UAV Assisted Cooperative Caching on Network Edge using Multi Agent Actor Critic Reinforcement Learning

by

Sadman Araf
17101354
Adittya Soukarjya Saha
17101148
Salman Ibne Eunus
17101051

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
December 2020

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]

Sadman Araf
17101354

[illegible]

Adittya Soukarjya Saha
17101148

[illegible]

Salman Ibne Eunus
17101051

Approval

The thesis/project titled "UAV Assisted Cooperative Caching on Network Edge using Multi Agent Actor Critic Reinforcement Learning" submitted by

1. Sadman Araf (17101354)
2. Adittya Soukarjya Saha (17101148)
3. Salman Ibne Eunus (17101051)

Of Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on December, 2020.

Examining Committee:

Supervisor:
(Member)

[illegible]

Sadia Hamid Kazi
Deputy Head and Assistant Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

[illegible]

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
Brac University

Dr. Mahbub Majumdar
Professor
Department of Computer Science and Engineering
Brac University

Abstract

In recent times, Multi-access edge computing (MEC) has been introduced to assist cloud servers by bringing the computation closer to the edge. This is a well-known replacement to deal with the strict latency faced by users while retrieving contents from long-distance data centers. To cope up with this latency while simultaneously improving users' QOS poses a limitation which can be handled through caching at edge nodes. However, where to cache and what to cache so that a higher cache hit rate is achieved also poses another significant issue which is addressed in this research. In this paper we have approached the problem of dynamic caching along with the selection of edge node that leads to better cache hit rate. We have also proposed the use of UAVs as aerial Base Station(BS) to assist in peak hours where a ground base station is not enough to support the surge in user requests.It also elaborates the optimal relocation of UAVs to effectively support user mobility, which then caters a cluster of users by the K-means clustering algorithm. In addition, to maximize the cache hit ratio we have proposed a cooperative deep reinforcement learning algorithm which ensured a global increase in cache hit ratio and also an efficient allocation of storage. We have shown simulations on UAV reallocation based on user mobility patterns and also achieved higher global cache hit ratio using our proposed multi-agent actor-critic algorithm. In this paper, emphasis was given on how to cache and where to cache based on the cooperation of UAV and GBS which open doors for further research.

Keywords: Unmanned aerial vehicle(UAV), Cooperative Edge Caching, signal-to-noise ratio, multi-agent deep deterministic policy gradient, K-means clustering

Dedication

This thesis is dedicated to our parents for their limitless support and encouragement.

Acknowledgement

Firstly, all praise to the Almighty God for whom our thesis have been completed without any major interruption even throughout this time of the pandemic.

Secondly, to our supervisor, Sadia Hamid Kazi and co-supervisor, Dr. Md. Golam Rabiul Alam, for their continuous support, immense knowledge, motivation and enthusiasm in our work. Without their guidance at each step, this paper would never have been accomplished.

Lastly, to our parents who have been the main pillar of strength for us. With their never ending support and prayers, we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ϵ	Epsilon
υ	Upsilon
$A2A$	Air-to-air
$A2G$	Air-to-ground
BS	Base Station
DRL	Deep Reinforcement Learning
$FANET$	Flying Ad-hoc Network
GBS	Ground Base Station
LFU	Least Frequently Used
LRU	Least Recently Used
$MAAC$	Multi-Agent Actor-Critic
RAN	Radio Access Network
UAV	Unmanned Aerial Vehicle
UE	User Equipment
$VANET$	Vehicular Ad-hoc Network

Chapter 1

Introduction

1.1 Motivation

The launching of Internet of Things(IoT) and the upcoming 5G technologies has been the prime reason for much higher quantity of data produced by devices presently compared to the last decade and is expected to rise rapidly in the next couple of years [1]. As per research conducted by Cisco Visual Networking Index, by the year 2021, there is likely to be 4.6B global internet users, 27.1B network devices and connections and also 82 percent of all traffic is predicted to be video files and will be rising exponentially over time. At present, cloud tackles most of the data [1] and 92 percent of all data center traffic will come from the cloud according to the global cloud index. Due to such plethora of interconnected devices, load of internet traffic will be exceedingly higher to manage and will eventually lead to network latency and delay. In addition, as the number of connected devices are increasing, most new technologies created require real-time processing capability. For instance- real-time processing and quick decisions are important for self-driving cars as they have to make the decision of braking within fraction of a second. UAVs and quad copters, they need to be smart enough to avoid an obstacle in real time. Transferring all these data to and forth from the cloud is time consuming and cannot provide solutions in real time.
To address the challenge of dealing with huge data traffic, a promising approach has emerged which is known as Multi-access Edge Computing or Mobile Edge computing. It is a distributed framework which brings the computation of data closer to the source device. Different architectural approaches to edge computing exist that have been defined by different groups [2]. Cisco and Intel have promoted OpenFog, ETSI has promoted Mobile Edge Computing. Frontier Computing, Mist Computing, Dew Computing, etc have also been promoted. Because of the computational power being brought closer to the source, many AI applications are now a possibility. Edge computing can exist in the form of device edge or cloud edge. In the device edge, edge computing is provided to the users that are in the existing environment, e.g. Microsoft Azure IOT Edge. Moreover, public cloud is further extended to form cloud edge, where the static contents are cached and are distributed among local edge servers, e.g., Vapor IO.

Edge computing provides the following solutions [3]:

1. Responsiveness: As the processing is done closer to the source device, the response time would be quicker.

2. Security: As the data hops many times before reaching its destination, there is always a chance of security risk. Since every raw data is not transmitted to the cloud server, keeping the contents locally, reduces the risk.

3. Edge AI: Self driving car or flying drones needs to make decisions in real time, so image processing can be done in the local edge rather than sending the data to the cloud for analysis.

1.2 Problem Definition

It is often seen that at a specified point in time(rush hour), when the passengers are taking the same route to a destination, they are faced with the occasion of clogged routes. This could be due to traffic signals staying red for quite sometime, or the overwhelming number of vehicles on road towards the destinations e.g. offices, etc. During those times, the passengers on board often seek the opportunity to indulge themselves in popular contents. So, the number of requested contents at that time is significantly high. During those times, it is not possible for GBS alone to handle this huge amount of data requests. The introduction of UAVs without proper positioning would not resolve the matter. In order to tackle all these problem, we are proposing the use of UAVs along with cooperative caching with the GBSs by the use of multi-agent actor critic reinforcement learning (MAAC) approach. We are also incorporating K-means clustering algorithm to get the clusters of users to serve via the UAVs. In order to properly position the UAVs, the users' location are taken into account through the Random Waypoint Model. Cooperation of one GBS with another GBS may not be feasible as they might be far apart. So we need a dynamic BS which can assist to increase the global cache rate during such scenarios. Previously when some of the passengers were not getting their desired content, it meant that the cache-hit ratio was low. With the use of cooperative caching mechanism, between a particular UAV and MEC, enables a greater chance of popular contents to be served to the users, i.e. a greater cache-hit ratio by the MAAC as proposed.

1.3 Research Objectives

In this paper we implemented a Deep Reinforcement learning based algorithm to cache contents dynamically. By intelligently analyzing context information and request frequency patterns, a context-aware caching is designed. In this proposed scheme, the base station nodes will be aware of its environment and will make decisions to cache the right content to maximize caching performance.

Next, we incorporated UAVs, which will act as flying base stations. We created a hierarchy of base stations where MEC servers will be acting as a ground base station and UAVs will be analyzing the mobility patterns of wireless users to efficiently serve them.

Thirdly, we optimized the UAVs trajectory by implementing a clustering based algorithm so that the UAVs dynamically cluster based on user mobility.

Furthermore, we integrated the air-ground caching by communication, caching, computation and control of mobility:

1. Communication model: In the sky, the capacity of the channel from the UAV to the cloud is limited and the UAVs transmission to the users is over mmWave and that the distance to the user is closer than the distance to the data center. In the ground, the MEC attached to the RSU communicates with the data center using a fiber backhaul link. We assume each RSU has a limited number of channels which can be used by a user one at a time.
2. Caching model: A hierarchy of caching of the contents will be established in the ground and as well as in the air base station. We are proposing a multi-agent actor-critic based cooperative reinforcement learning approach.
3. Computation model:
    - Computation at GBS: When a user requests a content, it is offloaded to the nearest GBS, and it will execute it if the server has enough computational resources. If not, it will be offloaded to the nearest UAV.
    - Computation at UAV: A UAV will compute the requested content only if it's computational constraints are met and if it has enough battery.

4. Mobility model: Since higher accuracy is to be aimed while effectively predicting the next states of the users, we are proposing the Random Waypoint Model as our most sought after mobility model. Random Waypoint aims to model mobile users into uniformly distributed waypoints or nodes. These nodes wait a random pause time at each waypoint before starting to move towards the next waypoint. As a result this perfectly models the user movements as shown in Figure 1.1.

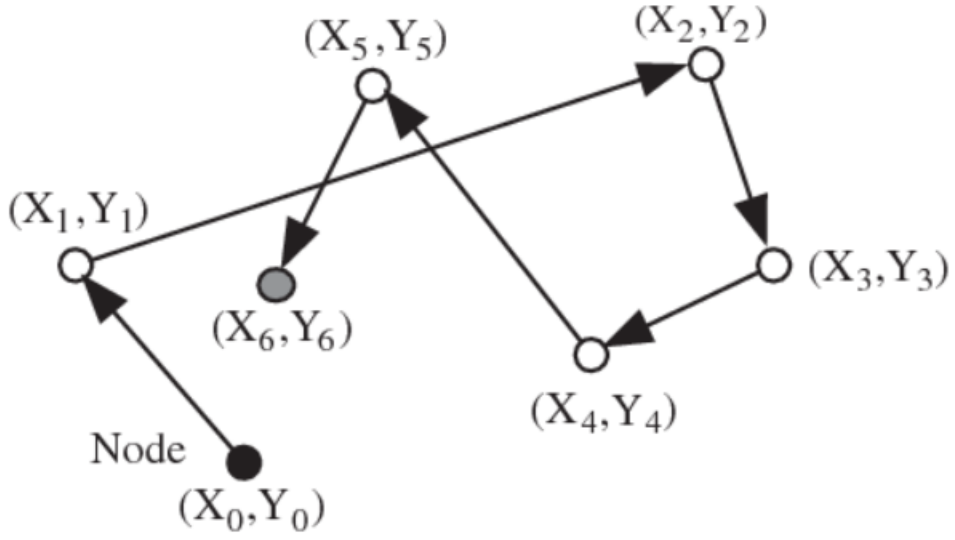


Figure 1.1: Movement of nodes in the Random Waypoint Model [4]

Our main contributions of this work are as follows:

- Communication model is built up using a concrete channel model which considers the downlink transmission only as download data content is relatively far greater compared to the amount of service request. The path loss of line-of-sight (LoS) and non-line-of-sight (NLoS) links is modeled according to the log-normal shadowing model by choosing specific channel parameters which ultimately lead to the calculation of the signal to noise ratio(SNR) that serves as a crucial parameter in the communication model as certain decisions are made on the basis of this threshold.

- A multi-agent actor-critic based cooperative reinforcement learning approach is being used for the caching model of the system. Computation model is designed in such a way that it takes into account the computational resources as well as storage constraints of the UAVs, MECs while serving the cached content requested by the users.

- Control of mobility is handled by placing UAVs in public congregation areas by using Random Waypoint Model which is the mobility model used for accurately predicting the users' location.

- To put our proposed framework to test, we compared the learning-based performance with caching policies, mainly least frequently used (LFU) and least recently used (LRU).

1.4 Thesis Orientation

This report emphasized to integrate the air and ground networks to create a hierarchy of caching to reduce the content transmission latency and to improve the quality of experience for the users. The goal of the authors is to implement a Deep Reinforcement learning based algorithm to cache contents based on context awareness. The overall report focuses on the steps followed by the researchers.

Firstly, Introduction part, Chapter 1, states the motivation behind research which inspired authors to address this particular problem statement. The goals of our research and summary of the work is briefly discussed here.

In Background Study section, Chapter 2, we have compared the performance of Cloud, Fog and Edge computing using various performance metrics to show that Multi Access Edge Computing leads to the minimization of latency, maximization of network capacity and minimization of energy consumption as it caches data close to the user. The purpose of background study was finding out the limitations of previous works and researches.

In Literature Review section, Chapter 3, we have discussed about papers from computer science background which have addressed similar issue. In addition to that, some statistical papers are mentioned which refers to the available secondary data.

In Proposed System Architecture section, Chapter 4, we talked about our System model and Problem formulation. In problem formulation, we further elaborated several models which we have proposed namely - computation model, mobility model, communication model and caching model.

In Cache Node selection and Cooperative Caching Framework section, Chapter 5, we have explained our algorithm for cache node selection using K-means clustering and also caching of contents using Multi-Agent Actor-critic Reinforcement Learning.

In Simulation Results section, Chapter 6, we have shown our results of simulation using various data sets and explained our results of simulation to prove our hypothesis.

In Conclusion section, Chapter 7, we finally conclude our report by clearly explaining our unique research contributions and also talked briefly about future resource scopes in this field.

Chapter 2

Background Study

2.1 Cloud vs Fog vs Edge Computing

The performance analysis between cloud, fog and edge computing must be done before digging much deeper into the details of Edge computing to prove it's capability and usability. The Internet of Things (IoT) is anticipated to utilise multi access edge computing to reduce delay and then make use of computing power through offloading IoT, which gather large amount of data but has less storage capacity with individual devices and cloud computing is used in general to examine the data. However, cloud computing gives rise to higher latency than edge computing. To decrease the bandwidth consumption and data transfer latency, edge computing is a more adequate substitute of cloud computing.
The three layers shown in the Fig 2.1 can be connected with each other by making use of gateways. We compare the delay in these three layers with the aid of numerical analysis obtained from simulation. The simulation configuration is similar to ones given in [**4**].

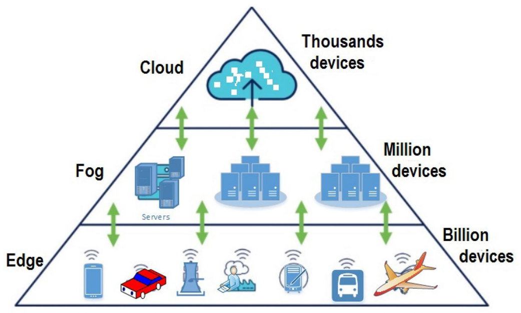


Figure 2.1: Block diagram for computing edge, fog and cloud computing in terms of number of devices and locations in the hierarchy [5].

We have considered computation delay as shown in Fig 2.2 and communication delay as shown in Fig 2.3. When cloud computing is used there will be highest delay compared to fog and edge computing , as shown in Fig 2.3. There will be lowest delay compared to cloud computing and fog computing when multi access edge computing is used as shown in Fig 2.3.

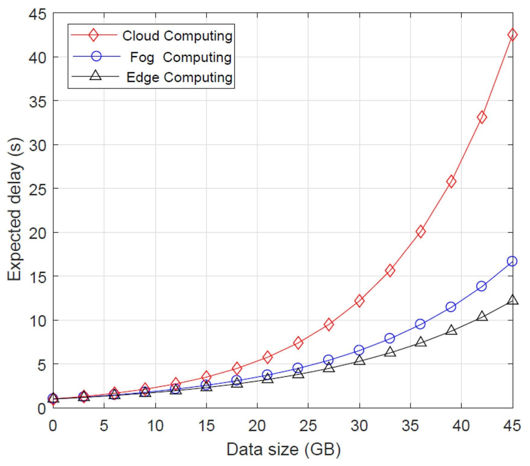


Figure 2.2: Comparison of expected computation delays when data is offloaded to edge, fog and cloud for processing and receiving the response back to the end device [5].

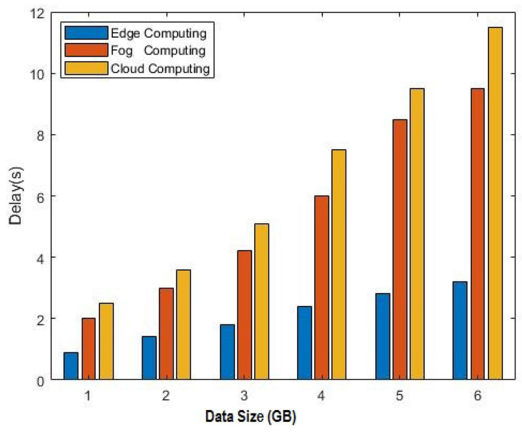


Figure 2.3: Comparison in terms of communication delay when end users communicate with edge , fog and cloud computing units [5].

2.2 Caching in Edge Computing

Quality-of-experience(QoE) is a key aspect to be focused on as the mobile data traffic will continuously increase as the users will get a grip on upcoming technologies in the near future.[4] In order to localize flow of data, the content caching strategy is designed in such a way such that the cached content is transmitted from the centralized location to end users in cache-enabled mobile networks.
Mobile edge caching thereby provides the following solution:

1. Reducing redundancy in the data traffic through caching of popular contents at mobile edge networks which can then be served to the customers locally.

2. Improvement in the QoE of users.

However, to make mobile edge caching significant, some key features are needed to be taken into consideration:

1. Positioning of Cache/ Cache placing: Caching popular contents at BS(Base stations) drastically cuts down redundancy in data traffic since duplication of the same popular content is being eliminated. This is due to the fact that the requests of users for the same content does not need to be fetched twice. Having 5th Generation networks, device to device(D2D) communication ensures exploitation of low-cost storage units at UEs(User Equipments), so constant delivery of the data can be made certain.

2. Popularity of Content: Currently mobile edge caching presumes that a Zipf distribution is maintained in terms of content popularity, but this is subject to change due to variations in predilection of the users. So, having a fore grip on content popularity is not possible, since user groups are connected with individual edge nodes.

3. Algorithms and Caching policies: QoE improvement, offloading of traffic, reduction in energy consumption; all these lead up to cache hit ratio either directly or indirectly. Caching policies successful in wired caching like LRU(least recently used) and LFU(least frequently used) could be less successful in terms of mobile edge caching since these do not consider the operating environment's characteristics that include user mobility, uncertainty of the topology of the mobile network, fading and interfered wireless channels, etc. Context aware caching policy takes into account the contexts that users might face, for instance the diversity of devices, location, individual characteristics, etc.

But since it is quite sophisticated to take all the parameters, modelling the complicated environment is hard, and as a result the mobile edge caching is difficult to achieve due to deteriorating performance levels in caching. On the other hand intelligent caching policy works by careful consideration of the ever changing natural dynamics of the wireless systems and taking those new states and actions to make the edge nodes learn about the complex operating environment. The feedback generated is incorporated into the policy in order to achieve intelligence in mobile edge computing.
Apart from mobile edge caching, other forms of caching such as proactive caching have also been used for MEC. Proactive Caching as an approach can be split into two stages [5]:

- The offline phase comes first , where a workflow-specified heuristic is automatically determined for pre-computation causing decisions that are cached to be accessed when a process is ongoing.

- The second stage is the online phase where the heuristics that was previously determined are used for the management of cache.

Another type of caching is available as well, which is cooperative caching. Basically in a network when more cache is coordinated and the sharing of resources takes place so that other applications can be served, this mechanism portrays cooperative caching. So, for instance if a particular node's local cache does not contain the required data item, then the data requests can be sent to a nearby cooperating cache, which delivers the data that has been requested, faster than the original data server [6].

To increase the availability of data, data caching and replication is done which proves to be a prerequisite in the environment of wireless network. Due to the fact that the system is unsteady and at the same time dynamic as well, at any point in time, nodes have the scope of entering and leaving the system [6]. Caching can boost the performance of request processing by aiding in the shortening of the routing path of a data request, since that particular data may be near to the origin node of the request.

2.3 Multi Access Edge Computing

Multi-access Edge computing(MEC) cuts down the usage of back-haul or internet bandwidth remarkably as it is able to process huge quantity of data which are close to the source. It also provides the assistance to reduce costs and confirms that the applications can function without accessing expensive and high delay back-haul links [7]. Moreover, since, different types of Base Stations(BSs) were utilized, it will led to diverse edge networks in future. In edge networks, caching will be established at several locations in various Base Stations(BSs). When the requested content is not found at the edge node during the starting point of the process, it has to be fetched from the central server and the edge server will then keep a copy of the content so that it can be used if requested later. This will reduce the delay notably while sending the contents to end users as the data is no longer needed to be retrieved from the central server every time and therefore will not give rise to a slow back-haul link. [8] Also, due to the improvement and installation of less expensive storage units and different BSs, deploying caching mechanisms at Small Base Stations(SBSs) has now became simple and cost effective. The content selection process has to take the decision about which contents must be cached, which contents should be upgraded and the amount of time for which the contents will be cached. To ensure effective usage within MEC-enabled network, network resources like - cache storage size, computing, energy and communication bandwidth should not exceed the maximum limit. Furthermore, to give rise to the productivity of MEC, data which is to be cached, placing the caching data and removing contents from the cache storage by taking into consideration the quality of data, diversity and the mobility of the end users should be done in an optimized manner. Caching optimization handles any limitations linked to optimization on network as well as end user performances, for example - network architecture, analytical point of view and content caching strategies.

2.4 Computing in Mobile Edge Networks

Edge computing is also a major part of the future 5G technologies. By implementing edge computing, different content providers can give solutions that allow content, services and applications to be delivered quicker than if they were delivered from the cloud. Mobile subscribers will also have a smooth experience even if content providers cannot come up with a solution for edge computing as 3rd-party partners will utilize the open radio network edge to provide infrastructure-as-a-service (IaaS), specially the services that depend heavily on mobility e.g. fleet management, transportation and logistics. The implementation of edge computing will benefit the 5G technologies as follows [9]:

2.4.1 Minimization of Latency

The Round Trip Time(RTT) for a wireless network system that is running on 5G is typically 1ms, which when compared to that of a 4G wireless network system is 10 times less. When a MEC server serves it's end users, it takes a shorter period of time than the content that is being delivered from a central server in a centralized data center. In accordance with that, a higher delay is generated for the tasks that are offloaded to the cloud for applications that require data offloading, and these extensive delays are not accepted in many applications. So, in order to reduce the latency , implementation of high-density small base stations(SBSs) with edge computing is even more feasible thing to do.

2.4.2 Maximization of Network Capacity

It is anticipated that the wireless network of 5G is going to enable huge volumes of mobile data per region than the existing network of 4G [10]. In order to tackle this surreal amount of expected data , wireless networks of future require large capacity in the radio access networks(RANs) as well as in front-haul and back-haul. Technologies that involve offloading of data and context-aware computation offloading are looked forward, to deal with the difficulties in the RAN on top of utilizing more spectrum with higher spectrum efficiency. The introduction of MECs along with the caching of contents could aid the network capacity, through caching of popular contents at the BSs and the edge, and also by the reduction of back-haul bandwidth.

2.4.3 Minimization of Energy Consumption

A number of optimization strategies have been proposed in networks and separate devices to decrease the energy consumed. To obtain computational offloading in next generation networks which are heterogeneous in nature, the energy cost related to computing of task and transmission of file is assessed as one of the main element of cost [11]. It is really significant to design an energy efficient data or computation offloading scheme, which cooperatively optimizes radio resources and consumes energy while decreasing the overall delay in transmission. The end devices are characterized into three different types with respect to their capability and demand. Wireless channels of MBSs (Micro Base Stations) and SBSs are designated to mobile devices according to their priority until all devices gains the channels required. At each iteration, the scheme validates that lower energy is consumed by the system, with a huge quantity of end users in particular.

Chapter 3

Related Work

Already few researches have been done in the domain of edge computing, especially bringing in reinforcement learning to make the decisions of caching, which have been very efficient in dealing with the significant issues of the edge caching systems. Alongside this cooperative edge caching has opened up doors for research in the literature of edge caching, in order to improve the performance. This segment contains the discussion about how authors in related works have approached to solve some of the issue.

In [12], the researchers has came up with caching based on deep learning for self-driving cars, which were positioned on Multi-access Edge Computing hardware through approaches that involved Deep Learning. A Convolutional Neural Network (CNN) approach is used to forecast genders as well as age of people via facial recognition which satisfies the demands of the people in self-driving cars. In order to forecast the probability of contents which are asked when the self-driving car is in a certain location, this paper proposed a MultiLayer Perceptron (MLP) framework that is applied at DC(Data Center). Then, the outputs generated from the MLP prediction are placed at edge servers, basically at Road-side Units(RSUs) which are propinquity to the self-driving cars. In addition, AutoRegressive (AR) and the AutoRegressive Moving Average (ARMA) prediction algorithms are not taken into consideration here. Here, MLP has a great role since it holds the capacity to manage both linear and non-linearly prediction issues. The self-driving car then downloads MLP outputs from the MEC architecture because of the contents that are needed to be cached, which is then matched with the final output from the CNN. K-means and binary classification clustering algorithms are chosen since these are efficient in terms of computation shown by the comparison using deep learning and exploiting 4C parts in MEC server reduced content-downloading latency for the development of the caching in the self-driving car. To solve the formulated problem, they implemented Block Successive Majorization-Minimization (BS-MM) algorithm.

In [13], the authors makes use of deep learning algorithms to train and predict the future popularity of contents which allows better caching decisions. They have put forward a caching scheme which can be broken down into three parts, first they predicted the future target class of each content, then they predicted the popularity score in future of each content, and in the end cached the contents which had high popularity scores. They have also worked out an optimization problem which decreases the content access delay in the near future. To solve the problem they proposed a deep layered neural network based prediction system which works together with the information centric MEC architecture. Finally, they compared different models in the deep learning paradigm such as - Convolutional Neural Network(CNN), Convolutional Recurrent Neural Network (CRNN), Recurrent Neural Network(RNN), and using a randomized search algorithm chooses the model that is best to predict the target popularity class and request count in future. The paper considers the cloud as the master node which will train the deep learning model, and the base station is considered as the Slave node which will collect the raw data and cache accordingly to the master node and handle the content requests by the user.

In [12], the authors portrays a case when each multi-access edge server carries out their activity separately. Every MEC servers cannot carry out all computational and big data request originated from end devices and the edge server is not able to supply considerable amount of gains in overhead reduction of data exchange among users devices and data cloud. As a result ,joint computation, caching, communication, and control(4C) at the edge with MEC server collaboration is a must. To git rid of these limitations, the issue of joint 4C in big data MEC is articulated as an optimization problem whose objective is to optimize a linear combination of the bandwidth consumption and network latency cooperatively. Furthermore, the articulated problem is seen as a non-convex. Therefore, a proximal upper bound problem of the originally formulated problem is put forward. To look for a solution to the proximal upper bound problem, the Block Successive Upper bound Minimization(BSUM) method is implemented. Moreover ,different algorithms like- Douglas Rachford splitting method , Gauss-Southwell and Randomized selection rules , BSUM, etc were also compared and the BSUM method was found more suitable for 4C than other algorithms. It also explained 3C(Communication, caching and computation) and properly described why 4C is more suitable in this case.

In [14], it primarily discusses about the advantages of edge cache such as, when there is a joint effort with the edge computation,the edge cache can occur in many places, such as macro BSs (MBSs), small cell BSs (SBS), and mobile devices. The edge cache also caches the content based on integrative components, in which not only the popularity of the contents, but also the content active time, content size, and other elements should be considered. When we know the popularity of the content, the system can pre-cache the content according to the proactive content caching approach during the off-peak duration, or else, the reactive content caching strategy must be utilized. To lessen the limitations on the cache storage capacity and the radio resource, the coded caching is suggested. The contents are divided into multiple packets and the packet is used as side information for coded communication over the shared link are more sensible than only caching the most popular files. The edge cache moves the content closer to the user and allows the end user to get the content locally causing a reduction in the traffic load in the back-haul, which makes the SE(Spectral Efficiency) better and lessen the transmission delay in a massive amount. Therefore, Since the sum total of consumed energy consists of transport energy and the caching energy and the cost of edge cache is much less costly than the centralized cache limited by the back-haul transmission, and the corresponding EE(high energy efficiency) in Radio-Access Networks(RANs) can be greatly improved. They have done an extensive research on edge cache placement which includes the system architecture, key techniques, and their corresponding performances. The main, technical problems and performance of the reactive and proactive cache strategies for content placement are also briefly explained. Moreover, the coded and not coded cache are summarized entirely along with their advantages and disadvantages in various application environments. The performance gains from edge cache are highlighted and various main points on SE, EE, and latency are demonstrated. Finally, the future limitations and open issues are recognized for the convergence of the edge cache and 5G network for IoT.

In [15] they considered a multi-user MEC system in order to handle the issue of resource allocation policies and computation offloading strategies.The optimization objective that was set in this paper was the sum of the cost of intervals as well as energy consumption. A C-RAN dynamic resource allocation framework based on deep RL (DRL) was proposed in this paper that reduced duration and the energy consumed while keeping in check that every user demand is satisfied[8]. Computation offloading problems and resource allocation problems were proposed to be solved through a DRL-based theme in which the offloading of user computational tasks is proportional which was not seen to be done in other papers. They have defined the state, action and reward functions of the DRL agents and then gone off to formulate issues regarding the allocation of resources and made application of Deep Neural Network(DNN) to make an approximation of the action-value function of the action decision to draw out information directly from the present state.

The authors of the literary work in [16] used the Wolpertinger architecture to cache contents and improve the short and long term cache hit ratio. They then compared their agent with a deep Q network and showed that their work was better when it comes to runtime. In their proposed methodology, they have divided their framework into three parts, actor network, critic network and the K-nearest neighbor. They trained the policy using Deterministic policy Gradient(DDPG) for both the actor and the critic. The algorithm at first takes the current content request and cache state as input, and a proto actor is given as output. Afterwards, the proto actor given as input, is received by the KNN which then forms an action space based on l2 distance between each action. Finally, the critic network takes the action space as input and updates the actor network depending on the Q value. They compared their performance with other caching strategies in terms of cache hit rate for the short term and the long term and saw a greater cache hit rate.

In [17] they put emphasis on developing a smart content caching at mobile network edges to ease unnecessary traffic and enhance the productivity of content delivery. The information about users' preference is very helpful here and significant for productive content caching, yet often unavailable in advance, so they have used machine learning techniques to know the preference of the users based on historical demand data and decides about which of the video files is required to be cached at the MEC servers.They have come up with a multi-agent reinforcement learning (MARL)-based cooperative content caching policy for the MEC architecture where the preference of data by users' is not known and only the historical content demands can be observed. They figured out the cooperative content caching problem as a multi-agent multi-armed bandit problem and thus proposed a Multi Agent Reinforcement Learning based algorithm to come up with the solution of the problem. They also implemented Q-learning in MEC edge servers to know how to coordinate their caching decisions in multi-agent systems. MEC servers learn the Q-values of their own caching decisions and coexists with those of other MEC servers.The simulation experiments are carried out on a real data set from MovieLens and the quantitative results have shown that the proposed MARL based cooperative content caching scheme can reduce delay in a considerable amount while downloading contents and thus content cache hit rate can be improved when compared with other well-known caching strategies.

In [18] the authors have put forward the idea of implementing an integrated framework which is capable of enabling caching, dynamic orchestration of networking as well as computing resources to upgrade the performance of future-gen vehicular networks.They have formulated the strategy for resource allocation in this framework as a joint optimization problem, as they have taken into consideration the gains of networking, caching and computing as well.Since enormous complexity of the system arises when all three technologies are considered together, they decided to use a DRL approach.They made use of deep Q network to estimate a Q value-action function, that has been exploited in wireless networks, to attain resource allocation, automatically.The paper demonstrated that significant improvements in the performance of vehicular networks can be attained through caching, integrated networking, and computing.They have used a different type of hardware and virtualized vehicular networks that enable direct programming of vehicular network controls and abstraction of the infrastructure underneath, for a range of applications of connected vehicles, with better efficiency and greater flexibility in managing vehicular network.

In [19] the authors have optimized the QoE of wireless devices and caches contents, learning from human centric information. They proposed a echo state network based on conceptors which stores users historical mobility patterns and uses the result to predict users behavior. In the UAVs they planned to cache contents based on content popularity to reduce transmission delay. In the system model, they have grouped the remote radio(RRH) head using K-mean clustering and deployed UAVs with caching capabilities to serve the ground user and RRH. The caching in the UAVs are done during off-peak hours or during the docking period. They have also formulated the mobility,transmission and quality of experience model. The mobility model decides the placement of the UAV to ensure optimization of trajectory. In the transmission model, the communication between UAV to user, UAV to baseband unit(BBU) and RRH to users are defined. Finally, in the quality of experience model, they have taken into consideration, the delay of the user's content and the device type. Using all the models they formulated the minimization problem which finds the minimum rate to ensure the quality of experience and determine the minimum power required by the UAV to transmit. As their solution, at first they predicted the users behavior to store users location and content request distribution in the BBU. Using the predictions, the association of the users with the RRH is determined. The remaining users are clustered to associate with the UAV. After that the UAV decides the contents to cache, based on the users associated with them and then calculates the rate of transmission for each content. Using the rate of requirement, the UAV positions itself and finally transmits content to the users utilizing the least power.

Basically in [20], the authors discussed about delivering seamless connectivity to the aerial users by making use of transmission via coordinated multi-point (CoMP) alongside caching. In the cases of networks where there were clusters of cache-enabled SBSs, they came up with a novelty i.e. an upper bound expression was derived which focused on coverage probability that acted as a function of the system parameters.Their study showed that in the case of aerial users, the probability of coverage significantly made an improvement from 10% to 70%, once transmission via CoMP was used. This was particularly for the cases where a collaborative distance of 200 m was taken into consideration along with a lower SIR threshold. The contribution found in the paper is mainly improvement in the received SIR when cellular-connected UAVs in high altitudes are considered that made use of transmission via CoMP. Through this form of transmission nearby SBSs transfer the previously cached contents to the aerial UE where these contents are downloaded. It was also seen that, in this paper they used certain tools from stochastic geometry resulting in the derivation of a tighter upper bound on the content coverage probability. Content availability, target bit rate as well as collaborative distance were set as the parameters for achieving performance of an aerial user. The central point of this literary work was on analyzing CoMP transmission for the UAVs that were cellular-connected in networks that were cache-enabled.

The authors in [21] discussed about numerous UAVs, leading to the formation of a sub-network in the aerial region for assisting the vehicular sub-network on the ground by means of A2A and A2G communications.They talked about how UAVs can be used for disaster rescue and polluted area investigation with the help of two layer cooperative networking.They presented three types of communication link in multi-UAV-aided VANET which includes V2V links,A2A links and A2G links. For transmission of data amongst UAVs, the aerial sub-network used the A2A links. In A2A links for instance, XBee-PRO (IEEE 802.15.4) as well as Wi-Fi (IEEE 802.11), the radio interfaces which act in a heterogeneous manner can be taken into consideration as their operation in the unlicensed spectrum and can be combined seamlessly. The air–to-ground networks are primarily used for transmission of sensing data and are responsible for controlling information in between the ground and aerial sub-networks. Sub-networks in the ground are a type of sparse VANET, where intermittent V2V links are utilized for carrying out the interchange of information among vehicles. On the other hand, efficiency of the system control schemes significantly determine the cooperative networking performance. At first the base stations transmit the commands for scheduling. Upon receiving them, the UAVs as well as the rescue-vehicles start working in a cooperative fashion. Four controlling schemes are taken into account during the process of collaborative control. Those are Onboard control , Onboard sensing , Onboard processing and Onboard diagnosis.These schemes work in coordination for the control schemes to work in a productive manner.

The authors in their literary work [22] proposed a novel air-ground integrated mobile edge network that would work through scheduling and deployment of UAVs in order to provide aid in caching, communication as well as computing in edge networks.Proposal of the AGMEN contained numerous drone cells that were dispatched in a flexible way, for the provision of agile RAN coverage for the temporal and spatial changes of users and traffic of data. The AGMEN architecture that they proposed involved a two-layer networking architecture. One of the layers involved deployment of UAVs thereby creating a multi-UAV aerial network, mobile users and vehicles whereas the other layer which formed the ground network was made up through the RAN infrastructure. The UAVs had upgrades in the form of add-ons such as embedded processors, communication modules, sensors as well as storing devices which boosted these into becoming multi-functional network controllers. For transmission of information amongst UAVs, which includes coordinating and controlling information, sensing data to form a FANET, they used A2A communications through radio interfaces that worked in a heterogeneous manner and WiFi too. Some specific tasks were made possible to be conducted in this paper using exchange of information and control of mobility such as DTNs, monitoring of traffic, connectivity maintenance of wireless sensor networks, etc. For serving mobile users, self driving cars, IoT devices, etc. the heterogeneous RAN in the ground network that was proposed within the AGMEN architecture came into play which included small cells, macrocells and WiFi. In order to establish cooperation between the ground network and the aerial network, they made use of A2G communication-links. The ground control center controlled the UAVs mobility whereas collection of data from the aerial-networks were transmitted over to control center in the ground for advanced processing, so that proper exploitation of these data could be done. They have used wireless fronthaul connection so that the UAVs could properly serve BSs and also ensured large-scale sensing through UAVs. These lead up to users getting flexible internet access. It can be inferred from this paper that the UAVs play a major part as popular contents can be cached in these as well as scheduling of computing tasks can also be done through use of AGMEN.

Chapter 4

Proposed System Architecture

4.1 System Model

As first we are proposing a joint air-ground architecture for content caching as shown in figure 4.1 which will consist of both the ground base station and UAV. In the ground we have the base stations, which will cache contents dynamically. In the air, the UAVs will position themselves according to user mobility to support the caching for the mobile users. In related papers, the cloud server sends the global parameters chosen after every request that is being made by the users, i.e. both the GBS and UAVs send out data to the cloud server and the cloud server in turn provides information to the GBS and UAV after every request. This causes the backhaul traffic to be at a significant level. In order to overcome this limitation, our proposed methodology considers the fact that at the start of each time stamp, which is of 1 hour, the cloud server sends out the global parameters to the UAV and the GBSs only once in that whole time stamp. The GBSs and UAVs continously send out data to the cloud server, but the cloud server only sends out information once, at the starting of each time stamp. In this way, all of the data is being processed by the cloud server, but only after considering all of these data after every request that is generated. As only once the data gets sent out from the cloud server, it reduces the amount of backhaul traffic.

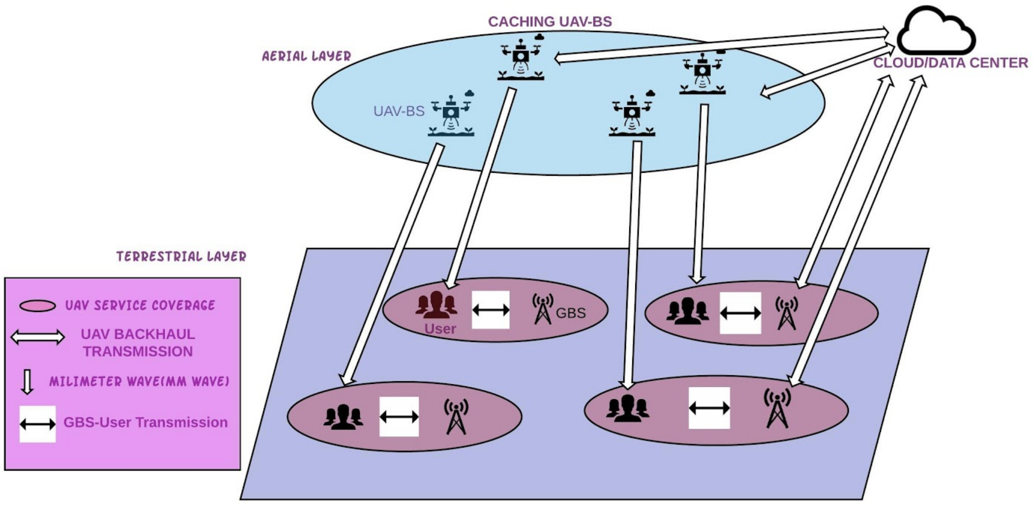


Figure 4.1: Joint air-ground architecture for caching.

4.2 Problem Formulation

The integration of the air and the ground networks leads to some constraints that must be satisfied to optimize the problem of efficient caching. In order to construct those necessary constraints, at first we needed to define some vectors. Throughout the paper, $G = \{g_1, g_2, \ldots, g_N\}$, has been used to refer to the number of ground base stations where g_N represents the capacity of the Server n to cache contents. The "N" in this case corresponds to the total number of servers of the GBSs. On the other hand, $U = \{u_1, u_2, \ldots, u_M\}$, has been used to denote the number of unmanned aerial vehicles where u_M represents the capacity of the servers of the UAVs to cache contents. The "M" in this case corresponds to the total number of servers of the UAVs. The data that we are considering from the MovieLens dataset, are the movies. So, we have chosen $D = \{d_1, d_2, \ldots, d_I\}$, to represent the total movie items and d_I corresponds to the i^{th} movie item's movie size. Alongside this, time has been assumed to be taken into slots where each slot represents a certain time period, represented by $T = \{1, 2, \ldots, T\}$. In addition to this, each user is being associated with the closest UAV-GBS pair. So, we have defined $r^t_{n,x,i} = \{r_{n,x,1}, r_{n,x,2}, \ldots, r_{n,x,I}\}$ where $r^t_{n,x,i} = 1$ represents the information that the data item,i, which is requested by the user,x, is being served by the closest GBS at time t. Similarly we have also defined $r^t_{m,x,i} = \{r_{m,x,1}, r_{m,x,2}, \ldots, r_{m,x,I}\}$ where $r^t_{m,x,i} = 1$ represents the information that the data item,i, which is requested by the user,x, is being served by the closest UAV at time t. Besides this, the caching state of GBS,n is defined as $c^t_{i,n} = \{c^t_{1,n}, c^t_{2,n}, \ldots, c^t_{I,n}\}$ where $c^t_{i,n}$ is the movie item stored in ith position of the nth GBS. Similarly the caching state of UAV,m is defined as $c^t_{i,m} = \{c^t_{1,m}, c^t_{2,m}, \ldots, c^t_{I,m}\}$ where $c^t_{i,m}$ is the movie item stored in ith position of the mth UAV at time t. Finally, to record the request frequency of movie item i at time t, $F^t = \{f^t_1, f^t_2, \ldots, f^t_I\}$ is introduced.

4.2.1 Computation model

1. At GBS(Ground Base Station): In the model that we are proposing, a requested $r^t_{n,x,i}$, where x is the x^{th} user, is executed by GBS, if GBS has enough computation power. We have considered g_n as the capacity of a GBS of server n where $n \epsilon N$. Furthermore, we define $y^{x \to n} \ \epsilon \ \{0, 1\}$ as a decision variable, which indicates whether or not GBS at server, n, has to compute the request $r^t_{n,x,i}$ by user x, where $y^{x \to n}$ is given by equation 4.1:

$$y^{x \to n}_x = \begin{cases} 1, & \text{if } \mathrm{r}^t_{n,x,i} \text{ requested by user x} \\ & \text{is computed at GBS, n} \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

   Each GBS at server,n, needs to satisfy the following constraint given by equation 4.2:

$$\sum_{i=1}^{I} d_i \ y^{x \to n}_x \cdot c^t_{i,n} \leq g_n \tag{4.2}$$

   which means that the summation of the size of all the movie items,i, that are cached in GBS,n and the decision of whether or not to offload the task, along with the caching state of the GBS, c_n, at a particular time t, has to be less than the capacity of the GBS at the particular server,n.

2. At UAV: In the case of the UAV, a requested task $r^t_{m,x,i}$, where x is the x^{th} user, is executed by UAV, if UAV has enough computation power. We have considered U_m as the available computational resources at UAV server $m \epsilon M$. Furthermore, we define $y^{x \to m} \ \epsilon \ \{0, 1\}$ as a decision variable, which indicates whether or not UAV at server,m, has to compute the task $r^t_{m,x,i}$ offloaded by user x, where $y^{x \to m}$ is given by equation 4.3:

$$y^{x \to m}_x = \begin{cases} 1, & \text{if } \mathrm{r}^t_{m,x,i} \text{ requested by user x} \\ & \text{is computed at UAV, } \mathrm{U}_m \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

   Each UAV at server,m, needs to satisfy the following constraint given by equation 4.4:

$$\sum_{i=1}^{I} d_i \ y^{x \to m}_x \cdot c^t_{i,m} \leq u_m \tag{4.4}$$

   which means that the summation of the size of all the movie items,i, that are being cached in UAV,m by users,x, and the decision of whether or not to offload the task, along with the caching state of the UAV, c_m,at a particular time t, has to be less than the capacity of the UAV at the particular server,m.

4.2.2 Mobility Model

In order to accurately predict the mobility of the users, the Random waypoint model is being applied which pauses time whilst changing the direction and/or speed. The mobile nodes (MNs) which in our case can be considered as the users, begin at a starting point where it resides for a certain amount of time. After the pause, the MNs starts moving towards the randomly chosen destination points. The speed of the MNs can be randomly chosen but in order to keep the model on a fixed track, we have chosen to keep the minimum and maximum speed fixed. In addition to this, a particular waiting time for each node is chosen in order to select the next nodes to be travelled to if the waiting time for the chosen node exceeds the one that has been fixed. Alongside these factors we also took into account the pause probability, the residual time and the initial speed which are crucial to successfully run the Random Waypoint model.

For this model to work efficiently, first of all we have taken a uniform distribution, defined a truncated power law distribution as well as defined an exponential distribution. The Random Waypoint Model took in the number of nodes i.e.(users), the simulation area which is the dimension of "x" multiplied by "y", the minimum and maximum velocity as well as the corresponding waiting time needed for the nodes. Once these were taken into account, the nodes' movements were assigned simply by getting the product of the direction and the node velocity. After the updated node position has been calculated, the distance between the new node position and the waypoint is calculated. If we arrived at the correct node, then the information for the arrived nodes is being updated. This decision of arriving at the desired node is being taken by comparing if the previously calculated distance is less than or equal to the velocity at which the node was travelling and also by checking if the waiting time for the node was less than or equal to 0.

If by any chance the nodes have surpassed the waypoint then a step back is done to ensure that the position of the arrived node is now assigned to that of the waypoint. In this way the Random Waypoint model is being used to predict mobility of the users and tackles one of the " C's " in the UAV assisted dynamic caching of infotainment on edge using Joint 4C.

4.2.3 Communication Model

In order to clearly define the different communication ways between the users, GBSs, UAVs as well as the cloud, some sort of channel model has to be defined which takes into account particular aspects of communication. So, what we do is build up a concrete channel model which has been explained below:

The time horizon of UAV-GBS is split into T equal time slices, indexed by t $\epsilon\{1, 2, ..., T\}$ and the UAV-GBS transmission occurs through the mmWave frequency band for the A2G links. For the channel model, the downlink transmission is only been taken into consideration due to the fact that the sheer volume of download data content is relatively far greater compared to the amount of service request that are being fetched.The path loss of LoS and NLoS links is modeled according to the log-normal shadowing model by picking specific channel parameters, given by the following set of equations 4.5 and 4.6:

1. For UAV:

$$\begin{aligned} l_u^{\text{LoS}}\left(u_m^t, v_x^t\right) = l_{FS}\left(d_0\right) + 10\mu_{\text{LoS}}\left(d_{m,x}^t\right) + \chi_{\sigma\text{LoS}} \\ l_u^{\text{NLoS}}\left(u_m^t, v_x^t\right) = l_{FS}\left(d_0\right) + 10\mu_{\text{NLoS}}\left(d_{m,x}^t\right) + \chi_{\sigma\text{NLoS}} \end{aligned} \tag{4.5}$$

2. For GBS:

$$\begin{aligned} l_g^{\text{LoS}}\left(g_n^t, v_x^t\right) = l_{FS}\left(d_0\right) + 10\mu_{\text{LoS}}\left(d_{n,x}^t\right) + \chi_{\sigma\text{LoS}} \\ l_g^{\text{NLoS}}\left(g_n^t, v_x^t\right) = l_{FS}\left(d_0\right) + 10\mu_{\text{NLoS}}\left(d_{n,x}^t\right) + \chi_{\sigma\text{NLoS}} \end{aligned} \tag{4.6}$$

where the attributes that have been used in the equations 4.5 and 4.6 represent the following meanings:

u_m^t = position of the m^{th} UAV in the t^{th}time index
g_n^t = position of the n^{th} GBS in the t^{th}time index
v_x^t = the x^{th} user's position in the t^{th}time index
$d_{m,x}^t$ = the distance from the m^{th} UAV to the x^{th} user in the t^{th} time index
$d_{n,x}^t$ = the distance from the n^{th} GBS to the x^{th} user in the t^{th} time index
$l_{FS}\left(d_0\right)$ = free space path loss from a reference distance

The $l_{FS}\left(d_0\right)$ is calculated by applying the following formula that takes into account a number of attributes:

$$l_{FS}\left(d_0\right) = 20\log\left(d_0 f_c 4\pi/c\right) \tag{4.7}$$

where the attributes in equation 4.7 represent:

d_0 = reference distance
f_c = carrier frequency
c = speed of light

The equation in 4.7 is used for both the cases of the UAV and the GBS as well, to calculate the free space path loss from a reference distance.
From the formula that we previously defined for calculating the path loss of line-of-sight (LoS) and non-line-of-sight (NLoS) links, we again took one parameter each for modelling the links using the shadowing model which are as follows:

XσLoS, XσNLoS = Shadowing random variable

The elevation angle $\phi_{t,m}$ between the UAV and the user plays a crucial factor in determining the probability of the LoS links given by the equation 4.8 :

$$\phi_m^t = h_m^t / \left(u_m^t - v_x^t\right) \tag{4.8}$$

where the attributes in equation 4.8 represent:

ϕ_m^t = elevation angle between UAV and User
h_m^t = height of the m^{th} UAV in the t^{th} time index.

The probability of the LoS link for the UAV is calculated through the following equation 4.9:

$$\mathrm{Pr}(\mathrm{LoS})_m = \frac{1}{1 + Ze^{-W(\phi_m^t - Z)}} \tag{4.9}$$

where the attributes in equation 4.9 represent:

Z,W = constants depending on environment

The elevation angle $\phi_{t,n}$ between the GBS and the user is a key factor in determining the probability of the LoS links. So, $\phi_{t,n} = Z$ is considered for this case. The probability of the LoS link for the GBS is calculated through the following equation 4.10:

$$\mathrm{Pr}(\mathrm{LoS})_n = \frac{1}{1 + Ze^{-W(\phi_{t,n} - Z)}} \rightarrow \mathrm{Pr}(\mathrm{LoS})_n = \frac{1}{1+Z} \tag{4.10}$$

The height and density of the buildings largely affects the probability of LoS connection. For instance, rural areas have very low density of buildings, i.e. there are less number of buildings in close proximity that results in higher LoS probability whereas an urban area exists a lower LoS probability. So, the total path loss is expressed as follows:

1. For UAV:

$$PL_{\mathrm{total}_m} = \mathrm{Pr}(\mathrm{LoS}_m) \times PL_{\mathrm{LoS}_m} + \mathrm{Pr}(\mathrm{NLoS}_m) \times PL_{\mathrm{NLoS}_m} \tag{4.11}$$

2. For GBS:

$$PL_{\mathrm{total}_n} = \mathrm{Pr}(\mathrm{LoS}_n) \times PL_{\mathrm{LoS}_n} + \mathrm{Pr}(\mathrm{NLoS}_n) \times PL_{\mathrm{NLoS}_n} \tag{4.12}$$

where each component of the equations 4.11 and 4.12 corresponds to:

$\mathrm{PL}_{\mathrm{LoS}_m} = l_m^{\mathrm{LoS}}\left(u_m^t, v_x^t\right)$

$\mathrm{PL}_{\mathrm{NLoS}_m} = l_m^{\mathrm{NLoS}}\left(u_m^t, v_x^t\right)$

$\mathrm{Pr}(\mathrm{LoS})_m =$ The probability of the LoS link for UAV

$\mathrm{PL}_{\mathrm{LoS}_n} = l_n^{\mathrm{LoS}}\left(g_n^t, v_x^t\right)$

$\mathrm{PL}_{\mathrm{NLoS}_n} = l_n^{\mathrm{NLoS}}\left(g_n^t, v_x^t\right)$

$\mathrm{Pr}(\mathrm{LoS})_n =$ The probability of the LoS link for GBS

Another significant factor which has to be closely looked at is the signal to noise ratio(SNR). Let γ_x denote the SNR of the x_{th} user, which can be expressed as follows:

1. For UAV:
$$\gamma_{x,u} = \frac{P_a \left| h_m \right|^2}{10^{PL_{\text{total}_u}/10} \sigma^2} \tag{4.13}$$

2. For GBS:
$$\gamma_{x,g} = \frac{P_a \left| h_n \right|^2}{10^{PL_{\text{total}_g}/10} \sigma^2} \tag{4.14}$$

where the transmit power of the cache node is being denoted by P_a , $\mid h_m \mid^2$ is the Rayleigh fading channel gain. This value of SNR has a certain threshold for each GBS, UAV. So, the communication model is defined in such a way that exceeding the threshold of GBSs causes the users to then communicate with the UAV and even if that fails, as a last resort the communication occurs between the user and the cloud. In this way the communication model was developed.

4.2.4 Caching Model

One of the most crucial parts is to cache the data that is going to be provided to the users. Till now what has been seen is that based on some factors the caching was done which mainly included popular content demand. Since caching of contents in the devices or nodes used is significant to the end users, i.e. the ground users in this case, the main problem remains on how to best predict this cache. Several caching policies are already in place such as context aware caching policy taking into account the contexts that users might face, for instance the diversity of devices, location, individual characteristics, etc. Alongside choosing the caching policy, another problem arises as to where to cache the data. A hierarchy of cache models including caching in the GBSs and UAVs and integration of all these two nodes proves to be a major challenge. As a result, for the caching problem we have proposed to maximize the global cache hit ratio, Ψ^t of the UAV and GBS to improve the backhaul traffic reduction and improve QoE. The global reward function can be formulated by the equation 4.15:

$$\Psi^{\mathrm{t}} = \frac{1}{R}\sum_{r=1}^{R} \chi \cdot \mathrm{l}^{\mathrm{r}} + \upsilon \cdot \mathrm{o}^{\mathrm{r}} \tag{4.15}$$

where l^r represents the hit ratio of the local server and o^r represents the hit ratio in the neighboring server at request r. The weights χ and υ is used to balance the rewards gained by the GBS and other UAV. When υ is 0, it means there is no cooperation with the neighboring server. So the weights can be tuned to set the cooperation among the servers.

Based on the the discussion that was made above, maximization of global cache hit ratio can be formulated by the equation 4.16 as an optimization problem which would then lead to the best caching decision to be chosen. In order to do this, the constraints of equation 4.2 and 4.4 have been taken into account.

$$\begin{aligned} &\max \tfrac{1}{T} \textstyle\sum_{t=1}^{T} \Psi^{\mathrm{t}} \\ &\text{s.t.} \ \textstyle\sum_{i=1}^{I} d_i \ y_x^{x\to n} \cdot c_{i,n}^t \leq g_n, \quad \forall d_i \in D, \forall n \in N, \mathrm{y}_x^{x\to n} \in \{0,1\} \\ &\qquad \textstyle\sum_{i=1}^{I} d_i \ y_x^{x\to n} \cdot c_{i,m}^t \leq u_m, \quad \forall d_i \in D, \forall m \in M, \mathrm{y}_x^{x\to m} \in \{0,1\} \\ &\qquad C_{i,n}^t \neq C_{i,m}^t, \quad C_n^t \in N, C_m^t \in M \\ &\qquad 1 \leq n \leq N, \quad 1 \leq m \leq M \end{aligned} \tag{4.16}$$

However, to reach the exact solution for a cache state to output the maximum global cache hit ratio, will be computationally very expensive as the above problem is NP hard.
In the later sections, in order to address the challenges of cooperative caching, a cooperative DRL approach is being proposed in which the GBS server would adjust caching decisions with the UAVs based on the actions and states, following a global reward.

Chapter 5

Cache Node Selection and Cooperative Caching Framework

5.1 Cache Node Selection Using K-Means Clustering

From the communication model that we proposed earlier we ended up with the channel model as well as the SNR which caused our joint air-ground architecture to segregate the users. Since the path loss, which is being calculated from the channel model, is dominated by the distance between the user and the UAV or GBS, the initial position of the UAVs and GBSs are taken into account. This can be done based on the joint air-ground architecture. The application of the K-means algorithm is done as the minimum distance from a centroid to each user can be attained. The K-means algorithm mainly consists of two parts namely initialization and iteration. For initialization, the m cluster centers are required to be determined. A position is picked at random to be the cluster center. From there on, in the iteration part, each users are assigned to its closest cluster center i.e, the set of the m_{th} cluster u_m is defined. After that,the new cluster center is recomputed until the iteration error is less than a threshold that had been previously set. In this way each particular UAV gets assigned to a cluster and the K-Means Clustering algorithm holds true leading to proper cache node selections.

Algorithm 1: Cache Node Selection Using K-Means Clustering:

```
Input: User's mobility matrix $\mathrm{M_n}$; Number of clusters K;
Output: Communication link a $\in \{0, 1, 2\}$.
1: while $t \leq$ T
2:      Use K-means to derive the centroid of clusters $[\mathrm{u_1, u_2, \ldots u_M}]$, where $\mathrm{u_m}$ is the
position of the mth UAV at the current time index t.
3:     while v $\leq$ X, where $v_x \in X$ do
4:         while n $\leq$ N where $\mathrm{g_n} \in$ G do
5:             Use the distance between the user and the GBS $\|\mathrm{v_x^t - g_n^t}\|$ to derive the
path loss $\mathrm{PL^{GBS}}$
6:             Calculate the user SNR $\gamma_{x,n}$
7:             Compare the user SNR $\gamma_{x,n}$ with the SNR threshold $\eta$GBS.
               If $\gamma_{x,n} > \eta$GBS:
                   set $c_x = 1$
                   a.append(0)
                   Store $\gamma_{x,n}$
8              end while
9:          while m $\leq$ U, where $\mathrm{u_m} \in$ U do
10:                Use the distance between the user and the UAV $\|\mathrm{v_x^t - u_m^t}\|$
                    to derive the path loss $\mathrm{PL^{UAV}}$
11:                Calculate the user SNR $\gamma_{x,m}$
12:                Compare the user SNR $\gamma_{x,m}$ with the SNR threshold $\eta$UAV.
13:                If $\gamma_{x,m} > \eta$UAV:
                     set $c_x = 1$
                     a.append(1)
                     Store $\gamma_{x,m}$
14:             end while
15:         If $\mathrm{c}_x \neq 1$ then
16:             a.append(2)
17: end while
18: return a
```

User's mobility matrix M_n is taken as the input from the Random Waypoint Model that has been generated. In addition to this, the communication links, a $\in \{0, 1, 2\}$ signifies that for a $= 0$ communication of the users are being done with the GBSs; a $=$ 1 means communication of the users with the UAVs; a $= 2$ shows that communication is done between the user and the Cloud. The K-means algorithm is used to generate the centroid of clusters $[\, u_1, u_2, \ldots, u_M]$ by taking the total number of clusters.

For V number of users, first of all, each GBS is checked to see if the Signal to Noise Ratio(SNR) value, γ_x, is greater than the SNR threshold. In order to calculate the SNR value, the PL_{Total} value is used from the Channel Model equations. Only if these constraints are met, then 0 value is appended to the connection matrix and the SNR values are stored for each possible connection,which will then be utilized in Algorithm 2. Else, the same set of conditions are being run for the UAVs and if the criteria are met, then the value of 1 gets appended to a. If the above conditions are not satisfied, then the value of 2 is set to a. In this way the selection of caching nodes is done by picking out caching nodes from clusters defined from the use of K-means clustering algorithm.

5.2 Caching using Multi-Agent Actor-Critic

There have been many proactive caching algorithms which cache popular contents in BSs' and UEs' during off-peak hours, but such techniques do not adapt to changing popularity and is not feasible when the contents are in large scale.

As a result, DRL is an appropriate alternative to tackle this issue. A DRL agent can be deployed at local servers and can dynamically adapt to changing conditions.According to [23], policy control problem can also be solved by DRL, i.e finding out which content to store in cache. Caching in DRL is usually done by replacing contents which were less frequently requested. But caching contents without co-operation with nearby servers will lead to inefficient caching, as both the servers may store the same content. As a result, cooperation is also very important while implementing DRL in edge networks.Ensuring cooperation should avoid duplicate data caching and fully utilize the computational capacity of each servers.

To ensure cooperation, each server has to know the caching state and request features of it's neighboring servers. But communicating information among servers will lead to additional overheads. So an efficient cooperative caching framework should be enabled.

After selecting where to cache, using the cache node selection algorithm, we would need to develop a dynamic caching policy which would decide what to cache in the UAV and GBS server. As a result, we have developed a multi-agent actor-critic algorithm to serve our purpose. For simplicity, we are assuming that a user request will be handled by the closest GBS and UAV, since they would have better SNR, rather than a server which is located far away.After receiving a user request, the server selects the caching action, and sends the features and actions to the cloud server, which acts as a centralized server. The server then evaluates the action performed by the UAV and GBS, and based on how they performed, sends the updated policies back to them to improve their respective caching actions. The evaluation is done using temporal difference error and is used to update the local server models. Since, our goal is to maximize global cache hit ratio, the content can be served by any one of the servers. As a result, the servers need to cooperate to reach the global goal. For example, if GBS cannot serve a content request, then the request is passed to the UAV server, depending if the UAV has the requested item.

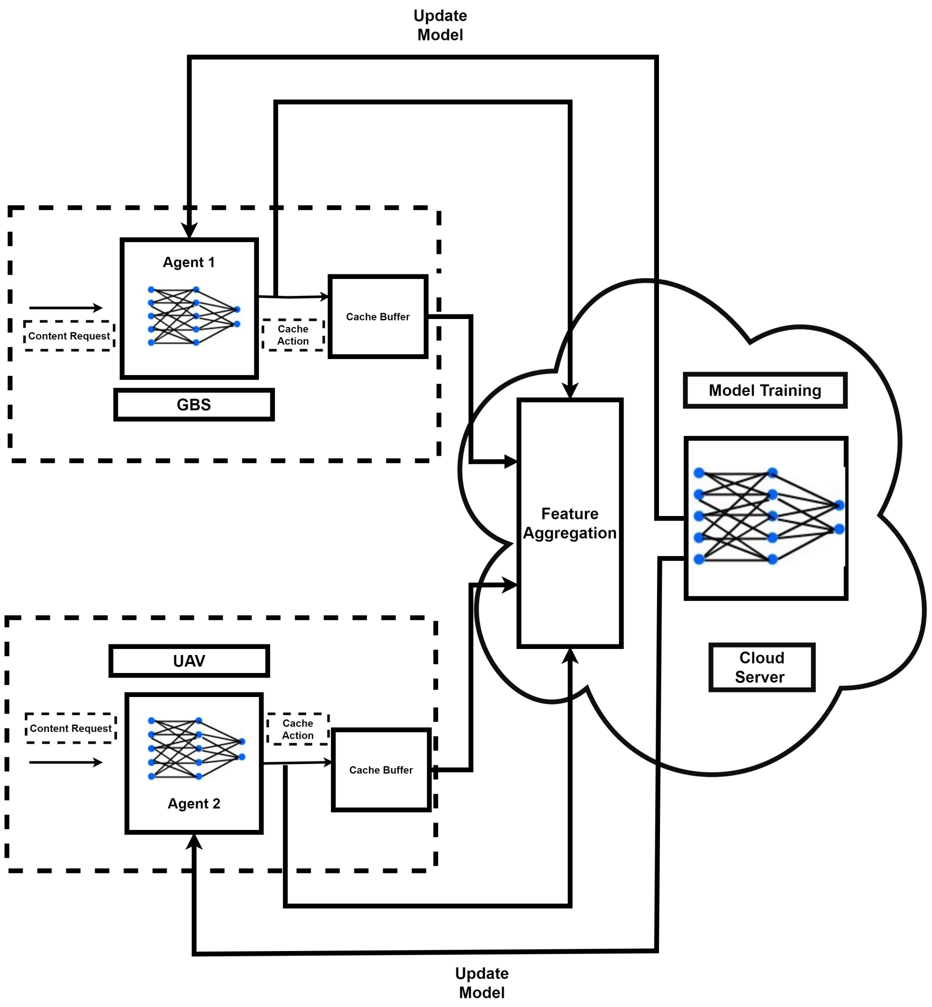


Figure 5.1: Cooperative Caching between UAV and GBS

Using centralized DRL, we will be training our model. The cache buffer in the GBS will determine a local action from the data requests that it received. The local cache actions, buffer states and request features from all GBS and UAV, will be sent to the central cloud server at each time step. The cloud server evaluates the actions sent by the local servers, and sends the updated policy depending on the parameters that it has received. The update that the central server sends will consist of only the parameters and cooperation reward. This will indicate the local servers as to which caching policy to be used, in the next time step. As a result, the transmission overheads will be insignificant. For instance, when a user request is received by either the UAV or GBS, they make the caching decisions based on their current local model. Then they send the caching states,actions and request features to the central server. The central server evaluates the received states and trains it's parameters to better evaluate the caching actions. The evaluation is made by the Critic network that sends the TD-errors to the local servers which uses them to update their action policy.

5.2.1 Reward Function, Feature Selection and Action Space:

Before designing the algorithm for the proposed framework, we need to define the reward function, feature selection and action space.

Feature Selection

Notation	Description
C_n^r	The cache state of GBS$_n$ at r
C_m^r	The cache state of UAV$_m$ at r
c_n^r	The nth data item in GBS at r
c_m^r	The mth data item in UAV at r
d^r	The data item in Cache at time r
Ω^r	The feature of a data item in Cache at r
α^r	The request arrival time of data item at r
ω^r	The total request frequency of data item at r
ω_s^t	Short-term request frequency of data item at t
ω_m^t	Medium-term request frequency of data item at t
ω_l^t	Long-term request frequency of data item at t
μ	Average user rating of item

Table 5.1: Feature Space Selection

It is extremely important to recognize the features that would lead to better cache hit ratio. Conventionally, a binary matrix $C_n^r = \left\{c_{i,n}^r \mid d_i \in D, g_n \in G\right\}$ and $C_m^r = \left\{c_{i,m}^r \mid d_i \in D, u_m \in U\right\}$ can be pre-defined for GBS and UAV respectedly, however it would be inefficient to train using a sparse matrix and will be computationally more expensive.Also it would not be realistic to know about the total data items beforehand in practical scenarios as stated in [24].

As a solution, we are proposing a fixed length caching state of size N for GBS and M for UAV which can be described as $C_n = \{c_1, c_2, \ldots, c_N\}$ and $C_m = \{c_1, c_2, \ldots, c_M\}$ respectively.Here $c_n = \{d_n, \Omega_n\}$, where d_n represents the item in nth location and Ω represents the feature.

For our work, we have considered 6 features $\{\alpha, \Omega, \Omega_s, \Omega_m, \Omega_l, \mu\}$ as described in Table 5.1. When the buffer is full these will be used to determine the content that is to be replaced when a new request comes in. The first two features,are used for conventional caching techniques such as LRU and LFU where α is the arrival time of each request, so that when a new request arrives, the buffer can replace the least recent one with the new one.The second one Ω^r can be defined as $\Omega = \sum_r f^r_{x,i}$ where $f^r_{x,i}$ represents the frequency of data item i at rth request by user x. Next we have also considered the short ω^t_s, medium ω^t_m and long term ω^t_l request frequency of items as introduced by [23]. They are calculated as follows:

$$\Omega^t_s = \sum_{t-\tau_s}^{t} \sum_r f^r_{x,i} \tag{5.1}$$

$$\Omega^t_m = \sum_{t-\tau_m}^{t} \sum_r f^r_{x,i} \tag{5.2}$$

$$\Omega^t_l = \sum_{t-\tau_l}^{t} \sum_r f^r_{x,i} \tag{5.3}$$

where τ_s,τ_m,τ_l are considered as 5,10 and 50 time steps. Finally, since the dataset we used had user ratings for each movie, we also considered using average rating μ as a feature since rating is treated as a popularity metric in deep learning paradigm. So we will be replacing the least rated movie in the buffer when a new request arrives.

Action Space

Notation	Description
a_1	Cache with least recently requested item
a_2	Cache with least frequently requested item
a_3	Cache with least frequently requested item in last τ_s steps
a_4	Cache with least frequently requested item in last τ_m steps
a_5	Cache with least frequently requested item in last τ_l steps
a_6	Cache with least rated item

Table 5.2: Action Space Selection

In our proposed framework, the output of the Actor network will determine the caching action and we will also need to define them like we selected our features. Instead of bothering about the replacement behavior of the cache buffer the action will simply output a caching policy which will be used by the servers to update their state. As a result, our proposed algorithm will output the caching policy, which will be followed in the next time step by the server to cache and replace items. By this approach our action space will be greatly reduced and will be discrete. The policy vector can be denoted by $A_n = \{a_1, a_2, \ldots, a_n\}$ where a_n represents the policy which will be used to make caching decisions.For example, a_1 represents replacing cached items that are least recently requested. So, when the actor network signals a_1 for the local server, the policy will cache a new item by replacing the least requested item throughout that time step. For our work, we have determined six action policies as defined in Table 5.2, which are; replacing items which were requested least recently, which had least request frequency,least request frequency in last 5,10,50 time steps and the least data ratings.

Reward Function

To ensure the cooperation among the UAV and GBS, we have considered the global cache hit ratio as an appropriate choice. In our framework, the content cached in the GBS as well as the content stored in the UAV, both contribute to our global reward. So, the reward function at *rth* request can be considered as the weighted sum of both the servers as stated in the problem statement 4.15, i.e

$$\Psi^{\mathrm{r}} = \chi \cdot \mathrm{l}^{\mathrm{r}} + \upsilon \cdot \mathrm{o}^{\mathrm{r}} \tag{5.4}$$

5.2.2 Caching Methodology

Algorithm 2: GBS-UAV Cooperative Caching Algorithm:

```
1: Initialize: Actor network $\pi_{\mathrm{n}}\left(\mathrm{C_n} \mid \Theta_{\mathrm{n}}^{\pi}\right)$ and $\pi_{\mathrm{m}}\left(\mathrm{C_m} \mid \Theta_{\mathrm{m}}^{\pi}\right)$ with weights $\Theta_{\mathrm{n}}^{\pi}$ and $\Theta_{\mathrm{m}}^{\pi}$ ; Target Actor network $\pi_{\mathrm{n}}'$ and $\pi_{\mathrm{m}}'$ with weights $\Theta_{\mathrm{n}}' \leftarrow \Theta_{\mathrm{n}}^{\pi}$ and $\Theta_{\mathrm{m}}' \leftarrow \Theta_{\mathrm{m}}^{\pi}$ ; Critic network $\mathrm{Q}_{n,m}\left(\mathrm{C1}, \mathrm{C2}, \ldots\ldots, \mathrm{C_{n,m}}, \mathrm{a_{n,m}} \mid \Theta_{\mathrm{n,m}}^{\mathrm{Q}}\right)$ with weights $\Theta_{\mathrm{n,m}}^{\mathrm{Q}}$ ; Target Critic network $\Theta_{n,m}' \leftarrow \Theta_{n,m}$ ; Cache Buffer state $\mathrm{C_n}$ and $\mathrm{C_m}$; Replay Buffer $\mathrm{B}_{n,m}$
2: for $t=1,\ \mathrm{T}$ do
3:     Select action $\mathrm{a_n^t} = \pi_{\mathrm{n}}\left(\mathrm{C_n} \mid \Theta_{\mathrm{n}}^{\pi}\right)$ and $\mathrm{a_m^t} = \pi_{\mathrm{m}}\left(\mathrm{C_m} \mid \Theta_{\mathrm{m}}^{\pi}\right)$ with current policy
4:     for $\mathrm{r}=1, \mathrm{R}^{t}$ do
5:         if $C_n^r$ and $C_m^r$ not full then
6:             if $\mathrm{r}_i^t$ in $C_n^r$ or $C_m^r$
7:                 Update cache and hit ratio then end
8:             else
9:                 Update in $C^r$ which has better Snr
10:                  Update hit ratio then end
11:        else
12:            Keep track of caching state $C_n^r$ and $C_m^r$
13:            if $\mathrm{r_i^t}$ in $\mathrm{C_n}$ or $\mathrm{C_m}$ then
14:                Perform $\mathrm{a_n^t}$ or $\mathrm{a_m^t}$, then send the state $\mathrm{C}_{\mathrm{n}}^{r-1}$ or $\mathrm{C}_{\mathrm{m}}^{r-1}$, action, new state $\mathrm{C}_{\mathrm{n}}^{r}$ or $\mathrm{C}_{\mathrm{m}}^{r}$ to the cloud server and update the hit ratio.
15:            else
16:                Perform $a^t$ in C which has better Snr then send the state $\mathrm{C^{r-1}}$, action, new state $\mathrm{C^r}$ to the cloud server and update the hit ratio.
17:    end
   In cloud:
18:    The cloud server calculates reward $\psi_{\mathrm{n}}^{\mathrm{r}}$ and $\psi_{\mathrm{m}}^{\mathrm{r}}$ for G and U and store $(\mathrm{C_n^r}, \mathrm{a_n^r}, \psi_{\mathrm{n}}^{\mathrm{r}}, \mathrm{C_n^{r+1}})$ and $(\mathrm{C_m^r}, \mathrm{a_m^r}, \psi_{\mathrm{m}}^{\mathrm{r}},\ \mathrm{C}_{\mathrm{m}}^{r+1})$ for all $r$ it received during $t$ in $\mathrm{B_{n,m}}$
19:    Sample a random mini batch of S transitions $\left(\mathrm{C}_{\mathrm{n,m}}^{s}, \mathrm{a}_{\mathrm{n,m}}^{s}, \psi_{\mathrm{n,m}}^{s}, \mathrm{C}_{\mathrm{n,m}}^{\mathrm{s+1}}\right)$ from $\mathrm{B_{n,m}}$
20:    Set $y_{n,m}^{s} = \psi_{n,m}^{s} + \gamma Q_{n,m}\left(C_1^{s+1} \ldots \cdot C_{N,M}^{s+1}, \pi_{n,m}\left(C_{n,m}^{i+1}\right) \mid \Theta_{n,m}' Q\right)$
21:    Calculate TD based on present parameters:
$\Phi_{\mathrm{n,m}} = 1/\mathrm{S} \sum_{\mathrm{s}}\left(\mathrm{y_{n,m}^s} - \mathrm{Q_{n,m}}\left(\mathrm{C_1^s} \ldots \mathrm{C_{N,M}^s}, \mathrm{a_{n,m}^s} \mid \Theta_{\mathrm{n,m}}^{\mathrm{Q}}\right)\right)$
22:    Update the critic network $Q_{n,m}$ by minimizing the Huber loss: $\mathrm{z}\left(\Phi_{\mathrm{n,m}}\right)$
23:    Update Critic target network: $\Theta_{\mathrm{n,m}}'^{\mathrm{Q}} \leftarrow \tau \Theta_{\mathrm{n,m}}^{\mathrm{Q}} + (1-\tau)\Theta_{\mathrm{n,m}}'^{\mathrm{Q}}$
24:    Send $\Phi_{\mathrm{n,m}}$ to GBS and UAV
   The UAV and GBS actor network is updated by:
25:    For GBS: $\nabla_{\theta n \pi}\mathrm{J} = \nabla_{\theta n \pi} \log \pi_{\mathrm{n}}\left(\mathrm{C_n}, \mathrm{a_n}\right) \Phi_{\mathrm{n,m}}$
26:    For UAV: $\nabla_{\theta m \pi}\mathrm{J} = \nabla_{\theta m \pi} \log \pi_{\mathrm{m}}\left(\mathrm{C_m}, \mathrm{a_m}\right) \Phi_{\mathrm{n,m}}$
27:    Update GBS Actor target network: $\Theta_{\mathrm{n}}'^{\pi} \leftarrow \tau \Theta_{\mathrm{n}}^{\pi} + (1-\tau)\Theta_{\mathrm{n}}'^{\pi}$
28:    Update UAV Actor target network: $\Theta_{\mathrm{m}}'^{\pi} \leftarrow \tau \Theta_{\mathrm{m}}^{\pi} + (1-\tau)\Theta_{\mathrm{m}}'^{\pi}$
29:    $\mathrm{C}_{\mathrm{n}}^{r-1} \leftarrow \mathrm{C}_{\mathrm{n}}^{r}$
30:    $\mathrm{C}_{\mathrm{m}}^{r-1} \leftarrow \mathrm{C}_{\mathrm{m}}^{r}$
31:    end
```

Finally, this is our proposed algorithm, which consist of the critic and actor networks implemented in the central server and local buffers accordingly.

Now, the work flow of the algorithm that we are proposing will be discussed below: At a given time step, based on the user position and user requests, the SNR is calculated using Algorithm 1. As a result, if a connection is possible to a server, will be known during that time period. Then each users request is handled by the nearest UAV-GBS pair. If the item is not available in either of the servers then the data is fetched from the cloud server to serve the user, and the item is cached in either UAV or GBS that has a better SNR value.The server will then process the content request and make a caching decision based on the current caching action policy as updated at the beginning of the time step. Then, the states,actions and new states is sent to the central server at the end of the time period. In the central server, the Critic network will evaluate the rewards based on the features and actions as well as calculate the TD-error. This will allow the Critic network to update it's parameters by minimizing the loss function. Finally, the TD-error will be sent to the UAV-GBS pair to update their caching policies. The process will repeat in the next time-stamp.

Actor Network

The actor networks are located in the local servers, i.e in the UAV and GBS, which takes caching state and request features as input. The network is denoted by $\Theta_{\mathrm{n}}^{\pi}$, which seeks the optimal caching policy by mapping the state C to action a. In each time step, the actions are choosen by the UAV and GBS based on the current parameters of the network.

$$a_n^t = \pi\left(C^t \mid \Theta^{\pi}\right) \tag{5.5}$$

Critic Network

Critic network: For evaluating the values of the selected actions performed by the local server, the critic network is used. As it's input, it takes all the caching state and actions performed by the GBS and UAV at time t. The action policy of a particular server is determined by the global states and actions performed by the UAV and GBS at time t. Then, the job of the critic is to calculate the TD-error,

$$\begin{gathered}\Phi_{n,m} = \frac{1}{S}\sum_{s}\left(y_{n,m}^{s} - \gamma Q_{n,m}\left(C_1^s, \ldots, C_{N,M}^s, a_{n,m}^s \mid \Theta_{n,m}^{Q}\right)\right) \\ y_{n,m}^{s} = \psi_{n,m}^{s} + Q_{n,m}^{\cdot}\left(C_1^{s+1}, \ldots, C_{N,M}^{s+1}, \pi_{n,m}\left(C_{n,m}^{s+1}\right) \mid \Theta_{n,m}^{Q}\right)\end{gathered} \tag{5.6}$$

where γ is the discount factor which enables a balance between recent and future accumulated reward, S is the size of the mini-batch that will be sampled, and $\Theta_{n,m}^{Q}$ is the network parameters.
After calculating the TD error the critic network is updated by the Huber loss function as defined in 5.7:

$$\text{Huber loss, } \mathrm{z}(\Phi_{n,m}) = \begin{cases} 0.5\left(\Phi_{n,m}\right)^2/\text{beta}, & \text{if } \left|\Phi_{n,m}\right| < \text{beta} \\ \left|\Phi_{n,m}\right| - 0.5 * \text{beta}, & \text{otherwise} \end{cases} \tag{5.7}$$

And for the GBS and UAV, the actor network is updated using the policy gradient:

$$\begin{aligned} \nabla_{\theta \mathrm{n} \pi} \mathrm{J} &= \nabla_{\theta \mathrm{n} \pi} \log \pi_{\mathrm{n}} \left(\mathrm{C}_{\mathrm{n}}, \mathrm{a}_{\mathrm{n}}\right) \Phi_{\mathrm{n,m}} \\ \nabla_{\theta \mathrm{m} \pi} \mathrm{J} &= \nabla_{\theta \mathrm{m} \pi} \log \pi_{\mathrm{m}} \left(\mathrm{C}_{\mathrm{m}}, \mathrm{a}_{\mathrm{m}}\right) \Phi_{\mathrm{n,m}} \end{aligned} \tag{5.8}$$

Chapter 6

Simulation Results

6.0.1 Data Pre-processing and Visualization

For our simulation part, we are using MovieLens dataset [25] to implement the content caching policy. This dataset is mainly composed of rating data of certain movies by different users. The MovieLens 1M dataset records around 1 million ratings of users along with User ID, Movie ID and timestamp. The movie rating process is analogous to the content request process. We have assumed that when an audience is rating a movie at a particular time it means the user is also requesting the movie file in that specific time. It is also assumed that the server is updating the caching decision every hour, so the timestamp is divided into 1 hour each.

First of all, we explored the dataset to know about it's contents and how we can use it to simulate edge caching in MEC servers and UAV.

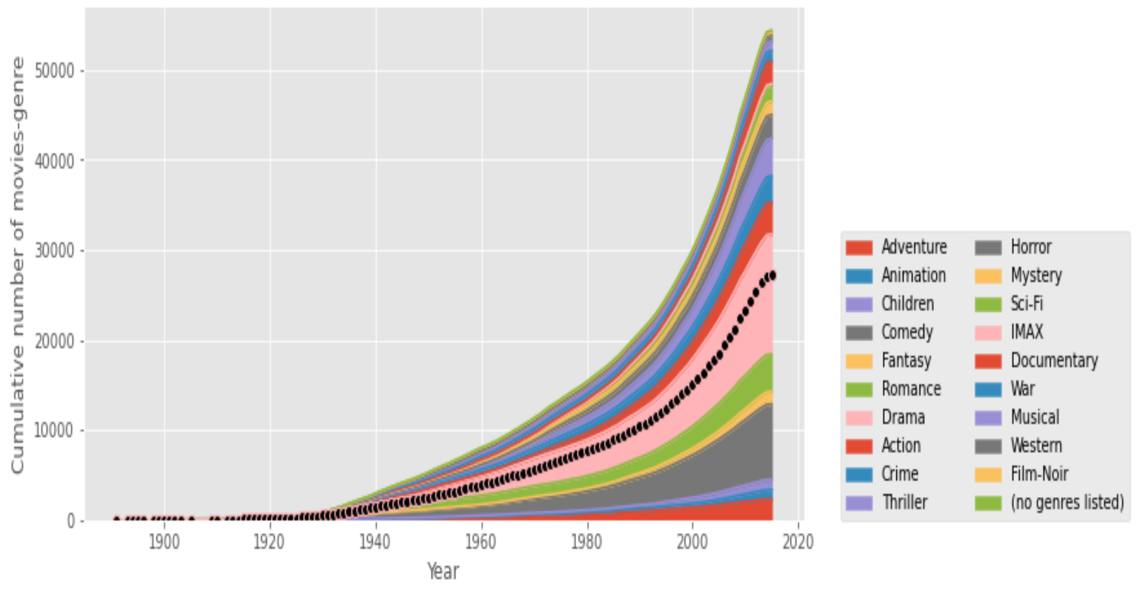


Figure 6.1: Cumulative or total number of movie genres for which the rating has been provided

In the figure 6.1, we have analysed the dataset to find the cumulative or total number of movie genres for which the rating has been provided. Comedy and dramas are the top genres here.

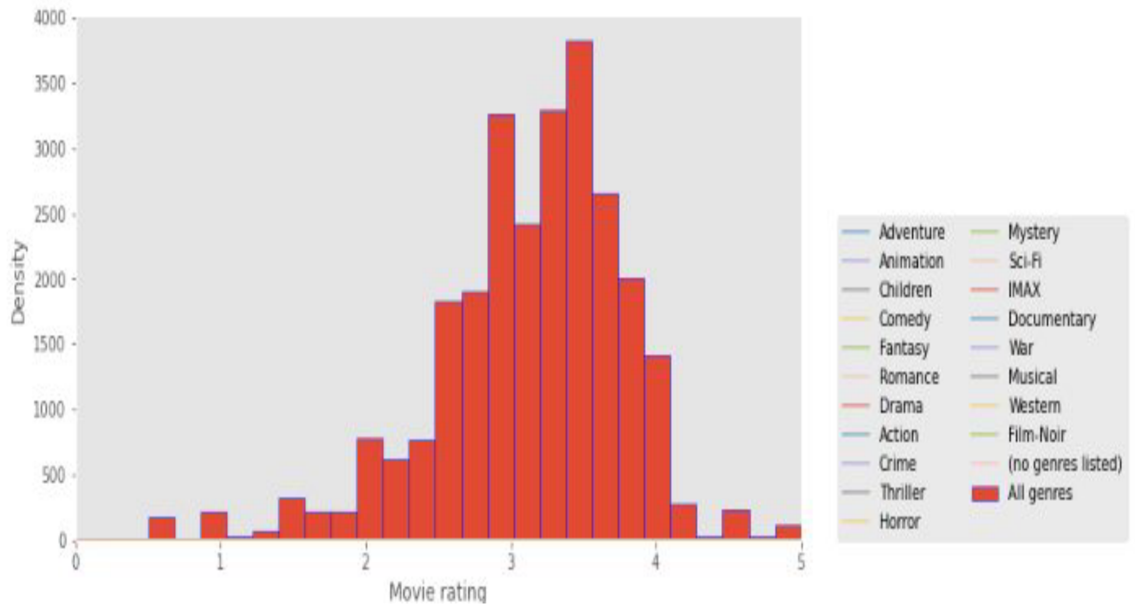


Figure 6.2: Density of movie ratings

This figure in 6.2, quantifies the density of movie ratings and shows that movie ratings of around 3.5 value are more in number, mean rating and those movie genres are mainly drama, comedy and also mystery.

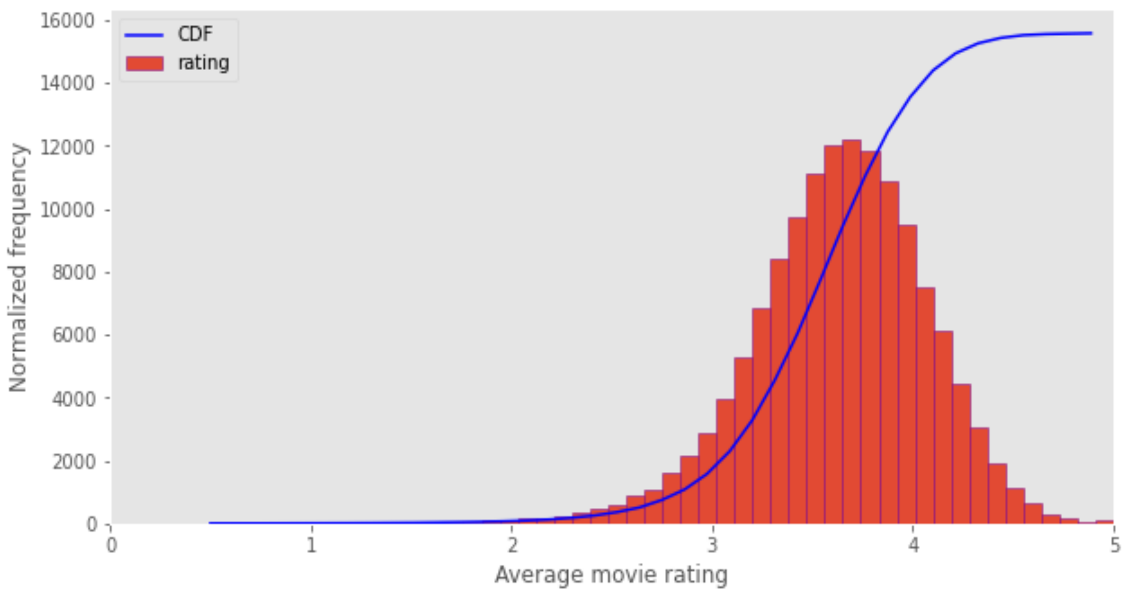


Figure 6.3: Average movie ratings

This figure in 6.3, shows the average movie ratings, with 95 percent of the average movie ratings above 2.5. This can be explained by the fact that users watch better movies due to good ratings of those movies.

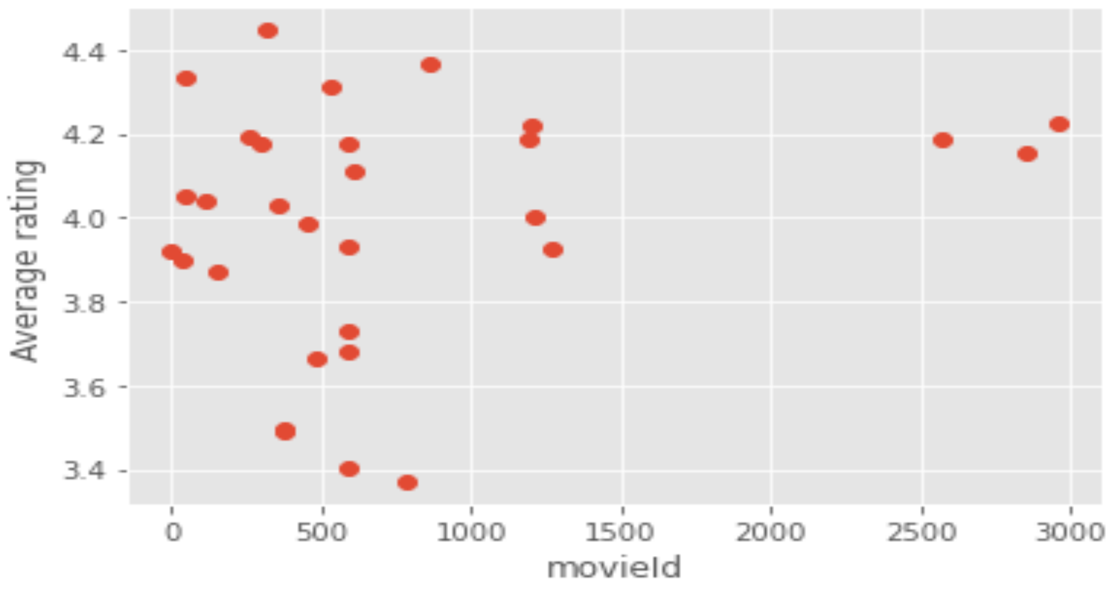


Figure 6.4: Average ratings of most popular movies

This plot in 6.4, shows the movieIds which have more average rating. It shows that movieIds less than 1500 have more average ratings.

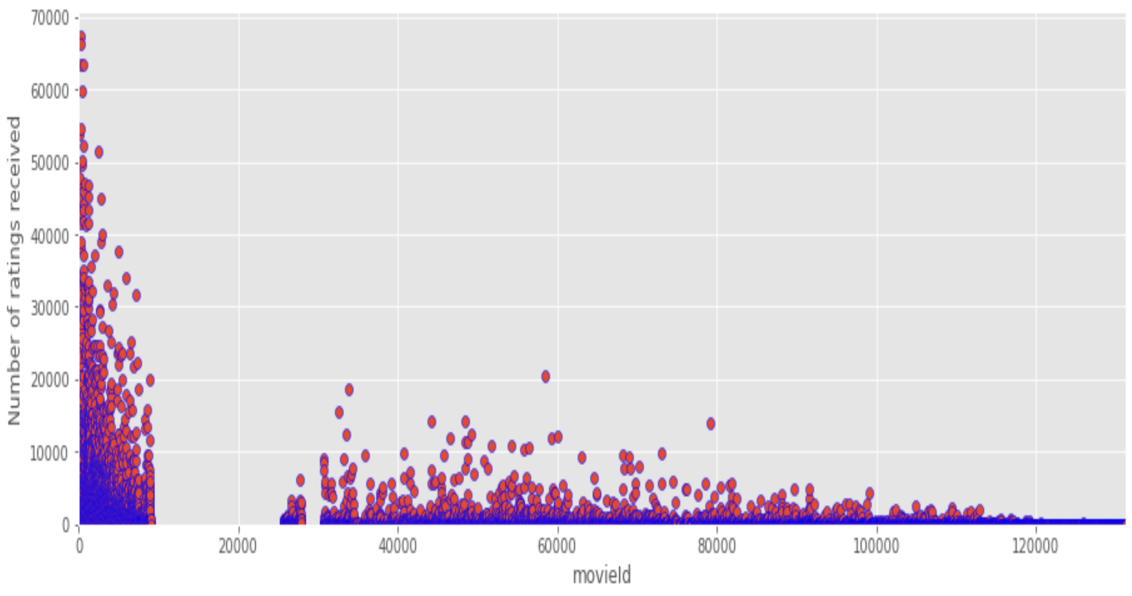


Figure 6.5: Movie IDs with highest rating

Figure 6.5, shows the number of ratings received for each type of Movie Ids which shows the Ids with the highest rating.

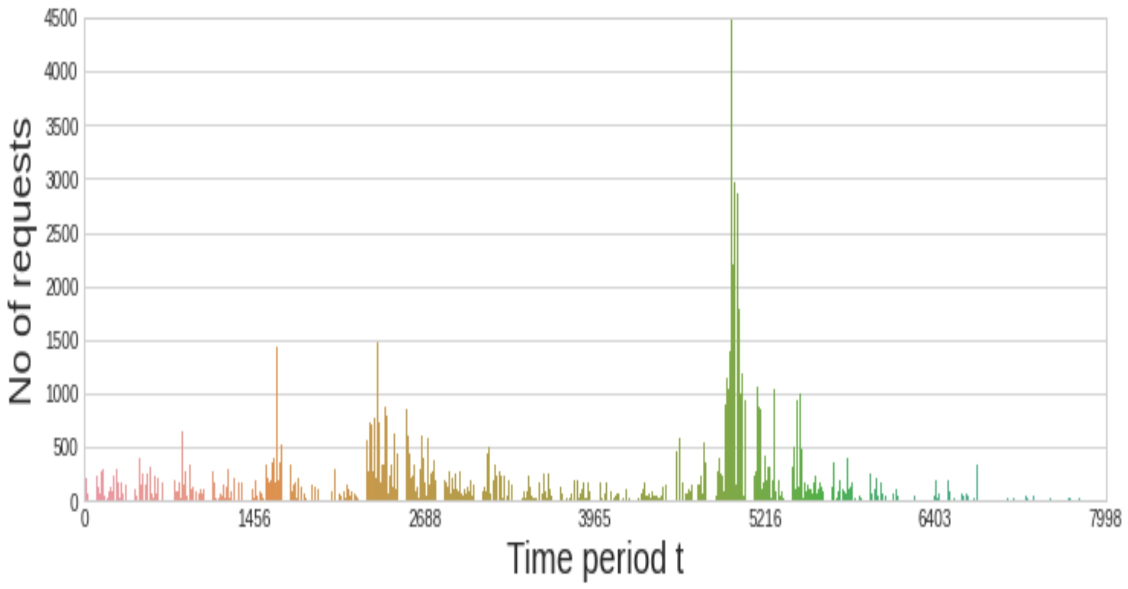


Figure 6.6: Number of Movie requests in each time period, t

This exploratory data analysis and visualizations which have been shown have helped us to understand the number of requests made per time stamp. For our work we have considered the ratings given by the users as the requests made, which is a valid assumption also done in previous works. As a result, ratings of movies given in an hour is considered as the number of requests by each user in each time period, t and shown in the plot 6.6.

6.0.2 Cache Node Selection Simulation

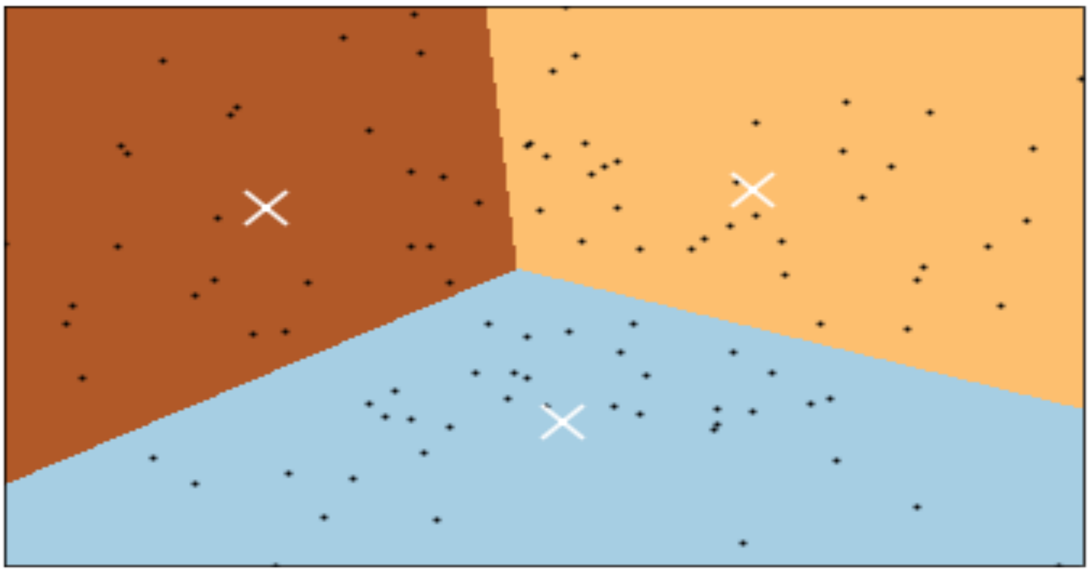


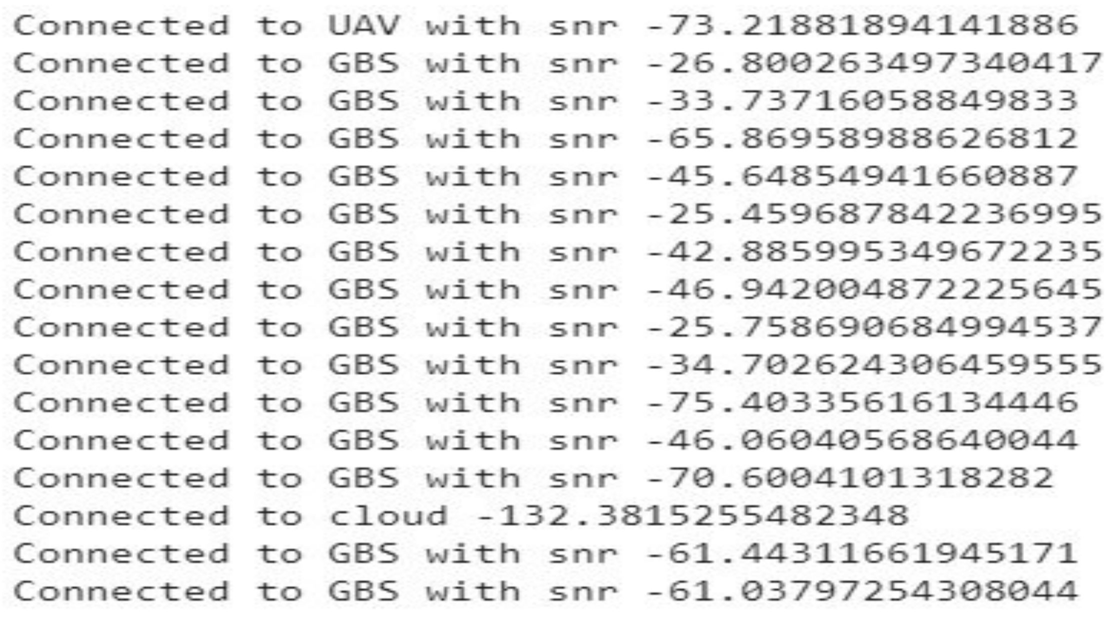
Figure 6.7: Clustering of UAVs according to user mobility where each UAVs are marked with white cross

Here the K-means clustering algorithm has been used as the estimator where the total number of clusters is taken to be equal to the number of UAVs that we have as each UAV is capable of serving one specific cluster of users. In this case the cluster number = 3. We also set a tolerance threshold = $1e^{-4}$ which is a hyper parameter used to more accurately run the K-means clustering algorithm which is also used as a terminating value for the segmenting error. In order for the algorithm to work on the huge number of users, it has been fitted on the user mobility matrix to give us a good prediction. The graph below depicts the ideal positioning of the UAVs once run on the user mobility matrix and it is clearly visible that there are 3 distinct regions due to the 3 clusters for the UAVs to serve.

Descriptions and Notations	Value
UAV-BS height,H	60m
Users,x	100
GBS, N	3
UAV, M	3
GBS link carrier frequency, fc,GBS	2.1GHZ
UAV-BS transmit power,P_{UAV}	20dBm
GBS transmit power,P_{GBS}	40dBm
UAV link carrier frequency, fc,UAV	38GHZ
Free-space reference distance, d_o	5
Shadowing random variables, $X\sigma$LoS, $X\sigma$NLoS	8.3, 8.27
UAV, snr	-93
GBS, snr	-97
Environment dependent constant, Z,W	11.9, 0.13
Path loss Exponent,	2

Table 6.1: Environmental Parameters

In Table 6.1, we have several parameters which we are mainly using for cache node selection. The first one is the height of the UAV from the user which is taken to be 60m. We have considered the snr threshold of UAV and GBS to be -93 dBm and -97dBm which is considered as a good connection as in [26] Then we have GBS link carrier frequency which is the frequency of the Ground base station that operates at 2.1 GHZ. We also have UAV-BS transmit power which gives the power of UAV transmission as 20dBm and Ground base station transmitting power as 40dBm. The UAV links are operated in mmWave band with carrier frequency of fc,UAV = 38GHZ. Finally we have shadow random variables, environmental constants and path loss exponents which are used to calculate the total path loss.



```
Connected to UAV with snr -73.21881894141886
Connected to GBS with snr -26.800263497340417
Connected to GBS with snr -33.73716058849833
Connected to GBS with snr -65.86958988626812
Connected to GBS with snr -45.64854941660887
Connected to GBS with snr -25.459687842236995
Connected to GBS with snr -42.885995349672235
Connected to GBS with snr -46.942004872225645
Connected to GBS with snr -25.758690684994537
Connected to GBS with snr -34.702624306459555
Connected to GBS with snr -75.40335616134446
Connected to GBS with snr -46.06040568640044
Connected to GBS with snr -70.6004101318282
Connected to cloud -132.3815255482348
Connected to GBS with snr -61.44311661945171
Connected to GBS with snr -61.03797254308044
```

Figure 6.8: Output of cache node selection according to snr values

The Figure 6.8 shows the SNR values of each user being connected to a corresponding node based on the threshold, while running the cache node selection algorithm.

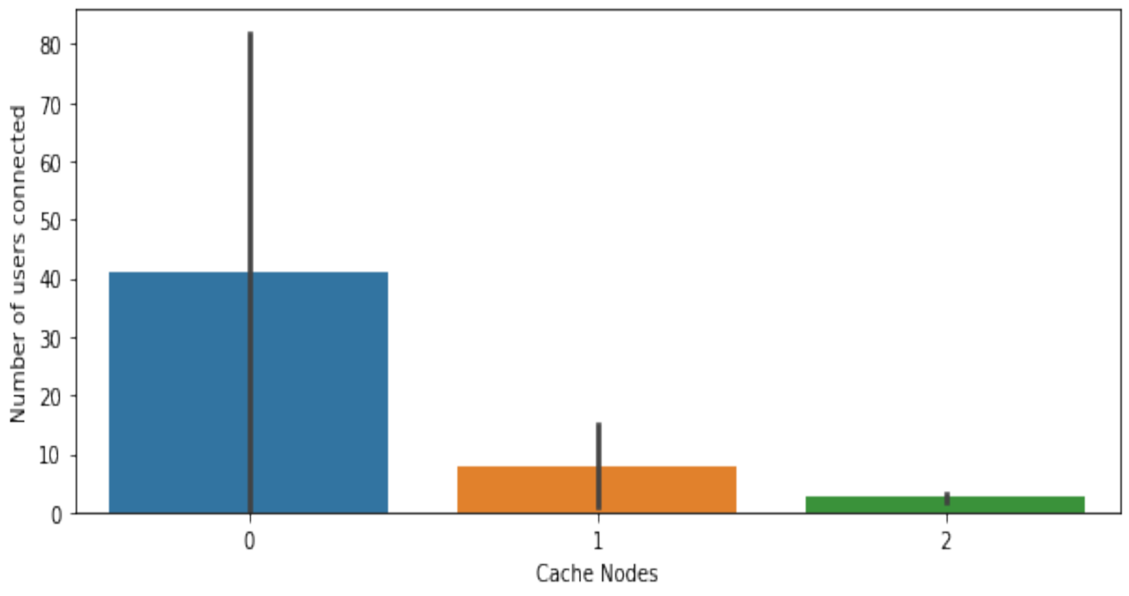


Figure 6.9: Number of Users connected in each server

Finally, in 6.9 we have a visualization which shows the connection of 100 users to a corresponding node after averaging through 1000 time steps. Most number of users get connected to cache node 0 which are the GBSs, and those of which that didn't get connected to a GBS, gets connected to the cache node 1, that are the UAVs. If neither of them provides a good connection, the user has to retrieve it's data from the cache node 2 which is the cloud server.

6.0.3 Caching using MAAC

For our research, we compared our proposed algorithm to two other conventional caching techniques used in edge caching.

- Least Recently Used(LRU): The method caches item based on the time they arrived i.e if the cache buffer is full and a new item is to be cached, then the algorithm replaces the item that arrived least recently.
- Least Frequently Used(LFU): In this method the request frequency of an item is tracked, so that when a new item is requested the the algorithm replaces the least frequently requested item from the cache server that is full.

Both this algorithm caches items that are relatively popular. But struggles to provide good hit rates when the popularity of requested item is uniform. This is where our proposed algorithm does better. Firstly, our dynamic approach does not depend only on the popularity of contents as a result it is able to adjust quickly when the popularity of content keeps changing or does not follow a pattern. Secondly, due to our collaborative approach the same item is not cached at both places which is not the case for the conventional approaches. Thirdly, since our proposed algorithm makes decision based on multiple features of a data item, allows it to make better caching decisions at each time step.

MAAC network Parameters:

We have considered the actor network with two fully connected hidden layers of shape 256 and 128 with learning rate of 0.001. In the critic network we have two fully connected hidden layers of shape 128 and 64 units with learning rate of 0.004. We have used Adam as the optimizer for both networks. And finally we initialized the replay buffer, $B_{n,m}$ with size 10000, and set the mini-batch, S sample size to 28.

Hyperparameters	Values
Buffer Size,$B_{n,m}$	10000
Batch Size,S	28
GBS storage,G	50
UAV storage,U	25
Movie Size,D	5
Action Size	6
Gamma,γ	0.9
Tau,τ	0.002
Reward weight,χ	0.7
Reward weight,υ	0.3
Actor FC1	256 neurons
Actor FC2	128 neurons
Critic FC1	128 neurons
Critic FC2	64 neurons
Actor Learning Rate	0.001
Critic Learning Rate	0.004

Table 6.2: MAAC Hyperparameters

Caching Algorithm Simulation

In the figures below, we have compared the cache hit ratios for different storage capacities using our proposed model along with the conventional approaches. Using the normal hours we have done all the simulations below.

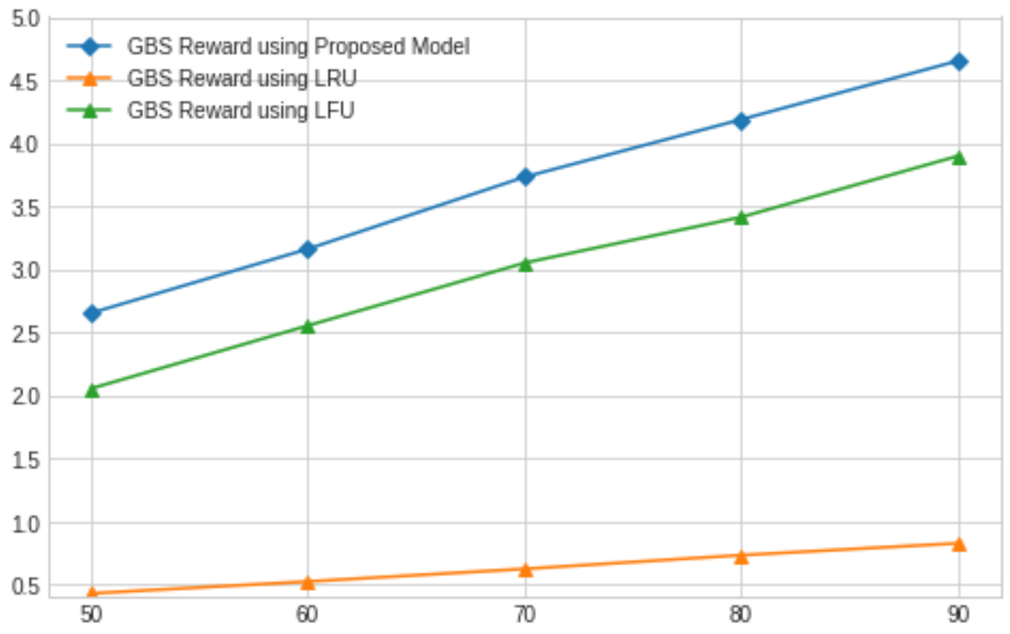


Figure 6.10: Comparing Cache hit ratio for different GBS Storage Capacity

In the figure above in 6.10, there is a linear rise in all three algorithms's cache hit ratios as we increase the GBS's storage capacity. Our proposed algorithm achieves the best hit ratio in all cases. However, the lru struggles to get a high Mec reward even when the storage capacity is high. Meanwhile, the lfu and our proposed algorithm kept a consistent gap along all storage capacities.

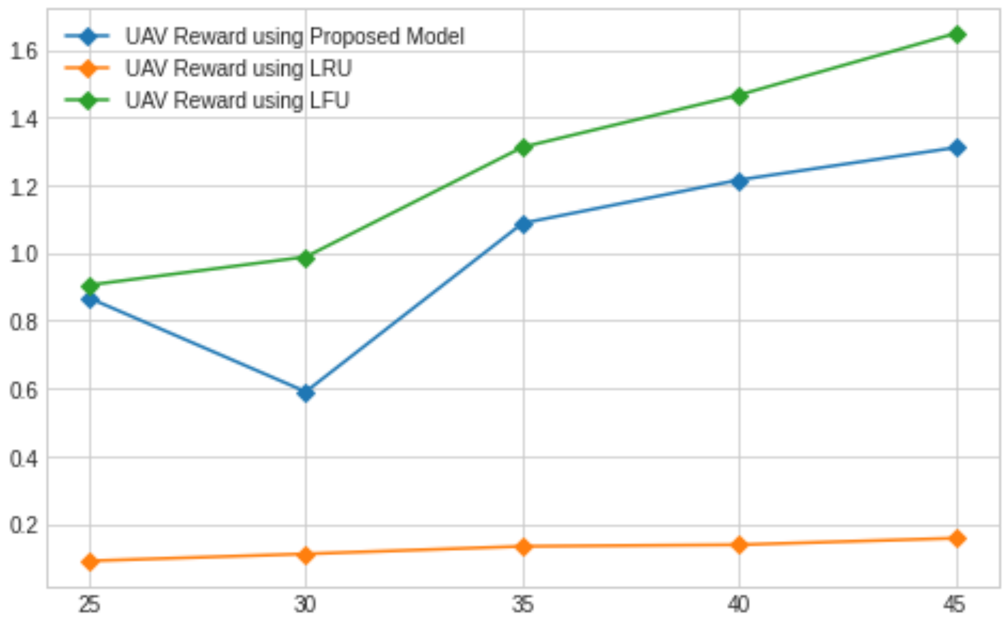


Figure 6.11: Comparing Cache hit ratio for different UAV Storage Capacity

In Figure 6.11, there was not enough increase in the hit ratio for LRU. However, the LFU achieved a better reward than our proposed model. This is because we first check for a connection in GBS and cache our item in the server which has a better snr value. Since GBS are also closer to most requests, the most requested items are cached in GBS and not in the UAV. As a result, to achieve a greater global hit ratio, the algorithm is sacrificing it's performance for the case of UAV. The performance decreases initially, because the proposed algorithm was adjusting it's cooperation, and then the increase is linear as we increase the UAV's storage.

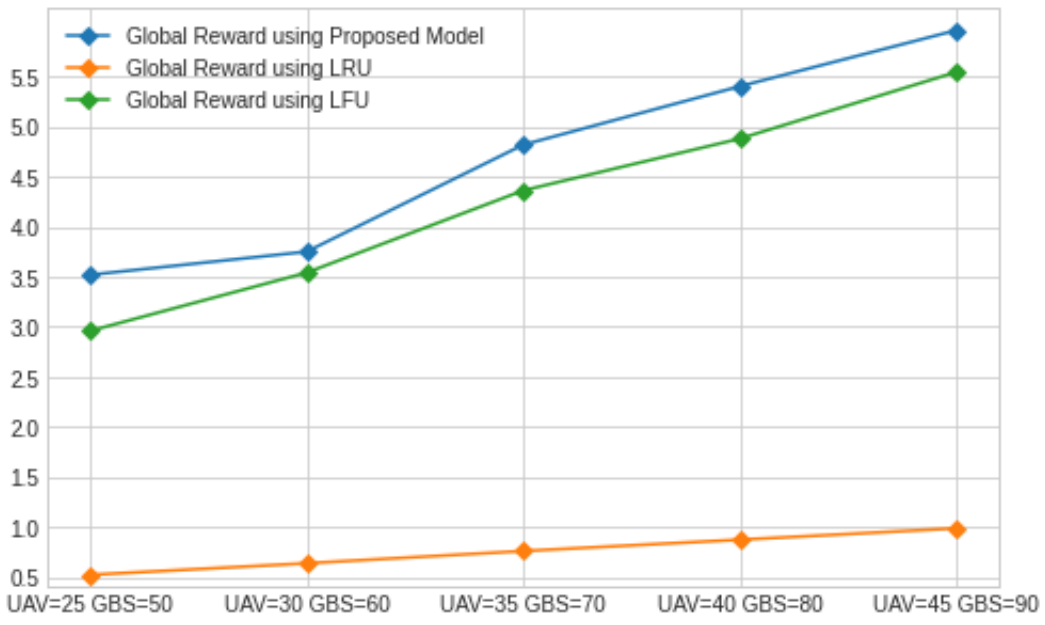


Figure 6.12: Comparing Global cache hit ratio for different GBS and UAV Storage Capacity

Finally, in Figure 6.12, we have shown the change in global reward as with different UAV and GBS storage. LRU performed the worst among all the algorithms. Our proposed model achieved the highest reward in all sizes. However, there is a smaller gap with LFU initially because of the drop in UAV's reward initially to adjust cooperation. But, afterwards there is a consistent gap between the proposed algorithm and LFU as the storage sizes where further increased.

Caching in Normal Time

The reason we used UAVs' was to assist GBS in situations where a GBS is not enough to handle the flood of user requests and another GBS might be located far away. In such a scenario, there will be a greater transmission delay to fetch the file from a GBS or from the central server located far away. This will result in a deterioration in user experience.

As a result, the purpose of this work is to show better cache hit ratio in situations when there is a high number of user requests i.e in peak hours. Alongside in situations where the number of requests is normal.
First we will be comparing the result of the algorithms in normal hours. For the simulation, we chose time period 1500-2000 from the Movie Lens data set, discarding a period where the total number of user request was less than 15. The average number of request in a given hour during this period was 137.

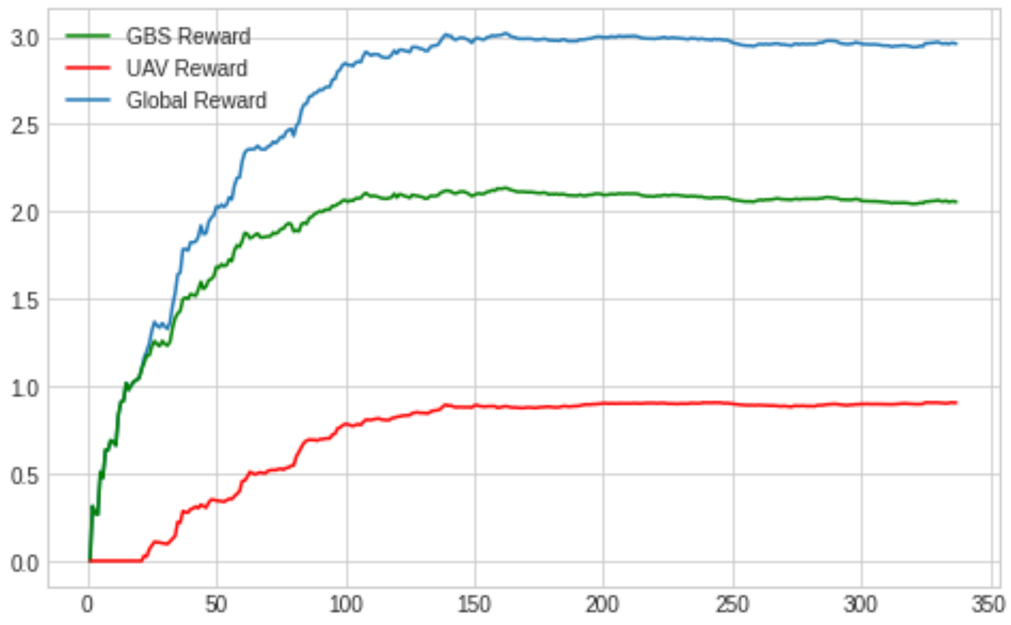


Figure 6.13: Comparing Cache hit ratio using LFU during normal hours

The Figure 6.13 shows the cache hit ratios for LFU during normal hours. There is a gradual increase for the first 100 hours both for the GBS and UAV and then becomes constant.

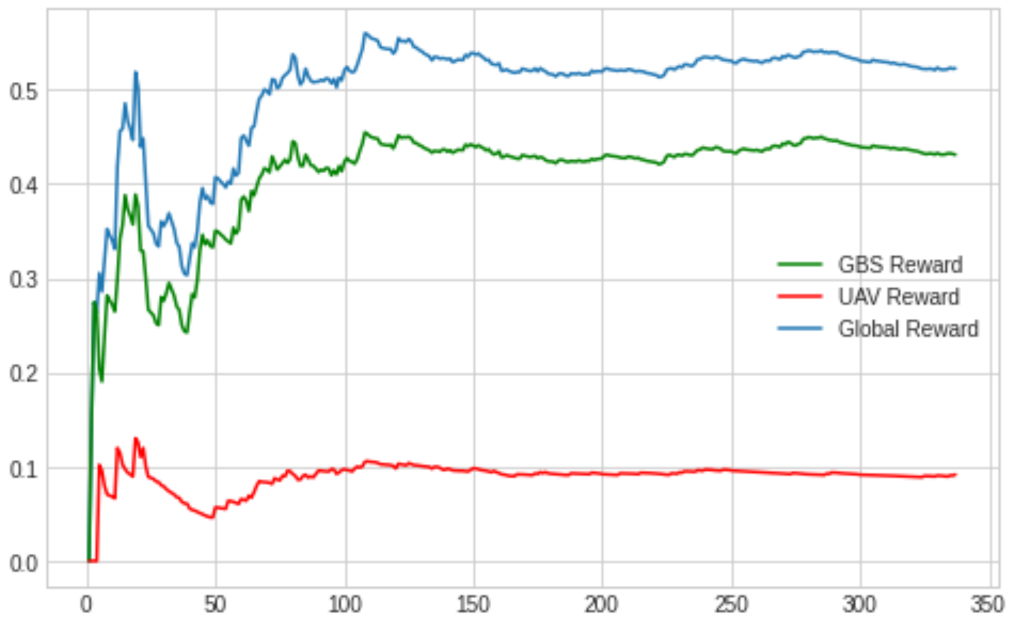


Figure 6.14: Comparing Cache hit ratio using LRU during normal hours

In Figure 6.14, after the initial rise, there is a temporary fall,afterwards the reward is constant for all time steps.

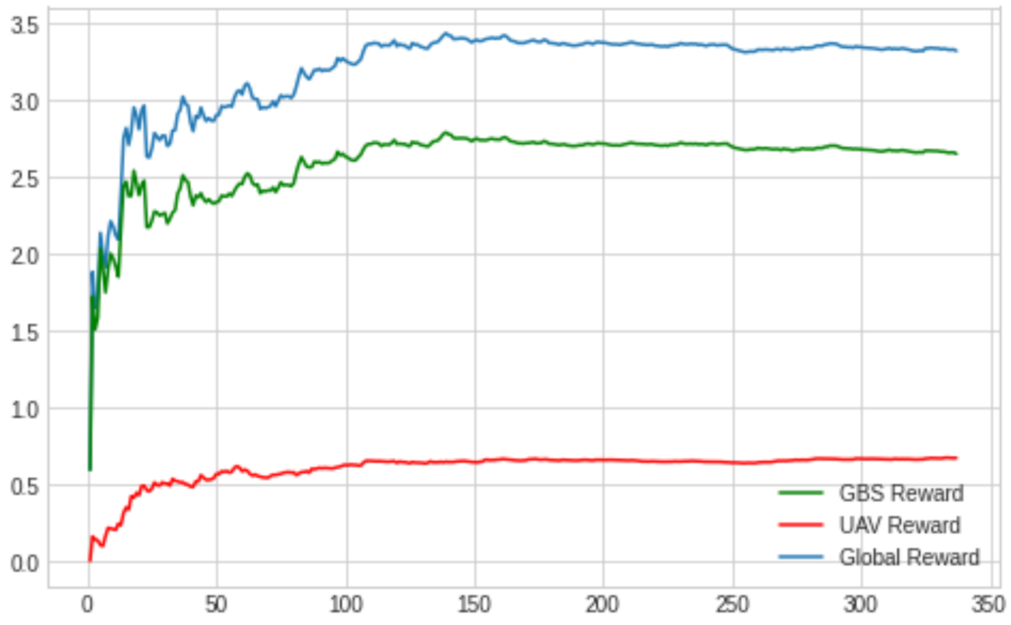


Figure 6.15: Comparing Cache hit ratio using Proposed algorithm during normal hours

The proposed model in figure 6.15 rose sharply and then stayed constant for all time steps. The UAV reward was lower than that of LFU's however the MEC reward was greater.

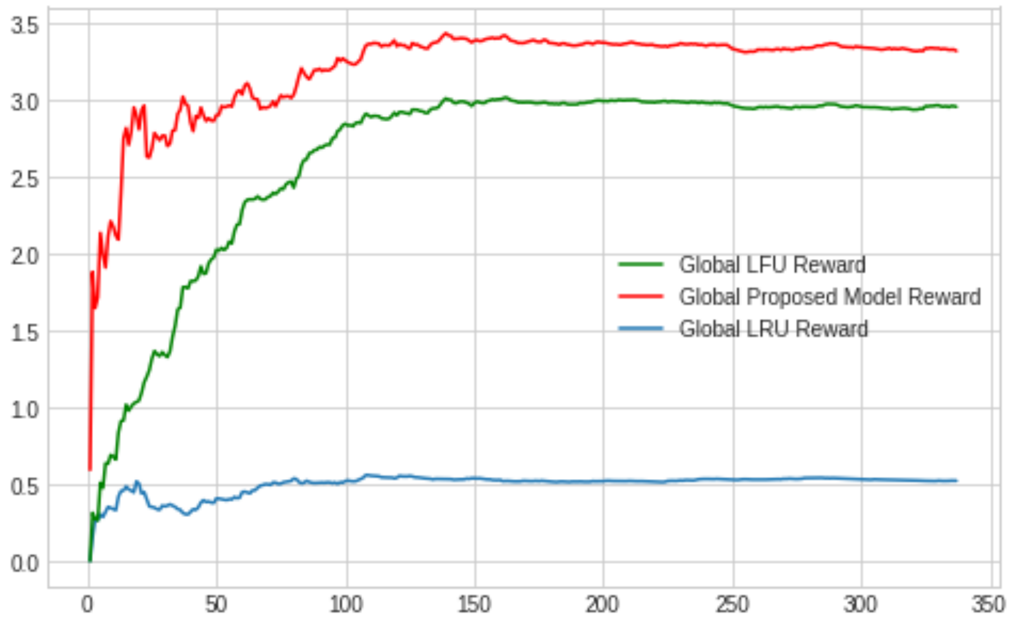


Figure 6.16: Comparing global cache hit ratio for all Algorithms in normal hours

Finally, in Figure 6.16 the proposed algorithm achieved greater global reward throughout the simulation. However, we have ran the proposed model twice using the weights of the first iteration in the second iteration. This resulted in the sharper rise in global hit rate in the beginning. The first iteration had a poor start because the model was initialized with random weights and as a result it took few time periods to adjust it's parameters to achieve a higher global hit rate. This also shows, if a request pattern is repeated, which is the case for people going for work everyday on the same time requesting almost similar items. The proposed algorithm takes more efficient caching actions resulting in better global reward.

Caching in Peak Time

Finally we will be comparing the result of the algorithms in peak hours. For the simulation, we chose time period 4800-6000 from the Movie Lens data set, discarding a period where the total number of user request was less than 15. The average number of request in a given hour during this period was 314.

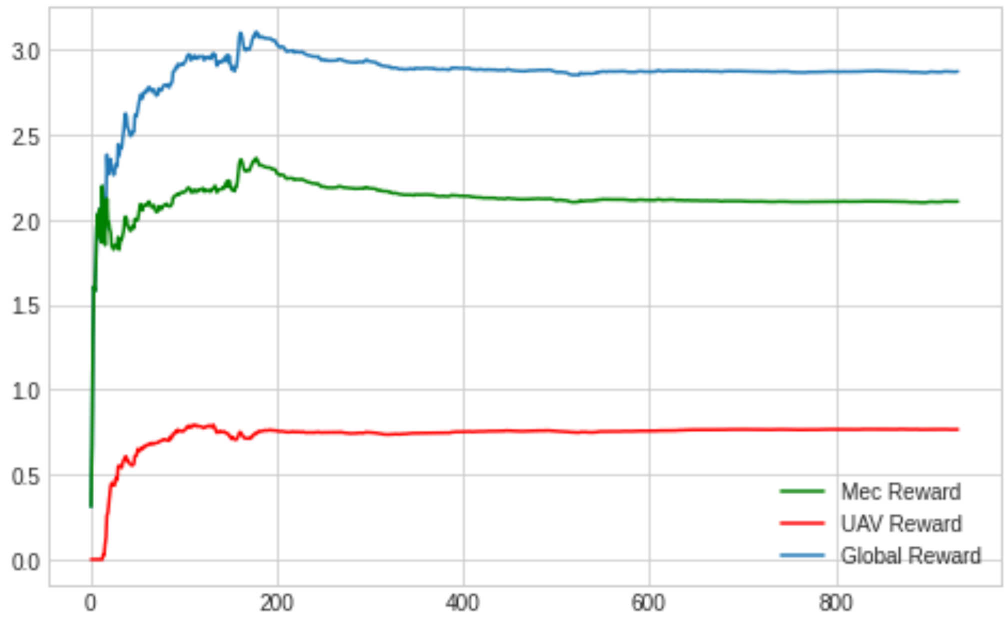


Figure 6.17: Comparing Cache hit ratio using LFU during peak hours

In Figure 6.17 the LFU shows a similar shape that it showed during normal hours. Initially increasing and then having a constant hit ratio for the rest of the simulation. The hit ratio has improved than what it was during the normal period. This is because, since the number of requests per hour increased. There is a greater chance for users to request the items more frequently, and since LFU caches based on the frequency of items it provided a better hit ratio as well.

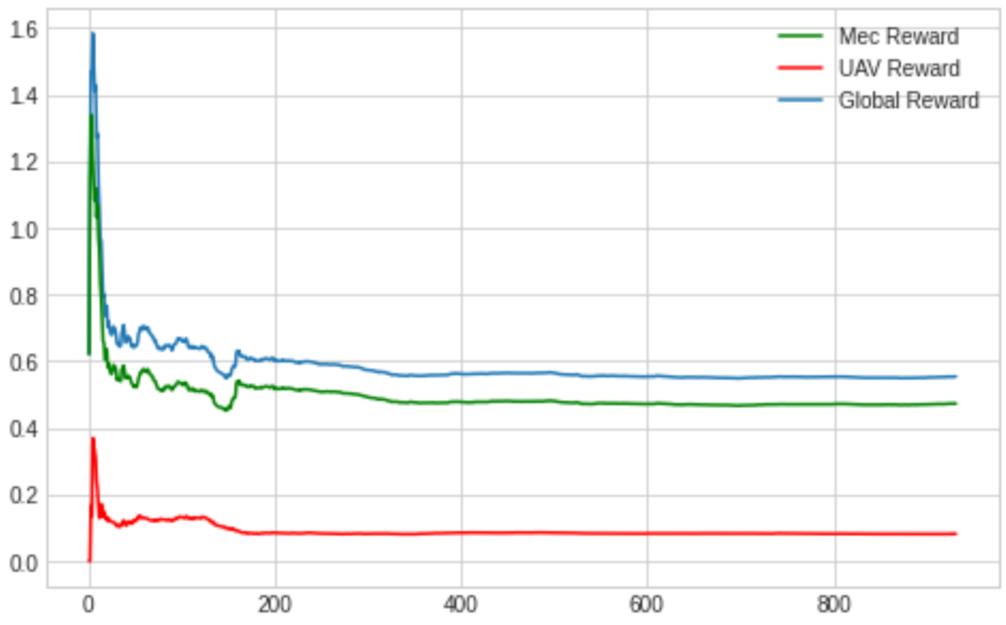


Figure 6.18: Comparing Cache hit ratio using LRU during peak hours

For the peak hours in Figure 6.18, the LRU struggled to provide a good hit ratio. There is an increase initially, however the hit rate gradually falls after few time steps and remained constant afterwards. The reason for this is the randomness of LRU. Out of the three algorithms LRU is more prone to caching newer items. As a result, it cached items that were not frequently requested by the users during that period.

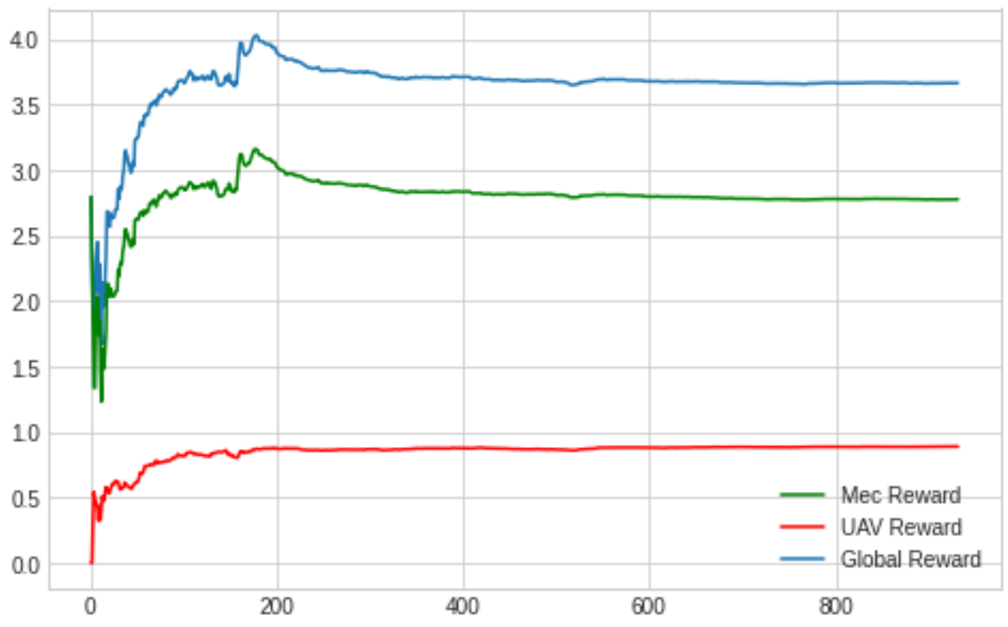


Figure 6.19: Comparing Cache hit ratio using Proposed algorithm during peak hours

In Figure 6.19 the proposed algorithm had a higher initial rise than in the normal period and then remained constant.

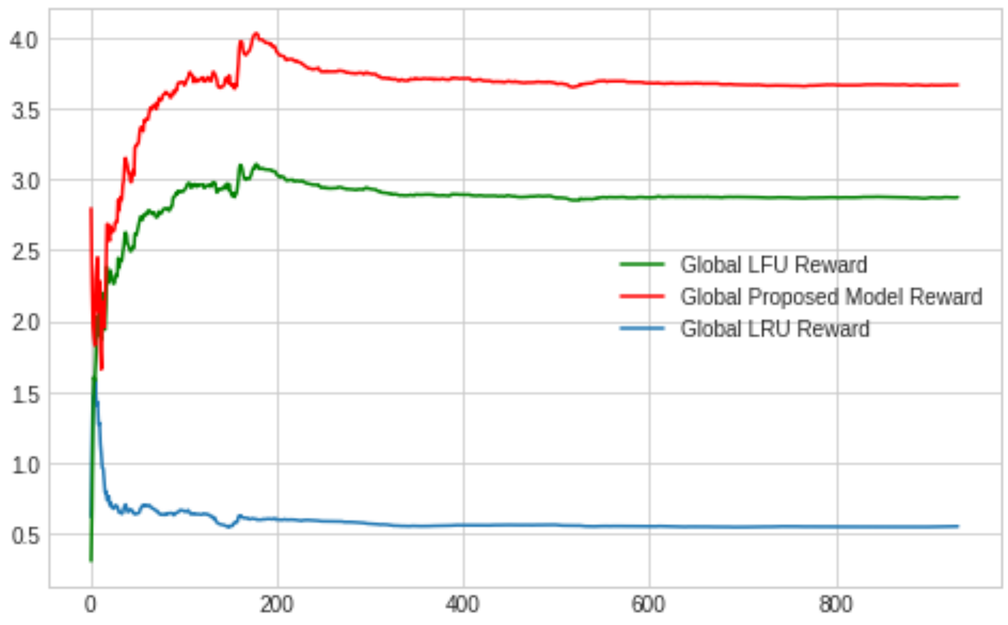


Figure 6.20: Comparing global cache hit ratio for all algorithms in peak hours

Finally, in figure 6.20 it can be seen that our proposed model achieved the highest global hit ratio throughout the simulation. Our algorithm was able to make more efficient caching actions because of the combination of multiple features that the MAAC was able to utilize.

Chapter 7

Conclusion

In this paper, we have researched the different scenarios of content caching at the edge network. We have found researches which have cached contents in the user's device, vehicles, ground base station or road side unit, and also in the sky using UAVs. We have gone through the network, computational, caching and collaborative constraints that exist in the different networks such as in VANET,FANET,D2D,A2G. After going through the works, we have been inspired to come up with solutions that haven't been tackled before in edge computing. For example, we planned to integrate the air and ground networks to create a hierarchy of caching to reduce the content transmission latency and to improve the quality of experience for the users. We have considered computation, caching, communication and control of mobility, to cache. In addition to this, the control of mobility was not previously been used in papers to cache. It is noteworthy to mention that, we have considered a clustering approach to dynamically allocate UAVs based on user mobility where the user position is modelled through the use of Random Waypoint. We have designed a cooperative multi-agent actor-critic based reinforcement learning algorithm, which has been trained and tested, where the edge device is intelligent enough to make caching decisions based on users content request.

However, there are some scopes for further improvement of our work. We have taken into account only the closest UAV and GBS pair since it will provide a better smaller SNR value. If the UAV and GBS cannot serve, then the request is sent to the cloud. This means that we have not considered a Multi GBS-UAV architecture, where there is communication among GBS and UAVs to reach a global cache hit maximization. This is one of the areas where there is future scope of work for researchers.

Bibliography

[1] *Cisco Annual Internet Report - Cisco.* Accessed: Jan. 08, 2021. [Online]. URL: https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html.

[2] *What is Edge Computing?. Author: Michael by Michael Wang - Medium.* Accessed: Jan. 08, 2021. [Online]. URL: https://medium.com/@miccowang/what-is-edge-computing-f997c0ab39fc.

[3] *A Beginner's Guide to Edge Computing — by Velotio Technologies - Velotio Perspectives — Medium.* Accessed: Jan. 08, 2021. [Online]. URL: https://medium.com/velotio-perspectives/a-beginners-guide-to-edge-computing-6cfea853aa11.

[4] Nasser M. Sabah and Aykut M. Hocanin. "Gamma random waypoint mobility model for wireless ad hoc networks". In: *Int. J. Commun. Syst.* 26 (2013), pp. 1433–1445.

[5] H. Zhu et al. ""Proactive caching a framework for performance optimized access control evaluations"". In: *IEEE International Symposium on Policies for Distributed Systems and Networks, POLICY 2009* (2009), pp. 92–94. DOI: 10.1109/POLICY.2009.31.

[6] A. O. A. Salem, T. Alhmiedat, and G. Samara. "Cache Discovery Policies of MANET". In: (2013).

[7] Michael Till Beck et al. "Mobile edge computing: A taxonomy". In: *Proc. of the Sixth International Conference on Advances in Future Internet.* Citeseer. 2014, pp. 48–55.

[8] Y. Mao et al. "A Survey on Mobile Edge Computing: The Communication Perspective". In: *IEEE Communications Surveys Tutorials* 19.4 (2017), pp. 2322–2358. DOI: 10.1109/COMST.2017.2745201.

[9] M. Agiwal, A. Roy, and N. Saxena. "Next generation 5G wireless networks: A comprehensive survey," IEEE Communications Surveys and Tutorials". In: *Institute of Electrical and Electronics Engineers Inc.* 18.3 (2016), pp. 1617–1655. DOI: 10.1109/COMST.2016.2532458.

[10] A. Osseiran et al. "Scenarios for 5G mobile and wireless communications: The vision of the METIS project". In: *IEEE Commun. Mag* 52.5 (2014), pp. 26–35. DOI: 10.1109/MCOM.2014.6815890.

[11] K. Zhang et al. "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks". In: *IEEE Access* 4 (2016), pp. 5896–5907. DOI: 10.1109/ACCESS.2016.2597169.

[12] Anselme Ndikumana and Choong Seon Hong. “Self-driving car meets multi-access edge computing for deep learning-based caching”. In: *2019 International Conference on Information Networking (ICOIN)*. IEEE. 2019, pp. 49–54.

[13] Kyi Thar et al. “Deepmec: Mobile edge caching using deep learning”. In: *IEEE Access* 6 (2018), pp. 78260–78275.

[14] Zhuying Piao et al. “Recent advances of edge cache in radio access networks for internet of things: Techniques, performances, and challenges”. In: *IEEE Internet of Things Journal* 6.1 (2018), pp. 1010–1028.

[15] Xiaoyan Jin et al. “Computation Offloading and Resource Allocation for MEC in C-RAN: A Deep Reinforcement Learning Approach”. In: *2019 IEEE 19th International Conference on Communication Technology (ICCT)*. IEEE. 2019, pp. 902–907.

[16] Chen Zhong, M Cenk Gursoy, and Senem Velipasalar. “A deep reinforcement learning-based framework for content caching”. In: *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2018, pp. 1–6.

[17] Wei Jiang et al. “Multi-agent reinforcement learning based cooperative content caching for mobile edge networks”. In: *IEEE Access* 7 (2019), pp. 61856–61867.

[18] Ying He, Nan Zhao, and Hongxi Yin. “Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach”. In: *IEEE Transactions on Vehicular Technology* 67.1 (2017), pp. 44–55.

[19] Mingzhe Chen et al. “Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience”. In: *IEEE Journal on Selected Areas in Communications* 35.5 (2017), pp. 1046–1061.

[20] Ramy Amer et al. “Caching to the sky: Performance analysis of cache-assisted CoMP for cellular-connected UAVs”. In: *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2019, pp. 1–6.

[21] Yi Zhou et al. “Multi-UAV-aided networks: Aerial-ground cooperative vehicular networking architecture”. In: *ieee vehicular technology magazine* 10.4 (2015), pp. 36–44.

[22] Nan Cheng et al. “Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities”. In: *IEEE Communications Magazine* 56.8 (2018), pp. 26–32.

[23] Chen Zhong, M. Cenk Gursoy, and Senem Velipasalar. “A deep reinforcement learning-based framework for content caching”. In: Mar. 2018, pp. 1–6. DOI: 10.1109/CISS.2018.8362276.

[24] Yuming Zhang et al. “Cooperative Edge Caching: A Multi-Agent Deep Learning Based Approach”. In: *IEEE Access* PP (July 2020), pp. 1–1. DOI: 10.1109/ACCESS.2020.3010329.

[25] *MovieLens 1M Dataset - GroupLens*. Accessed: Jan. 08, 2021. [Online]. URL: https://grouplens.org/datasets/movielens/1m/.

[26] *UAV Navigation in depth: How to measure the quality of the datalink - UAV Navigation*. Accessed: Jan. 08, 2021. [Online]. URL: https://www.uavnavigation.com/company/blog/uav-navigation-depth-how-measure-quality-datalink.