven content caching. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[138] Emira Ben Abdelkrim, Mohammad A Salahuddin, Halima Elbiaze, and Roch Glitho. A hybrid regression model for video popularity-based cache replacement in content delivery networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2016.

[139] SM Shahrear Tanzil, William Hoiles, and Vikram Krishnamurthy. Adaptive scheme for caching youtube content in a cellular network: Machine learning approach. *Ieee Access*, 5:5870–5881, 2017.

[140] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[141] Tingting Hou, Gang Feng, Shuang Qin, and Wei Jiang. Proactive content caching by exploiting transfer learning for mobile edge computing. *International Journal of Communication Systems*, 31(11):e3706, 2018.

[142] Shan Zhang, Peter He, Katsuya Suto, Peng Yang, Lian Zhao, and Xuemin Shen. Cooperative edge caching in user-centric clustered mobile networks. *IEEE Transactions on Mobile Computing*, 17(8):1791–1805, 2017.

[143] Noor Abani, Torsten Braun, and Mario Gerla. Proactive caching with mobility prediction under uncertainty in information-centric networks. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, pages 88–97, 2017.

[144] Lin Yao, Ailun Chen, Jing Deng, Jianbang Wang, and Guowei Wu. A cooperative caching scheme based on mobility prediction in vehicular content centric networks. *IEEE Transactions on Vehicular Technology*, 67(6):5435–5444, 2017.

[145] Hakima Khelifi, Senlin Luo, Boubakr Nour, Akrem Sellami, Hassine Moungla, and Farid Naït-Abdesselam. An optimized proactive caching scheme based on mobility prediction for vehicular networks. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[146] Ejder Baştuğ, Jean-Louis Guénégo, and Mérouane Debbah. Proactive small cell networks. In *ICT 2013*, pages 1–5. IEEE, 2013.

[147] Juan Liu, Bo Bai, Jun Zhang, and Khaled B Letaief. Cache placement in fog-rans: From centralized to distributed algorithms. *IEEE Transactions on Wireless Communications*, 16(11):7039–7051, 2017.

[148] Dong Liu and Chenyang Yang. Caching at base stations with heterogeneous user demands and spatial locality. *IEEE Transactions on Communications*, 67(2):1554–1569, 2018.

[149] Naifu Zhang, Kaibin Zheng, and Meixia Tao. Using grouped linear prediction and accelerated reinforcement learning for online content caching. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2018.

[150] Yanxiang Jiang, Miaoli Ma, Mehdi Bennis, Fu-Chun Zheng, and Xiaohu You. User preference learning-based edge caching for fog radio access network. *IEEE Transactions on Communications*, 67(2):1268–1283, 2018.

[151] Sami Kekki, Walter Featherstone, Yonggang Fang, Pekka Kuure, Alice Li, Anurag Ranjan, Debashish Purkayastha, Feng Jiangping, Danny Frydman, Gianluca Verin, et al. Mec in 5g networks. *ETSI white paper*, 28:1–28, 2018.

[152] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

[153] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[154] Ion Iancu. A mamdani type fuzzy logic controller. In *Fuzzy Logic-Controls, Concepts, Theories and Applications*. InTech, 2012.

[155] Jie Cui, Lu Wei, Hong Zhong, Jing Zhang, Yan Xu, and Lu Liu. Edge computing in vanets-an efficient and privacy-preserving cooperative downloading scheme. *IEEE Journal on Selected Areas in Communications*, 38(6):1191–1204, 2020.

[156] Nazmul Siddique and Hojjat Adeli. *Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing*. John Wiley & Sons, 2013.

[157] Jiaying Yin, Lixin Li, Huisheng Zhang, Xu Li, Ang Gao, and Zhu Han. A prediction-based coordination caching scheme for content centric networking. In *2018 27th Wireless and Optical Communication Conference (WOCC)*, pages 1–5. IEEE, 2018.

[158] Yang Du, Pengyu Gao, Xiaodong Wang, Binhong Dong, Zhi Chen, and Shaoqian Li. Monte-carlo tree search aided contextual online learning approach for wireless caching. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7. IEEE, 2018.

[159] Peng Yang, Ning Zhang, Shan Zhang, Li Yu, Junshan Zhang, and Xuemin Sherman Shen. Content popularity prediction towards location-aware mobile edge caching. *IEEE Transactions on Multimedia*, 21(4):915–929, 2018.

[160] Long Teng, Xiang Yu, Jianhua Tang, and Mingxia Liao. Proactive caching strategy with content-aware weighted feature matrix learning in small cell network. *IEEE Communications Letters*, 23(4):700–703, 2019.

[161] Teofilo F Gonzalez. *Handbook of Approximation Algorithms and Metaheuristics: Methologies and Traditional Applications, Volume 1*. CRC Press, 2018.

[162] Xiuhua Li, Xiaofei Wang, Keqiu Li, Zhu Han, and Victor CM Leung. Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design. *IEEE Transactions on Wireless Communications*, 16(10):6926–6939, 2017.

[163] Yongxue Tian and Li Pan. Predicting short-term traffic flow by long short-term memory recurrent neural network. In *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, pages 153–158. IEEE, 2015.

[164] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[165] Tzay Y Young and Thomas W Calvert. *Classification, estimation, and pattern recognition*. Elsevier Publishing Company, 1974.

[166] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[167] Manoj Kumar Somesula, Rashmi Ranjan Rout, and DVLN Somayajulu. Deadline-aware caching using echo state network integrated fuzzy logic for mobile edge networks. *Wireless Networks*, pages 1–21, 2021.

[168] Jong-Kwon Lee and Jennifer C Hou. Modeling steady-state and transient behaviors of user mobility: formulation, analysis, and application. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 85–96, 2006.

[169] Xuejun Zhuo, Qinghua Li, Wei Gao, Guohong Cao, and Yiqi Dai. Contact duration aware data replication in delay tolerant networks. In *2011 19th IEEE International Conference on Network Protocols*, pages 236–245. IEEE, 2011.

[170] Derek Leong, Alexandros G Dimakis, and Tracey Ho. Distributed storage allocations. *IEEE Transactions on Information Theory*, 58(7):4733–4752, 2012.

[171] Peng Lin, Qingyang Song, and Abbas Jamalipour. Multidimensional cooperative caching in comp-integrated ultra-dense cellular networks. *IEEE Transactions on Wireless Communications*, 19(3):1977–1989, 2019.

[172] Kumara Sastry and Graham Goldberg, David E.and Kendall. *Genetic Algorithms*. Springer, Boston, MA, 2014.

[173] Konstantinos Poularakis and Leandros Tassiulas. Publicly available code, 2016.

[174] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[175] Rose E Wang, Michael Everett, and Jonathan P How. R-maddpg for partially observable environments and limited communication. *arXiv preprint arXiv:2002.06684*, 2020.

[176] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[177] Mohamed Ahmed, Stefano Traverso, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. Analyzing the performance of lru caches under non-stationary traffic patterns. *arXiv preprint arXiv:1301.4909*, 2013.

[178] Dario Rossi and Giuseppe Rossini. Caching performance of content centric networks under multi-path routing (and more). *Relatório técnico, Telecom ParisTech*, pages 1–6, 2011.

[179] Aamer Jaleel, Kevin B Theobald, Simon C Steely Jr, and Joel Emer. High performance cache replacement using re-reference interval prediction (rrip). *ACM SIGARCH Computer Architecture News*, 38(3):60–71, 2010.

List of Publications

1. **Manoj Kumar Somesula** and Rashmi Ranjan Rout and D. V. L. N. Somayajulu. "Deadline-aware caching using echo state network integrated fuzzy logic for mobile edge networks." *Wireless Networks, Springer* (2021): 1-21. https://doi.org/10.1007/s11276-021-02578-2

2. **Manoj Kumar Somesula** and Rashmi Ranjan Rout and D. V. L. N. Somayajulu. "Contact Duration-Aware Cooperative Cache Placement using Genetic Algorithm for Mobile Edge Networks." *Computer Networks, Elsevier* (2021): 108062.

3. **Manoj Kumar Somesula** and Rashmi Ranjan Rout and D. V. L. N. Somayajulu. "Cooperative Cache Update using Multi-Agent Recurrent Deep Reinforcement Learning for Mobile Edge Networks." *Computer Networks, Elsevier*. (Under Review)

4. **Manoj Kumar Somesula** and Rashmi Ranjan Rout and D. V. L. N. Somayajulu. "User Preference Learning based Cooperative Cache Placement for Mobile Edge Networks with Adaptive User Clustering." *Future Generation Computer Systems, Elsevier*. (Submitted)