Delivery of 360^o videos in edge caching assisted wireless cellular networks

Pantelis Maniotis

A thesis submitted for the degree of
Doctor of Philosophy

School of Computer Science and Electronic Engineering
University of Essex

October 2020

Abstract

In recent years, 360^o videos have become increasingly popular on commercial social platforms, and are a vital part of emerging Virtual Reality (VR) applications. However, the delivery of 360^o videos requires significant bandwidth resources, which makes streaming of such data on mobile networks challenging. The bandwidth required for delivering 360^o videos can be reduced by exploiting the fact that users are interested in viewing only a part of the video scene, the requested viewport. As different users may request different viewports, some parts of the 360^o scenes may be more popular than others. 360^o video delivery on mobile networks can be facilitated by caching popular content at edge servers, and delivering it from there to the users. However, existing edge caching schemes do not take full potential of the unequal popularity of different parts of a video, which renders them inefficient for caching 360^o videos. Inspired by the above, in this thesis, we investigate how advanced 360^o video coding tools, i.e., encoding into multiple quality layers and tiles, can be utilized to build more efficient wireless edge caching schemes for 360^o videos. The above encoding allows the caching of only the parts of the 360^o videos that are popular in high quality. To understand how edge caching schemes can benefit from 360^o video coding, we compare the caching of 360^o videos encoded into multiple quality layers and tiles with layer-agnostic and tile-agnostic schemes. To cope with the fact that the content popularity distribution may be unknown, we use machine learning techniques, for both Video on Demand (VoD), and live streaming scenarios. From our findings, it is clear that by taking into account the aforementioned 360^o video characteristics leads to an increased performance in terms of the quality of the video delivered to the users, and the usage of the backhaul links.

Keywords: Tile encoding, Virtual Reality, 360^o video, Edge caching, Collaborative caching, Offline caching, Online caching, Live streaming

Acknowledgements

I would like to thank my supervisor, Nikolaos Thomos, for giving me the opportunity to work with him on a very interesting and challenging project. It has been a great honor for me to do research under his supervision. We had an excellent collaboration, which helped me develop my skills as a researcher. Without his insights and guidance, I would not be able to finish my thesis on optimizing caching techniques for the delivery of 360^o videos in cellular networks.

Contents

List of Figures

List of Algorithms

List of Symbols

$\mathcal{N}$	Set of Small Base Stations
$\mathcal{N}_{\mathcal{B}}$	Set of SBSs and MBS
$\mathcal{V}$	Set of 360^o videos
$\mathcal{G}$	Set of GOPs
$\mathcal{L}$	Set of quality layers
$\mathcal{M}$	Set of tiles
$\mathcal{U}$	Set of users
$\mathcal{T}$	Set of time slots
C_n	Cache capacity of the SBS n
o_{vglm}	Size of the tile $vglm$
δ_{vglm}	Distortion reduction from obtaining the tile $vglm$
t_{app}	Playback delay
t_{disp}	Time duration of a GOP

Acronyms

AR Augmented Reality

AVC Advanced Video Coding

CD Caching Decision

CDN Content Delivery Network

CE Caching Entity

CNN Convolutional Neural Network

CQ Cache Query

CR Content Retrieval

CS Cache Storage

D2D Device-to-device

DNN Deep Neural Network

DQN Deep Q-Network

DRL Deep Reinforcement Learning

FD Feature Database

FIFO First In First Out

FoV Field of View

FU Feature Updater

GOP Group of Pictures

HEVC High Efficiency Video Coding

HMD Head Mounted Display

IC Independent Caching

ICNT Independent Caching-No Tiles

IE Information Exchange

ILP Integer Linear Programming

JCL Joint Caching-Multiple Layers

JCNT Joint Caching-No Tiles

LFU Least Frequently Used

LRU Least Recently Used

LSR Last Sample Replication

LSTM Long Short-Term Memory

MAB Multi-armed bandit

MBS Macrocell Base Station

MDP Markov Decision Process

MR Mixed Reality

NN Neural Network

PF Popularity Forecasting

QoE Quality of Experience

RFB Retrieval from Backhaul

RNN Recurrent Neural Network

RTMP Real-Time Messaging Protocol

SBS Small Base Station

SHVC Scalable High efficiency Video Coding

SVC Scalable Video Coding

URF Users Requests Forecasting

URP Users Request Processor

URQ Users Requests Queue

VoD Video on Demand

VP Viewport Prediction

VR Virtual Reality

1

Introduction

1.1 Motivation

Nowadays, we are witnessing an enormous increase in the mobile data traffic of 360^o visual content originating from Virtual Reality (VR) and Augmented Reality (AR) applications. VR/AR mobile data traffic is forecast to grow 12-fold between the years 2017 to 2022 (See Fig. 1.1), resulting in 254 petabytes per month in 2022 [1]. This growth is fuelled by the proliferation of devices that can display VR content, e.g. smartphones, tablets, as well as services that offer users immersive multimedia experience such as online gaming. Also, a plethora of 360^o videos are available for downloading on many social media platforms such as YouTube and Facebook.

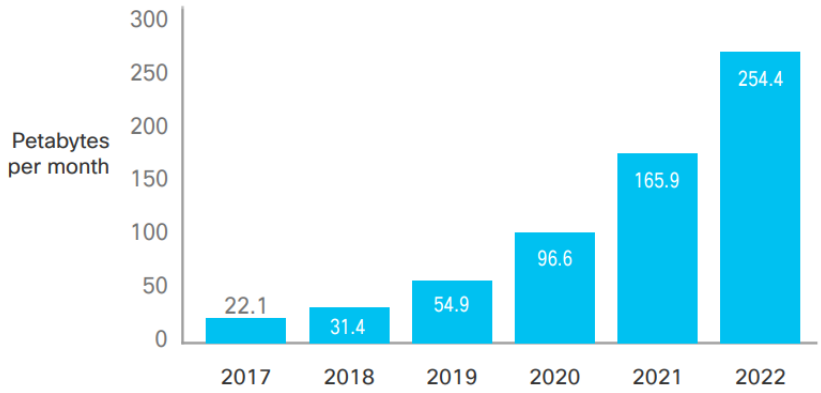

Figure 1.1: VR and AR traffic per month per year. [1]

Capturing of 360^o video can be accomplished using specialized equipment, e.g., omnidirectional camera, rig of multiple cameras, that provides overlapping angles of a video scene. The separate recordings are combined together using a method known as stitching [2]. Stitching of the various recordings is performed either at the camera, or at a computer using specialized software. Afterwards, the stitched 360^o field of view of the scene is mapped to a rectangular 2D representation. This is to allow current compression standards, e.g., H.264/AVC [3], H.265/HEVC [4], AV1 [5], VP9 [6], designed for encoding standard videos to be used for encoding 360^o videos.

Users may watch 360^o videos on various devices, e.g., smartphones, tablets, and PCs. The interaction of the users with the 360^o scenes is achieved by swiping the screen (smartphones, tablets), using the keyboard (PCs), or mouse (PCs). A more immersive experience can be provided with the use of specialized devices, such as Head Mounted Displays (HMDs). Some typical HMDs are the Oculus Rift [7], Samsung Gear [8], and HTC Vive [9]. In HMDs, each 360^o scene is be projected in the internal part of a spherical surface [10], while a user wearing an HMD views only a Field of View (FoV) of the spherical scene, known as viewport.

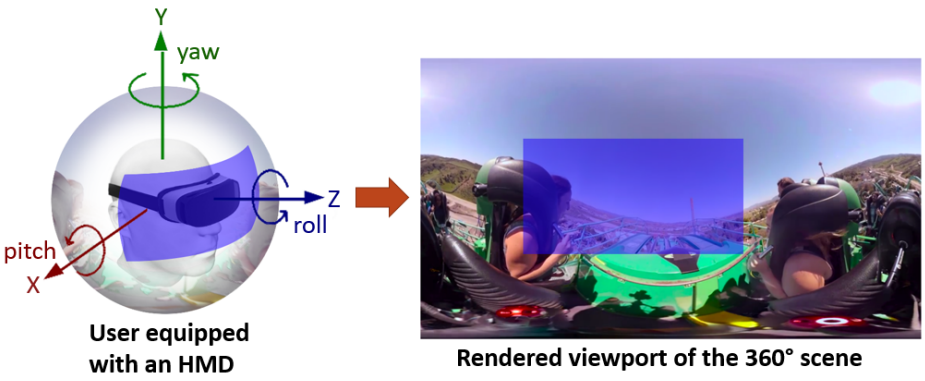

Figure 1.2: User wearing a Head Mounted Display (HMD) to view a 360^o video.

The projected viewport is controlled from the viewer's head movements in the x, y and z axes, as shown in Fig. 1.2. These axes are called pitch, yaw, and roll, respectively [11].

As portable 360^o video cameras are becoming more affordable to the public [12], any attendant of an event (e.g., game, show, conference) is able to capture that event using a 360^o video camera. This changes the way we share our stories, and let us share our experiences in a way that has a memorable effect for our audiences. However, the delivery of 360^o videos in mobile networks remains still challenging. This is because compared to standard videos, 360^o videos are larger in size, as they capture a 360^o FoV of a scene. The technical challenges related to 360^o video delivery can be better understood through the following example. A viewport typically covers 120^o of the overall scene and can have a resolution of up to 4K (3840×2160) [13], which means that the resolution of the whole 360^o video scene can be as high as 12K (11520×6480). Assuming that a 360^o video requires a frame rate of 120 frames per second (fps) to prevent users' dizziness, the network infrastructure should respond to the users' head movements in less than 10 ms. Furthermore, the bit-rate needed to deliver the

360^o video scene in high quality can exceed 100Mbps, and leads to significant bandwidth waste as only a part of the 360^o video is displayed, i.e., the requested viewport. Although the transmission of only the desired viewport could save significant bandwidth resources, state-of-the-art network streaming architectures cannot respond instantly to the users' head movements, due to the end-to-end communication delay between the user and the server where the 360^o video is stored. From the discussion above, it is clear that the delivery of 360^o videos in cellular networks, given the strict delivery deadlines imposed by VR and AR applications is challenging.

1.2 Edge caching

To facilitate the delivery of massive video content in cellular networks, mobile network operators may exploit edge caching [14–16]. In edge caching systems, Small Base Stations (SBSs), e.g., picocells and femtocells, are equipped with caches, which can store a limited amount of popular content files. This is inspired by the fact that only a small number of popular content items accounts for most of the network traffic load [14]. As a result, when there are multiple content requests for a cached content at an SBS, these can be served from the cache directly instead of obtaining the content through the core network using pricey backhaul links. This allows users to receive their requested content with lower latency, and the use of the backhaul links is limited. Depending on whether the content popularity distribution is known at the SBSs in advance, a caching policy can be updated either *offline* or *online*. Furthermore, SBSs may design their caching decisions independently or in a *collaborative* way.

In offline (proactive) caching, the content popularity distribution is consid-

ered for consecutive time periods, e.g., days, weeks, known and fixed [17]. The main assumption behind the design of offline systems is that the content popularity distribution changes smoothly [14], hence it can be accurately approximated by a fixed distribution. Commonly, the content popularity follows the Zipf distribution [18], which is characterized by the number of content files, and a skewness parameter. Specifically, the popularity of each content item is determined according to its rank, i.e., first (most popular), second, etc., and the skewness parameter which defines how concentrated is the distribution in a number of files. Offline caching is completed in two phases: a) the content placement phase, and b) the content delivery phase. In the content placement phase, the caches at the SBSs are populated with popular content during off-peak hours (e.g., night-time). In the content delivery phase, the cached content is delivered to the users when they request it.

The main drawback of offline caching systems is the requirement for the knowledge of the content popularity distribution in advance, which may not be always possible. This is because often in practice the content popularity may change dynamically, e.g., some videos may become viral, which leads to the estimated distribution to not be accurate. This problem has been addressed by online caching (reactive) schemes [19–22]. In online caching, the decision whether to cache a specific content is made after that content has been requested. Some popular content replacement algorithms are the Least Frequently Used (LFU), Least Recently Used (LRU), First In First Out (FIFO), and their variants [23].

Even though cache content placement is often performed independently, i.e., each SBS caches the most popular content, the performance of edge caching systems can be further improved by taking into account users' association [24] with multiple SBSs, and designing the caching decisions in a collaborative way

[25–27]. Collaboration opportunities emerge due to the densification of SBSs in 5G and beyond mobile networks. As a result, users may reside in the coverage area of multiple SBSs. In collaborative caching systems, when a content request arrives at an SBS which does not hold a copy of this content in its cache, this request can be forwarded to another (possibly neighboring) SBS within the user's communication range, instead of being redirected to distant back-end servers through backhaul links. This accelerates the content retrieval and helps to avoid the use of expensive backhaul links.

1.3 Challenges on caching 360^o videos

Despite the existence of edge caching solutions for standard videos [24–29], these systems are suboptimal for caching of 360^o video files. This is because 360^o videos have considerably larger sizes than traditional videos, which limits the number of videos that can be stored at the SBSs caches. Furthermore, edge caching schemes for standard videos cannot exploit the fact that large parts of the video scenes are never displayed, as is the case of 360^o video where users are interested in watching only a viewport. Without explicit consideration of these characteristics of 360^o videos, mobile network operators will struggle to accommodate users' (often diverse) demands for 360^o video content. Clearly, there is a need for novel 360^o video caching schemes which exploit 360^o video features, and do not necessitate the delivery of the entire video scene.

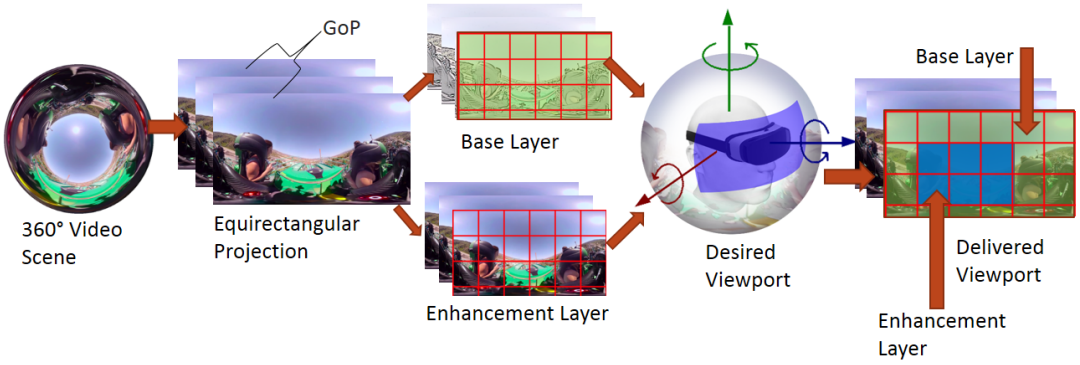

Figure 1.3: Encoding of a 360^o video scene into multiple quality layers and tiles, and delivery of different parts of the scene at different qualities.

1.4 Contributions

1.4.1 Overview

In this dissertation, we investigate how to build efficient edge caching schemes for 360^o videos, considering that users view only some parts of the 360^o video scenes. To this end, we exploit 360^o video coding tools, i.e., encoding in multiple layers and tiles (see Fig. 1.3), in order to optimize caching at the edge servers. With tile-encoding, each tile (rectangular area of the scene) can be encoded independently, which facilitates parallel processing. Furthermore, tile-encoding allows the transmission of different areas of the scene at different qualities. This allows SBSs to cache only the parts of the video content that are popular to the users without requiring caches of higher capacity to achieve the same level of Quality of Experience (QoE) (see Fig. 1.4). Our aim is to maximize the quality of the video delivered to the clients' population, while respecting the delivery deadlines of the 360^o video content. The research efforts of this thesis focused

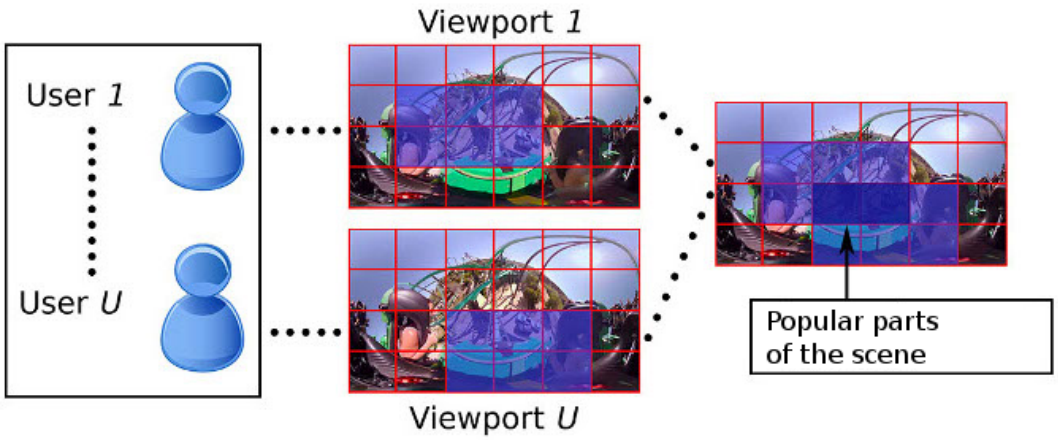

Figure 1.4: Users requesting different viewports that overlap.

on answering the following questions:

- **Research question 1:** How beneficial is the use of advanced coding tools for caching 360^o videos in mobile networks?
- **Research question 2:** Can content-awareness lead to better strategies for caching and delivering 360^o videos?
- **Research question 3:** How complex is the caching of 360^o videos, and how 360^o videos can be cached at the SBSs on a large scale?
- **Research question 4:** When the content popularity distribution is not known in advance, what is the optimal strategy to cache and update 360^o video files at the SBSs?
- **Research question 5:** How to optimize the edge caching placement/eviction policies to support 360^o live video streaming?

Based on our attempt to answer the aforementioned questions, the contributions of this thesis are summarized in the next sections.

1.4.2 Tile-Based Joint Caching and Delivery of 360^o Videos in Heterogeneous Networks

To answer the first two research questions, in Chapter 3, we propose a novel distortion-aware joint caching and delivery framework for 360^o videos that makes cache decisions on a per tile basis. In this work, we assume that the content popularity distribution is fixed and known at the SBSs (offline caching). To the best of our knowledge, this is the first work that considers edge caching for 360^o videos on a per tile basis. We formulate the joint caching and delivery problem as an integer linear programming (ILP) problem, which seeks to maximize the cumulative video distortion reduction among the users' population. To determine the optimal caching and delivery policy we take into account the physical limitations of the network, i.e., bandwidth and latency, as well as the opportunities for collaboration between the SBSs when users reside in the coverage area of more than one SBSs. We show that this problem is NP-hard, answering the first part of the third research question regarding the complexity of the problem of caching 360^o videos. Due to the high complexity of the formulated problem, we decompose it into a number of subproblems on per Group of Pictures (GOP) basis. The solution of each subproblem is obtained using the Lagrangian relaxation method and the subgradient algorithm. The solutions of the decomposed subproblems are combined to obtain the solution of the original problem. We evaluate and compare the performance of our solution with baseline schemes that do not exploit one or more of the following components: (i) tile encoding, (ii) layered encoding, and (iii) SBSs collaboration, to understand the impact of each component on the overall delivered quality of 360^o videos to the users. The results illustrate that exploiting 360^o video coding into multiple quality layers and tiles leads to better

strategies for caching and delivering 360^o videos. This work has been published in a conference paper in IEEE multimedia signal processing workshop 2019 [C1], and a journal paper in IEEE Transactions on Multimedia [J1].

1.4.3 Viewport-Aware Deep Reinforcement Learning Approach for 360^o Video Caching

Motivated by the findings in Chapter 3, where the benefits of caching 360^o videos on per tile basis instead of entire 360^o videos are shown, in Chapter 4, we seek to answer the fourth research question. To this aim, we propose a novel reactive caching scheme which does not require a priori knowledge of 360^o video and tiles popularity distributions (online caching). To the best of our knowledge, this is the first online caching scheme for 360^o videos in the literature. Our method aims at maximizing the overall quality of the 360^o videos delivered to the users, considering a limited history of users' requests. Compared to our work in Chapter 3, which was shown to be NP-hard, this solution scales well with the number of users and 360^o videos, and can be used for caching 360^o videos on a large scale, answering the second part of the third research question. In order to determine which 360^o videos and tiles should be cached at the SBSs, we formulate the problem of 360^o video online caching as a Markov Decision Process (MDP). To reduce the dimensionality of our problem, we introduce the concept of virtual viewport, as shown in Fig. 1.5. A virtual viewport has the same number of tiles with regular viewports, but it consists of the most popular tiles. Specifically, a virtual viewport is shaped from the overlap between the various requests for different viewports, and it is considered to be the most popular viewport of a scene, among the users' population. Virtual viewports differ from the original

ones in that the tiles that comprise them are not necessarily adjacent to each other, i.e., they do not form a rectangular area. When a user requests a viewport of a 360^o video in a certain quality, then if some tiles of the requested viewport also belong to the virtual viewport that is cached at the SBS that received the request, these tiles will be served from that cache. As a result, storing virtual viewports instead of regular ones is expected to lead to an increase in the cache hit ratio, due to the greater flexibility virtual viewports provide in terms of which tiles to cache in high quality. In order to solve large instances of the formulated MDP problem, we use a Deep-Q-Network (DQN). We evaluate the performance of our solution for both real and synthetic 360^o video navigation patterns, and compare its performance with that of known schemes such as the LFU, LRU, and FIFO algorithms. The results illustrate the advantages of the proposed method compared to its counterparts in terms of the overall quality users enjoy, the overall cache hit ratio, and the cost of delivering the requested content to the users. In addition, it is shown that the overlap between the various requests for different viewports shapes the popularity of each tile, and defines the virtual viewports. This work has been submitted for publication in the IEEE Transactions on Multimedia journal, and at the moment it is under revision. A preprint of this work is available in [P2].

1.4.4 A Tile-based caching framework for 360^o live video streaming

Considering that the content popularity distribution may be unknown at the SBSs, in Chapter 4, we examined the caching of 360^o videos for a Video on Demand (VoD) solution. However, this scheme cannot be trivially used for

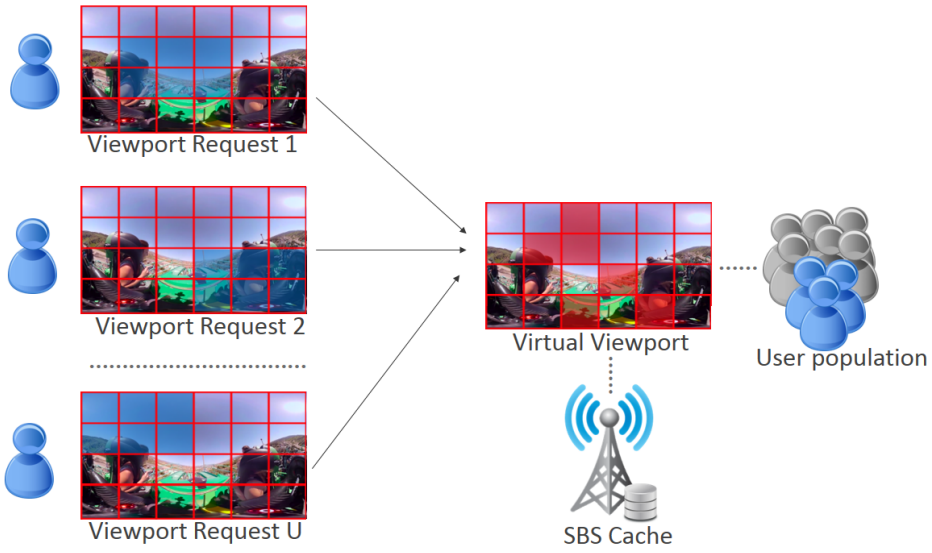

Figure 1.5: Virtual viewport, shaped from the overlap between the user requests for various viewports.

live streaming of 360^o videos where multiple simultaneous requests for the same video may occur. Inspired by the above, in Chapter 5, we propose a novel caching framework for live streaming 360^o videos, aiming to answer the fifth research question. To the best of our knowledge, this is the first caching scheme for 360^o videos that is applicable to live streaming scenarios. Our system aims at maximizing the overall delivered quality and reduce the backhaul usage, by optimizing the cache hit ratio. To achieve this, our algorithm uses observations of users' requests for the various 360^o videos and viewports in order to decide about the optimal cache eviction/placement strategy. Differently from Chapter 4 where we used virtual viewports, in this chapter, when our system decides to cache a 360^o video, it caches all the tiles of that video encoded in base quality to ensure interactivity, and a number of tiles in high quality to enhance the quality of the delivered video quality. The latter decision is based on the popularity of a video, i.e., more tiles in high quality are cached for the most popular videos,

as well as the tiles' popularity, which determines which tiles are most likely to be requested. To determine which 360^o videos and tiles should be cached at the SBSs, we use Long Short-Term Memory (LSTM) networks. This allows the continuous popularity forecast of the 360^o videos and tiles for the next GOP, which in turn is used for the prefetching of them at the SBSs. To enhance the quality of the rendered viewports and reduce the backhaul usage, we allow users who are located in the overlap of the coverage areas of multiple SBSs to receive data from any of these SBSs, at different communication delays. We evaluate the performance of our solution for both real and synthetic 360^o video navigation patterns, and compare it with that of the LFU, LRU, and FIFO algorithms. The results demonstrate the superior performance coming from the use of LSTMs to decide which 360^o videos and tiles to cache in terms of the overall cache hit ratio, the quality of the rendered viewports and the backhaul usage. This work resulted in a conference paper published in IEEE multimedia signal processing workshop 2020 [C2]. We aim at submitting this work for publication in the IEEE Transactions on Multimedia journal.

1.5 Thesis Outline

The remainder of this thesis is organized as follows. In Chapter 2, we overview works related to edge caching, and 360^o videos. Afterwards, in Chapter 3, we present the proposed framework for the joint caching and delivery of 360^o videos in wireless cellular networks. Next, in Chapter 4 we introduce the proposed reactive caching scheme for 360^o videos, which does not require knowledge of the 360^o video and tile popularity distributions. Then, in Chapter 5 we present the proposed caching framework for live streaming of 360^o videos, where video

and tile popularity distributions are non-stationary. Finally, in Chapter 6, we outline the conclusions drawn in this thesis, the main contributions of our work, and some potential future research directions.

2

Related Work

2.1 Introduction

The purpose of this chapter is to review the literature related to this thesis. This chapter is organized as follows. In Section 2.2, we discuss works on various research areas related to 360^o videos, e.g., tile encoding of 360^o videos, visual attention in 360^o videos, 360^o video projection schemes, etc. Next, in Section 2.3, we overview related works to edge caching, e.g., online caching, collaborative caching, etc, and discuss their advantages and disadvantages.

2.2 360^o video related literature

2.2.1 Tile encoding of 360^o videos

Recently, 360^o videos have become increasingly popular on commercial social platforms such as YouTube and Facebook. 360^o videos encode a 360^o field of view of a scene, offering users an immersive experience. Users interact through the scenes and view only a portion of the encoded 360^o video frames, which is known as viewport (see Fig. 1.2). However, this interactivity comes at the cost of very high bandwidth requirements, as the bitrate required to provide a high QoE to the users is much higher than that needed for regular videos. Tile-encoding has been investigated in [30–32], as a way to cope with the high bandwidth requirements in 360^o video streaming. With tile-encoding, each 360^o video scene is encoded into independent segments, the tiles. Furthermore, each tile is further encoded into multiple quality layers, which gradually increase the quality of each tile. In this way, tile-encoding allows the systems in [30–32] to save significant bandwidth resources by taking into account that users are interested in viewing only a viewport of the 360^o video scene, which removes the need to deliver the whole scene in high quality. The selection of the tiling scheme is investigated in [33], along with its impact on the compression and the resulting streaming bitrate. Specifically, each 360^o video is encoded in two versions with different resolutions, while each version is further encoded into tiles. Similarly to [33], in [34] authors investigate the performance of tile-based video streaming in 4G networks with respect to the coding efficiency and the bandwidth savings. In [11], a rate-adaptation scheme is presented, aiming to determine the quality level of each tile by leveraging the predicted available bandwidth in the future

and the distribution of users' head positions. Differently from [11], authors in [35] propose an HTTP adaptive streaming system (HAS) that seeks the optimal rate at which each tile should be downloaded as users navigate through the 360^o video scenes. Differently from [35], an algorithm that provides the streaming rate each tile should be downloaded is proposed in [36], to guarantee a QoE that degrades with low probability when the viewport prediction error is high. This algorithm outperforms its counterparts by 50% in terms of QoE.

2.2.2 Projection schemes

Although current video compression standards, e.g., H.264, HEVC, are efficiently used for encoding of standard videos, they have not been designed to be directly used for the compression of the spherical surface of the 360^o videos. To overcome this limitation, the spherical surface of the 360^o video scenes is initially projected into a rectangular 2D surface. Then, this surface is encoded in the same way to that of standard videos. Some of the most common projection schemes are the Equirectangular (ERP) [37, 38] projection used by YouTube, and the Cubemap (CMP) [37, 39] projection used by Facebook. The evaluation of a number of different projection schemes, i.e., Rhombic Pyramid, Square Pyramid, Truncated Pyramid, along with the multi-resolution version of the ERP, and CMP schemes is investigated in [40], in terms of the achieved rate-distortion performance. From the results, it is noted that the multi-resolution version of the ERP and CMP schemes showed the best performance in terms of compression gains. A hybrid projection scheme is presented in [41] based on the ERP and the Transverse Cylindrical Projection (TCP) scheme. This hybrid scheme improves the distortion of the ERP scheme by combining the advantages of the ERP around the equator, and of the TCP near the poles.

2.2.3 Visual attention in 360^o videos

Understanding visual attention of users when they are viewing 360^o videos is crucial for improving users' QoE [42, 43]. This is because as some regions of a scene are more popular than others, this understanding can be exploited to optimize the encoding and transmission of 360^o videos [44]. To this aim, many works [45–49] focus on creating datasets which contain information of the users' behaviour when viewing 360^o videos. Such an example is the dataset in [45], which consists of 10 videos viewed by 50 users aged between 20 and 48. This dataset provides image saliency maps [50–52] that highlight the regions of the scenes where there is the most attention, motion maps [53] that highlight moving objects, and sensor data from the HMDs that help to obtain the users head orientation. Similarly to [45], a dataset of 48 users viewing 20 different 360^o videos on HTC Vive is presented in [48]. Apart from the user head movements, users in [48] were asked to fill a questionnaire regarding the motion sickness experienced during the viewing of the 360^o videos. Differently from [45, 48], a dataset containing view trajectories from users viewing 360^o videos on computer screens is presented in [46]. By analyzing the trajectories, it is found that when 360^o videos follow a storyline, the trajectories are bounded within a small dynamic range. On the contrary, the trajectories are more dynamic for 360^o videos such as virtual tours. In contrast to [45–49], authors in [54] investigate how to identify clusters of users that watch the same parts of the 360^o videos, by analyzing datasets that contain users' navigation patterns. From the results, it is shown that their method leads to clusters where users watch at least 85% of the same content.

2.2.4 Viewport Prediction

In 360^o video streaming, the accurate prediction of the viewports that will be requested by the users in the future is essential for their timely delivery [55], and the ability of the users to enjoy a high QoE [56]. This is because as 360^o videos are associated with high bandwidth requirements, servers may transmit only the viewports that will be watched by the users in high quality. However, communication networks are not able to respond instantly to the users' head movements, hence, predicting the viewports that will be seen from the users allows the delivery of them prior to their demand [57, 58] and facilitates 360^o video playback. To predict the viewports that will be requested by the users, authors in [59] proposed a reinforcement learning approach based on contextual bandits. Differently from [59], authors in [60] presented a deep learning network that uses a Convolutional Neural Network (CNN) to extract the content-related features from the 360^o videos, and a Long Short-Term Memory (LSTM) network to predict the users' head movements. The results show that a 16.1% improvement can be achieved in terms of the prediction accuracy compared to a baseline approach that takes into account only the position data. Similarly to [60], the use of CNN for viewport prediction is proposed in [61] for a live streaming setup. In [62], viewport prediction is performed using Linear Regression (LR). Differently from [62], in [63], LR is initially used to predict a motion-based fixation, and then, cross-users viewing fixations are exploited by the K-Nearest Neighbors (KNN) algorithm to improve the prediction accuracy of LR. The prediction of the viewports for a long time horizon is examined in [64]. The evolution of the viewports through the 360^o scenes is modeled with trajectories in the roll, pitch and yaw angles. From the results, it is concluded that for a horizon of up to 10

seconds, the algorithm in [64] can increase the accuracy of the predicted viewport area by 13%.

2.2.5 360^o live video streaming

The use of tile encoding for 360^o live video streaming has been examined in [65], in order to deliver only the parts of the requested FoV in high quality. To this aim, an architecture that combines the Real-time Transport Protocol (RTP) and DASH is proposed, where authors examine the trade-off between video quality and bandwidth usage. The results show that 50% of bandwidth savings can be achieved. Differently from [65], authors in [66] propose a multicasting architecture for 360^o live video streaming. From the evaluation, it is shown that a quality gain of 3 dB is achieved for users with low bandwidth capacities and a quality gain of 3.5-4 dB for users with high bandwidth capacities. A generic measurement system is presented in [67] for the collection of key performance statistics, e.g., video quality change, rebuffering events, for evaluating the performance of live streaming 360^o videos on platforms such as YouTube and Facebook. To guarantee a high QoE, a live streaming system for 360^o videos is proposed in [68], which is based on MPEG media transport (MMT).

2.2.6 Quality of Experience Metrics

A key element in designing a 360^o video delivery system is the assessment of the perceived quality by the users [69]. The quality metrics used to assess the perceived quality can be classified in two categories, namely subjective, and objective [70]. In subjective quality metrics, a number of users (subjects) watch various 360^o videos and rate their perceived quality. Although in such an ap-

proach users are directly involved in the assessment of the QoE they enjoy, this approach is expensive and time consuming [71]. To overcome these limitations, objective quality metrics attempt to automate the assessment of the 360^o videos' quality through the modeling of the Human Visual System (HVS) [72]. A subjective assessment analysis is presented in [73], where users rate their perceived QoE in terms of perceptual quality, presence, acceptability, and cyber-sickness. In particular, perceptual quality and the acceptability are measured when users watch a 360^o video both with or without a Head-Mounted Display (HMD). Differently from [73], an objective quality metric based on Structural Similarity Index (SSIM) is proposed in [74]. The performance of this metric is verified using subjective quality assessments, while its usage is not dependent from the 360^o video projection method. Similarly, in [75], a number of objective quality metrics are explored for quantifying the quality of 360^o video encoding. These include PSNR, weighted to spherically uniform PSNR (WS-PSNR), spherical PSNR without interpolation (S-PSNR-NN), spherical PSNR with interpolation (S-PSNR-I), and PSNR in Crasters Parabolic Projection (CPP-PSNR). From the evaluation in [75], it is concluded that traditional PSNR is the most appropriate quality metric for measuring end-to-end distortion because of its low complexity.

2.3 Edge caching systems

2.3.1 Overview

We are currently witnessing an unprecedented growth in mobile data traffic [1]. To cope with this increase, the emerging 5G mobile networks are designed to

exploit better their spatial resources [14]. However, in order for this approach to be effective, a high-speed backhaul connectivity is required for each SBS, which is often not possible. To overcome the above limitation, edge caching emerged as a way to offload the traffic to the backhaul links. The benefits of edge caching are attributed to the fact that only a small portion of the available content accounts for most of the traffic load [76]. Edge caching allows users to experience services with less latency [77], improve the QoE they enjoy [78–80], and reduce the network operating cost [81]. This can be beneficial for applications with high bandwidth requirements, e.g., 360^o videos, which motivated us to study in Chapters 3, 4, and 5, how edge caching can be used with tile-encoding, to facilitate 360^o video streaming. The problem of using edge caching to facilitate the delivery of standard videos has been widely studied in the literature [77, 78, 82, 83]. For example, authors in [78] studied the delivery of standard videos adaptive video streaming, where videos are available in multiple representations of different qualities. The aim is to maximize the aggregate distortion reduction among the users' population, while the additional cost of downloading the representations is minimized. Differently, the caching placement for layered video content on edge caching systems is investigated in [82], where the decisions about which video layers to cache in each SBS are made by taking into account the caching cost, the available cache capacity at the SBSs, and the various social traits of the mobile users. In [77], authors examine how to assure the delivery of high video resolution, i.e., 4K, in LTE-A networks. Their work showed that the proposed live streaming system is able to sustain 4K video quality at a global scale. The impact of the content request patterns on video streaming with edge caching is studied in [83], where the video request patterns and users' behaviors are analyzed. Based on the users' request analysis, authors design a caching scheme

that outperforms the LRU/LFU schemes in terms of the cache hit ratio by 30%.

2.3.2 Online edge caching

The main challenge in edge caching systems is the requirement to know the video popularity profiles in advance, which in many cases is not possible. This is because the content popularity distribution may not be static, but evolve dynamically as new content is continuously added [84] or removed because it becomes obsolete. To this aim, the use of reinforcement learning (RL) algorithms that predict the content popularity based on the demand history have been proposed. Specifically, the use of Multi-Armed Bandit (MAB) algorithms [85] has been investigated in [86–90], where the content popularity is predicted from the observations of users' requests. For example, SBSs in [86] learn the content popularity online, while considering the switching costs associated with the addition of new files to the cache. In the same manner, three different MAB algorithms that lead to different exploration-exploitation trade-offs are studied in [87]. Differently from [86, 87], the use of armed-bandits that exploit contextual information, e.g., age, sex, is proposed in [88, 89], in order to predict the popularity of files. The content cache placement is jointly designed using an MAB framework in [90], considering that SBSs may share their cached content via wired links, and content retrieval comes at a cost, i.e., transmission energy. A reinforcement learning framework for finding the optimal caching policy is presented in [91], where the users' requests are modeled as Markov Decision Processes (MDPs). The optimal caching placement is found using the Q-learning [92] algorithm. A trend-aware caching scheme is presented in [93], where the popularity trend of each video is learnt online to determine the cache policy. From the results, it is shown that the proposed method can achieve a 40% improvement in terms of

cache hit ratio. Differently from our approach, which will be presented in Chapter 4, where we study the problem of online caching for 360^o videos using neural networks (NN), i.e., DQN, the use of NN for the problem of online caching for standard videos is investigated in [94, 95]. Specifically, a Deep Reinforcement Learning-based framework is presented in [94], which aims at maximizing the long-term cache hit ratio. To solve the formulated problem in [94], an Actor-Critic algorithm [96] based on the Wolpertinger architecture [97] is proposed. Differently from [94], an Actor-Critic algorithm is presented in [95] where the actor uses the Gibbs distribution [98] and the critic uses a deep neural network with the goal to minimize the average transmission delay. To do so, the user scheduling and content caching are jointly addressed. The main difference between DQN and Actor-Critic is that in the former case, the employed DRL framework consists of only one NN that decides which action should be taken. On the contrary, in the latter case, the employed framework is comprised of two NNs. The first NN (Actor), is used to decide given a state, which action should be taken. The second NN (Critic), is used to provide feedback on how good was the selected action from the Actor. The benefit of using Actor-Critic frameworks is that they tend to work better for extremely large or continuous action spaces. On the contrary, DQN frameworks as our approach followed in Chapter 4, have the benefit of being less complicated.

2.3.3 Collaborative edge caching

To enhance the performance of edge caching in cellular networks, network operators may exploit the collaboration opportunities that emerge because of the overlap between the coverage areas of the SBSs [25–27], in 5G and beyond networks. In this way, when a content request is not found in the cache of the

SBS where the request was issued, it can be served to the user from the cache of a neighboring SBS, provided the requested content is cached in it. Hence, the requests may not be redirected to distant back-end servers, which reduces the usage of the backhaul links. As a result, users experience services with less latency, and their perceived QoE is increased. The joint caching placement and users' association is studied in [24], while taking into account the backhaul delay and the wireless channel quality. The aim in [24] is to minimize the average download delay. From the simulation results, it is shown that jointly designing the caching decisions and the users' association with the SBSs can significantly reduce the average download delay. Similarly to [24], the joint design of cache placement and user association policies is examined in [99], aiming to optimize the trade-off between load balancing and backhaul saving. The benefits of collaborative edge caching have been also studied in [100, 101], where it is shown that the overall cache hit ratio is increased when SBSs decide collaboratively which video files to cache. In [100], collaborative edge caching is investigated from a network economics point of view. This work explores how to maximize the profit earned when a video request is served by an SBS in two cases: a) when the video file is entirely cached in only a single cache, and b) when the video is encoded by means of network coding [102] and the resulting coded data is split into a number of segments that can be stored separately in various SBSs. The impact of collaborative caching for video streaming systems in mobile networks is studied in [101]. From the evaluation, it becomes obvious that collaboration between caches enhances the performance of the overall edge caching system in terms of both the cache hit ratio and the QoE perceived by the users.

2.3.4 Edge caching of layered videos

Edge caching systems can be further improved and can offer higher QoE to the end users by exploiting video encoding into multiple quality layers. Layered videos can be generated by scalable video coders such as H.264/SVC [103], HEVC/SHVC [104], among others. The flexibility in caching decisions coming from encoding in multiple layers is exploited in [28, 29], where network operators can collaboratively optimize which video layers to cache. To solve this problem, SBSs caches are split into two parts: a part allocated for contents requested by users of the operator who owns the cache, and a second part which is given for caching video layers of content requested by users who belong to other operators. From the results, it is observed that an up to 25% delay gains can be achieved, compared to layer-agnostic caching schemes. The exploitation of video encoding into multiple quality layers for Dynamic Adaptive Streaming over HTTP (DASH) is studied in [105], where SBSs cache the base layer along with some of the enhancement layers, depending on the encountered network conditions. It is shown that the cache hit ratio of commonly used algorithms such as the LRU and LFU can be improved by advancing their policies so that they decide either the eviction of a content, or its "trimming" where only some of the stored layers are evicted.

3

Tile-Based Joint Caching and Delivery of 360^o Videos in Heterogeneous Networks

3.1 Introduction

In this chapter, we investigate the problem of collaborative joint caching and delivery of 360^o videos in a video on demand setting. The proposed scheme takes advantage of encoding of 360^o videos in multiple tiles and layers, in order to maximize the cumulative video distortion reduction. Tile encoding allows to make fine-grained decisions regarding which tiles to cache in each SBS, and from

where to deliver them to the end users. To ensure continuous video playback, we explicitly consider the time delivery constraints. Due to the high computational complexity of the studied optimization problem, we split it into a set of subproblems. Each subproblem seeks for the optimal caching and delivery policy on per GOP basis. The solution of each subproblem is obtained using the Lagrangian relaxation and the subgradient algorithm. We evaluate and compare the performance of our solution with tile-agnostic and layer-agnostic schemes, in order to understand the impact of each component on the overall deliverd quality to the users. The results make apparent the benefits coming from designing the caching and delivery decisions on a per tile basis, and the importance of exploiting SBSs collaboration.

Chapter 3 is organized as follows: In Section 3.2, we describe our system setup. Afterwards, we provide the problem formulation in Section 3.3. In Section 3.4, we propose a reduced complexity algorithm to solve our optimization problem. In Section 3.5, we extensively evaluate the performance of the proposed scheme and compare it with other methods. Finally, we draw conclusions in Section 3.6.

3.2 System Setup

In this section, we examine the joint design of the caching and delivery of 360^o videos. We consider that the caching occurs at the SBSs on a per-tile basis. We further make use of SBSs collaboration to serve most of the requests from local caches. In the following, we first introduce the system model and explain the various parts of the considered network architecture. Then, we provide an analytical example that reveals the benefits of collaborative caching for tile-

encoded 360^o videos.

3.2.1 Network

We consider a mobile network consisting of a set of N Small-cell Base Stations, as shown in Fig. 3.1. Let $\mathcal{N} = \{1, \dots n, \dots, N\}$ denote the set of SBSs indices. We also assume that the SBSs can communicate with a Macro-cell Base Station (MBS) indexed by $N + 1$, through which they can retrieve the requested content (360^o video files) from distant servers via backhaul links. For notational convenience, we define the set $\mathcal{N}_{\mathcal{B}} = \mathcal{N} \cup \{N + 1\}$ which includes the indices of all the SBSs and the MBS.

In the considered network, there are $U = |\mathcal{U}|$ users, where $\mathcal{U} = \{1, \dots, U\}$ is the set of user indices. Each user is associated with one or more SBSs. The association of the users with the SBSs depends on the transmission range of each SBS. The values of the transmission range of the SBSs form the set $\mathcal{P} = \{p_1, \dots p_n, \dots, p_N\}$, where p_n is the communication radius of the nth SBS. The transmission range of the MBS is denoted by p_{N+1}, and is considered to be large enough so that each SBS can establish a connection with the MBS. The overlap in the coverage areas of the SBSs allows users to access multiple SBSs. We introduce the binary variable $\alpha_{nu} \in \{0, 1\}$ that equals to 1 when user u resides in the coverage area of the SBS n, and 0 otherwise.

Each SBS is equipped with a cache with capacity $C_n \geq 0,\ \forall n \in \mathcal{N}$, where content files can be cached. A user's request can be satisfied by any of the associated SBSs, if the requested 360^o video file is stored in their cache. Differently, when the file is not cached in any of the SBSs associated with the user, the requested content is fetched from a remote content server through the backhaul link with the help of the MBS.

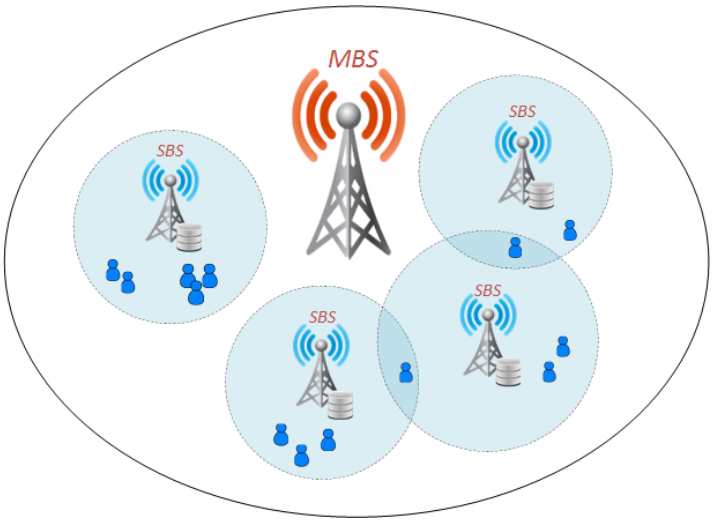

Figure 3.1: Mobile network architecture consisting of multiple SBSs and a single MBS. Due to dense placement of SBSs, users can reside in the coverage area of multiple SBSs.

Our scheme is an offline edge caching scheme, where content retrieval is carried out in two phases, namely content placement, and content delivery. In the content placement phase, content is fetched during off-peak hours from distant back-end servers to the caches of the SBSs. In the delivery phase, the cached content is delivered to the users by either the SBSs or through the backhaul link via the MBS, according to the users requests. If the requested content is not cached in the local SBSs, it first has to be fetched via the backhaul, and then it can be delivered to the user. Offline edge caching can help to avoid network congestion during peak hours, as user requests are diverted from remote content servers to local edge caches (SBSs).

3.2.2 Video Library

We assume that the video content catalogue contains $V = |\mathcal{V}|$ files, where $\mathcal{V} = \{1, \dots v, \dots, V\}$ represents the set of indices of the 360^o videos in the catalogue.

All the video files are stored at back-end content servers, while SBSs cache only part of the available content catalogue. The caching decisions are taken offline and the files to be cached are moved to the SBSs caches during the content placement phase.

Each 360^o video file is encoded in a set of GOPs $\mathcal{G}$, which in turn are each encoded in a number of M independently coded tiles. Let the set $\mathcal{M} = \{1, \ldots, M\}$ denote the M independently encoded tiles of a 360^o video. The use of independently encoded tiles is motivated by the fact that users are interested in watching different viewports of the demanded videos. That is, while users may request different video files, they may also be interested in watching different parts of a video scene, i.e., different viewports. As a result, the popularity of tiles depends on both video and viewport popularity. In particular, the overlap between the various viewports shapes the popularity of each tile. Thus, not only the popularity across the viewports of the same video may be different, but also the tiles within the same viewport may have different popularity due to overlap among requested viewports. This in turn affects the optimal caching and routing decisions.

Each tile is further encoded into a set of quality layers $\mathcal{L}$. The most important layer is called base layer. When the base layer is received by a user, it offers a reconstruction of a tile at the lowest available quality. The next layers are known as enhancement layers and contain information that can improve the reconstruction quality of each tile. However, in order to reconstruct a tile at the quality that corresponds to an enhancement layer, all previous enhancement layers including the base layer must be available to the user. Encoding of the 360^o video files in layers and tiles offers greater flexibility in deciding which data should be stored in the SBSs caches. Thus, when some viewports are more

popular than the others or/and there is significant overlap between some of the viewports, the tiles that form these viewports/overlap regions can be cached at higher quality at the SBSs, if there is sufficient space, while the rest of the tiles can be cached in lower quality. For more information on video coding, we advise the interested reader to refer to *Appendix A*.

When a user requests a viewport of a 360^o video file, this request translates into requests for all the tiles of the frame at base layer and all the tiles of the requested viewport at the highest available quality. As we mentioned in Chapter 1, while the delivery of only the desired viewport would save bandwidth resources, it is not, in general, an optimal strategy. This is due to the fact that, as users navigate through the scene, the actual video consumption pattern may deviate from the expected requests used to determine the joint caching and routing policy. If only the requested viewport was delivered, any deviation in viewing pattern would require a re-transmission of the correct viewport, which in turn would lead to large switching delays and bandwidth waste [35]. By delivering the entire frame at base layer we accommodate for any deviations of the actual viewing pattern of the users from the user requests. Hence, rapid degradation of the perceived QoE is avoided.

3.2.3 Motivating Example

To show the benefits of the proposed approach let us consider the network setting illustrated in Fig. 3.2, where there are two SBSs, SBS_1 and SBS_2, and five users $u_1, u_2, \ldots, u_5$. The coverage area of SBS_i, $i \in \{1, 2\}$ is shown by a circle. Each user demands only one video file from a content library consisting of two 360^o videos v_1 and v_2 with v_1 being more popular than v_2. For the sake of simplicity, we assume that the video files are of equal size. User requests for v_1

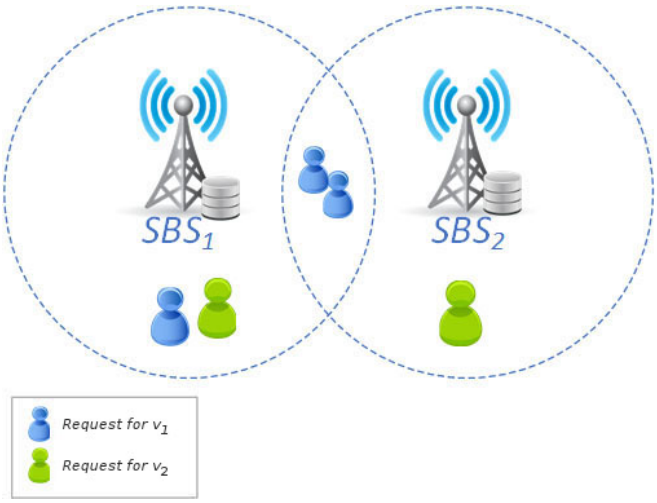

Figure 3.2: Motivating example.

and v_2 are depicted with blue and green color, respectively. We further assume that each SBS has a cache with capacity $C_i,\ i \in \{1, 2\}$ sufficient to store only one of the 360^o videos. A naïve caching policy is to cache the most popular content at each SBS. According to this policy, both SBS_1 and SBS_2 would cache video v_1 and three users' requests would be satisfied from content cached in the SBSs. However, this caching policy is suboptimal, as it disregards SBSs collaborative caching opportunities emerging since users may be associated with more than one SBSs. Considering the above, the optimal caching policy would be to cache video v_1 at SBS_1 and video v_2 at SBS_2. This would result in four users requests being satisfied from content cached in the SBSs compared to three when SBSs collaboration is overlooked. From this example, it is obvious that collaborative caching helps to increase the number of the 360^o videos that will be delivered locally (from the SBSs) to the users and leads to improved overall users satisfaction.

Differently from the previous network setting, let us now consider that the cache capacity of each SBS is less than that needed to store an entire 360^o

video. If each 360^o video file is encoded in a single file, the SBSs will not cache any video file, no user request will be satisfied from the SBSs and the video files should be retrieved through backhaul links. However, when encoding of 360^o video in tiles is enabled, then a number of the tiles may be cached at the SBSs if they have a capacity sufficient to store a single tile. This would lead to some user requests being partially satisfied and the load of the backhaul link being decreased. Considering that users are interested in watching a viewport of each 360^o video, the encoding of the video in tiles would possibly allow caching the most important tiles at the SBSs, which could potentially lead to smaller delays and higher QoE. Following the same line of arguments, the advantages of encoding 360^o video in multiple layers can be shown. From the above, we can conclude that the encoding of 360^o video files in multiple layers and tiles allows to make more fine-grained caching decisions.

From the above discussion, it is clear that by exploiting both the association of some users with more than one SBSs and 360^o video encoding into multiple quality layers and tiles, the QoE perceived by the users would increase.

3.3 Problem formulation

In this chapter, we aim at finding the optimal joint caching and delivery policy for tile-encoded 360^o layered videos that maximizes the cumulative distortion reduction experienced by the users. To determine the optimal policy we take into account network formation, SBSs capabilities, channel characteristics, users requests, video-specific limitations arising from the encoding of 360^o video in tiles and layers, and the requirement for smooth playback.

Let us introduce the binary variable $y^{nu}_{vglm} \in \{0, 1\}$, where $y^{nu}_{vglm} = 1$, if the

tile $m \in \mathcal{M}$ of the layer $l \in \mathcal{L}$ that belongs to GOP $g \in \mathcal{G}$ of the video $v \in \mathcal{V}$ will be delivered directly from the SBS $n \in \mathcal{N}$ to the user $u \in \mathcal{U}$, and $y_{vglm}^{nu} = 0$ otherwise. Similarly, let $y_{vglm}^{(N+1)u} \in \{0,1\}$ be equal to 1 when a tile is fetched from the backhaul through the MBS, and 0 otherwise. Therefore, the routing decisions in our system can be described by the vector[1]:

$$\mathbf{y} = (y_{vglm}^{nu} \in \{0,1\} : n \in \mathcal{N}_{\mathcal{B}}, u \in \mathcal{U}, v \in \mathcal{V}, g \in \mathcal{G}, l \in \mathcal{L}, m \in \mathcal{M}) \tag{3.1}$$

We note that requests for tiles are *unsplittable*. This means that each tile request is entirely satisfied either by only one SBS or it has to be fetched through the backhaul.

Now, let us define the binary decision variable $x_{vglm}^{n} \in \{0,1\}$, which takes the value 1 when the tile $m \in \mathcal{M}$ of the layer $l \in \mathcal{L}$ that belongs to GOP $g \in \mathcal{G}$ of the video $v \in \mathcal{V}$ is cached at the nth SBS, and 0 otherwise. Hence, the caching decisions for the entire system are described by the vector:

$$\mathbf{x} = (x_{vglm}^{n} \in \{0,1\} : n \in \mathcal{N}, v \in \mathcal{V}, g \in \mathcal{G}, l \in \mathcal{L}, m \in \mathcal{M}) \tag{3.2}$$

Recall that SBSs have limited cache capacity, i.e., only enough only to store a number of tiles from the tile-encoded 360^o layered videos. If o_{vglm} denotes the size of a tile (in *Mbits*), we have:

$$\sum_{v \in \mathcal{V}} \sum_{g \in \mathcal{G}} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} o_{vglm} x_{vglm}^{n} \leq C_n, \forall n \in \mathcal{N} \tag{3.3}$$

where C_n the cache capacity (in *Mbits*) of the nth SBS. Eq. (3.3) is the cache

[1]Boldfaced letters correspond to vectors.

capacity constraint.

Another constraint of the optimization problem arises from the fact that in order to deliver a tile requested by a user from the cache of an SBS, the tile has to be stored in the cache and the user has to reside in the coverage area of the SBS. Hence, it should hold that:

$$y^{nu}_{vglm} \leq \alpha_{nu} x^{n}_{vglm}, \forall n \in \mathcal{N}, u \in \mathcal{U}, v \in \mathcal{V}, g \in \mathcal{G}, l \in \mathcal{L}, m \in \mathcal{M} \tag{3.4}$$

Recall that α_{nu} is a binary variable that takes value 1 when user u resides in the coverage area of the nth SBS, otherwise its value is 0.

The following constraint ensures that each tile will be received only once by each client:

$$\sum_{n \in \mathcal{N}_{\mathcal{B}}} y^{nu}_{vglm} \leq 1, \forall u \in \mathcal{U}, v \in \mathcal{V}, g \in \mathcal{G}, l \in \mathcal{L}, m \in \mathcal{M} \tag{3.5}$$

In addition, we have to take into consideration limitations arising from the encoding in multiple tiles and layers. Hence, in order to recover the video in the lowest available quality, only the base layer should be delivered to the user. Differently, in order to achieve the quality that corresponds to the lth enhancement layer, the user should receive not only the lth enhancement layer, but also the base layer and all the previous enhancement layers. Therefore, we have:

$$\sum_{n \in \mathcal{N}_{\mathcal{B}}} y^{nu}_{vg(l+1)m} \leq \sum_{n \in \mathcal{N}_{\mathcal{B}}} y^{nu}_{vglm}, \forall u \in \mathcal{U}, v \in \mathcal{V}, g \in \mathcal{G}, l \in \mathcal{L}, m \in \mathcal{M} \tag{3.6}$$

The requirement for smooth playback introduces another constraint to our system. The video packets should reach the user within a specific time constraint.

When this does not happen, buffer underrun occurs, where the buffer is fed with data at a lower speed than the data is consumed. Let us denote by d_{nu} (sec/Mbit), the delay needed to transmit one Mbit from the nth SBS to the uth user, and $d_{(N+1)u}$ (sec/Mbit) the corresponding delay to transmit one Mbit from the backhaul. Obviously, $d_{(N+1)u} > d_{nu},\ n \in \mathcal{N}$ due to the additional delay needed to transmit the requested tiles from the backhaul to the SBSs. Therefore, it should be:

$$\sum_{n\in\mathcal{N}_{\mathcal{B}}}\sum_{g'\in\{1,\ldots,g\}}\sum_{l\in\mathcal{L}}\sum_{m\in\mathcal{M}} o_{vg'lm}d_{nu}\alpha_{nu}y^{nu}_{vg'lm} \leq t_{app}+(g-1)t_{disp}, \forall u\in\mathcal{U}, v\in\mathcal{V}, g\in\mathcal{G} \tag{3.7}$$

where t_{app} and t_{disp} correspond to the playback delay and the time needed to display a GOP, respectively. The idea of this constraint is that since each GOP of a video is viewed sequentially, i.e., first GOP, second GOP, etc, future GOPs have more time to be stored at the user's buffer, without interrupting smooth playback. For example, when a user starts watching a 360^o video, the gth GOP needs to be available to the user's buffer before the previous $g-1$ GOPs have been displayed. This is captured from the term $(g-1)\cdot t_{disp}$ of the right part of (3.7). For the first GOP, the available time is t_{app}, which corresponds to the start up delay a user may afford, e.g., 1-2 sec, to watch a 360^o video. The left part of (3.7) accounts the time needed for each tile of a GOP to be downloaded either from a SBS ($n \in \mathcal{N}$), or through the backhaul ($n = N+1$). If the mth tile of the lth layer of the gth GOP of the vth video is decided to be downloaded to the uth user ($y^{nu}_{vglm} = 1$), the time needed to download this tile is added up on the left part of (3.7), leaving less time available for the rest of the requested tiles to be delivered on time. This is because the sum of the time needed to deliver

all of the requested tiles should be less or equal to the available time indicated by the right part of (3.7). Furthermore, as the size of each tile, which is given by the parameter o_{vglm}, is different for each quality layer $l \in \mathcal{L}$, the acquisition of a tile at different quality layers affects the remaining time to download the rest of the requested tiles differently. Similarly, as the delay parameter d_{nu}, which denotes the delay needed to download each tile, is larger when a tile is downloaded through the backhaul, the available time to download the rest of the requested tiles is also affected from where each tile is downloaded. Finally, the parameter a_{nu} ensures that a tile will be downloaded to a user only from an SBS the user resides ($a_{nu} = 1$).

When a user obtains a tile, the distortion observed by the user decreases. Let us denote by δ_{vglm}, the average distortion reduction associated with tile $m \in \mathcal{M}$ of layer $l \in \mathcal{L}$ that belongs to GOP $g \in \mathcal{G}$ of the video $v \in \mathcal{V}$. The expected achievable cumulative distortion reduction Δ across the user population is given by:

$$\Delta = \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{V}} \sum_{g \in \mathcal{G}} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} z^u_{vglm} \delta_{vglm} \tag{3.8}$$

where the variable $z^u_{vglm} \in [0, 1]$ denotes the probability of user u to request the lth layer of tile m that belongs to the gth GOP of the vth video file. This probability can be computed based on the video and the viewport popularity distributions.

The optimization objective is to maximize the normalized expected cumulative distortion reduction over the client population. Denoting this quantity by D, we have:

$$D = \frac{1}{\Delta} \sum_{n \in \mathcal{N}_\mathcal{B}} \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{V}} \sum_{g \in \mathcal{G}} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} z^u_{vglm} \delta_{vglm} \alpha_{nu} y^{nu}_{vglm} \tag{3.9}$$

By taking into consideration the above constraints, the problem of joint caching and delivery of 360^o videos can be formally expressed as:

$$\begin{aligned} &\mathcal{P} : \max_{\mathbf{x},\mathbf{y}} D \\ &s.t. \qquad\qquad (3.1) - (3.7) \end{aligned} \tag{3.10}$$

3.4 Distributed Algorithm

The optimization problem in Eq. (3.10) is NP-hard. This is because, as we will show later, it can be decomposed into a number of subproblems, where each subproblem consists of two components that are NP-hard. The first component is the caching component that can be translated to a set of 0-1 knapsack problems, while the second component can be translated to the multidimensional multiple choice knapsack problem. Both the 0-1 knapsack problem and the multidimensional multiple choice knapsack problem are known to be NP-hard [106].

To solve the problem efficiently, we simplify the original problem by introducing a fairness constraint with respect to the cache space allocation and the delivery delay per GOP. Specifically, in order to ensure fairness regarding the cache, we limit the cache available for each GOP to $\lfloor C_n/G \rfloor$. Similarly, for the delivery delay, we assume that the delay for delivering each GOP is $t_{app}/G + t_{disp}$. This allows us to decompose the original problem in (3.10) into G subproblems $\mathcal{P}_1, \ldots \mathcal{P}_g, \ldots, \mathcal{P}_G$, where:

$$\begin{aligned} &\mathcal{P}_g : \max_{\mathbf{x}_g,\mathbf{y}_g} D_g \\ &s.t.: \end{aligned} \tag{3.11}$$

$$\mathbf{y}_g = (y^{nu}_{vglm} \in \{0,1\} : n \in \mathcal{N}_\mathcal{B}, u \in \mathcal{U}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M}) \tag{3.12}$$

$$\mathbf{x}_g = (x^{n}_{vglm} \in \{0,1\} : n \in \mathcal{N}, v \in \mathcal{V}, g \in \mathcal{G}, l \in \mathcal{L}, m \in \mathcal{M}) \tag{3.13}$$

$$\sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}}\sum_{m\in\mathcal{M}} o_{vglm} x^{n}_{vglm} \leq C_n(g), \forall n \in \mathcal{N} \tag{3.14}$$

$$\sum_{n\in\mathcal{N}_\mathcal{B}}\sum_{l\in\mathcal{L}}\sum_{m\in\mathcal{M}} o_{vglm} d_{nu} \alpha_{nu} y^{nu}_{vglm} \leq t(g), \forall u \in \mathcal{U}, v \in \mathcal{V} \tag{3.15}$$

$$y^{nu}_{vglm} \leq \alpha_{nu} x^{n}_{vglm}, \forall n \in \mathcal{N}, u \in \mathcal{U}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M} \tag{3.16}$$

$$\sum_{n\in\mathcal{N}_\mathcal{B}} y^{nu}_{vglm} \leq 1, \forall u \in \mathcal{U}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M} \tag{3.17}$$

$$\sum_{n\in\mathcal{N}_\mathcal{B}} y^{nu}_{vg(l+1)m} \leq \sum_{n\in\mathcal{N}_\mathcal{B}} y^{nu}_{vglm}, \forall u \in \mathcal{U}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M} \tag{3.18}$$

where

$$D_g = \frac{1}{\Delta}\sum_{n\in\mathcal{N}_\mathcal{B}}\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}}\sum_{m\in\mathcal{M}} z^{u}_{vglm}\delta_{vglm}\alpha_{nu}y^{nu}_{vglm} \tag{3.19}$$

In the above subproblem, $\mathbf{x}_g$ and $\mathbf{y}_g$ correspond to the cache and routing variables for the gth GOP. Eq. (3.14) is the cache constraint of the subproblem $\mathcal{P}_g$. $C_n(g)$ stands for the cache space available for the gth GOP of SBS n and is defined as:

$$C_n(g) = \lfloor C_n/G + C^{rem}_n(g-1) \rfloor, \tag{3.20}$$

where $C^{rem}_n(g-1) \geq 0$ corresponds to the amount of cache that has not been filled in with content after solving subproblem $\mathcal{P}_{g-1}$. Similarly, Eq (3.15) is the delay constraint for the subproblem $\mathcal{P}_g$, where $t(g)$ represents the delivery delay of the gth GOP and is calculated as:

$$t(g) = t_{app}/G + t_{disp} + t^{rem}(g-1) \tag{3.21}$$

Algorithm 3.1 Problem decomposition

1: Decompose $\mathcal{P}$ into $\mathcal{P}_1, \ldots, \mathcal{P}_g, \ldots, \mathcal{P}_G$ according to Eq. (3.11)
2: Set GOP index $g \leftarrow 1$
3: Set cache $C_n(1) \leftarrow \lfloor C_n/G \rfloor,\ \forall n \in \mathcal{N}$
4: Set delay $t(1) \leftarrow t_{app}/G + t_{disp}$
5: **while** $g \leq G$ **do**
6: Determine optimal delivery and cache policy $\mathbf{x}_g, \mathbf{y}_g$ for gth GOP by solving $\mathcal{P}_g$
7: Compute $\forall n \in \mathcal{N}$
$C_n^{rem}(g) \leftarrow C_n(g) - \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} o_{vglm} x_{vglm}^n$
8: Compute
$t^{rem}(g) \leftarrow t(g) - \sum_{n\in\mathcal{N}_\mathcal{B}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} o_{vglm} d_{nu} \alpha_{nu} y_{vglm}^{nu}$
9: Update cache $\mathcal{C}_n(g+1) \leftarrow \lfloor C_n/G + C_n^{rem}(g) \rfloor$
10: Update delay $t(g+1) = t_{app}/G + t_{disp} + t^{rem}(g)$
11: $g \leftarrow g + 1$
12: **end while**
13: $\mathbf{x} = \bigcup_{g\in\mathcal{G}} \mathbf{x}_g$
14: $\mathbf{y} = \bigcup_{g\in\mathcal{G}} \mathbf{y}_g$

with $t^{rem}(g-1) \geq 0$ being the time remaining from the delivery of the previous GOP. To obtain the global policy for routing and caching, i.e., $\mathbf{x}$ and $\mathbf{y}$, we solve each subproblem sequentially starting from the first GOP. After solving the subproblems for all GOPs, we combine the optimal routing and caching solutions $\mathbf{x}_g$ and $\mathbf{y}_g$ to obtain the global routing and caching policy. This procedure is summarized in Algorithm 3.1.

We would like to note that the above approach of solving each subproblem $\mathcal{P}_g$ for each GOP sequentially is one of the possible ways to solve our problem of offline caching for 360^o videos. The employment of more advanced policies, could further improve the performance of the proposed system. However, more advanced policies would be more complicated, without affecting the conclusions

regarding how the use of advanced coding tools, i.e., encoding into multiple layers and tiles, is beneficial for edge caching systems of 360^o video.

To solve each subproblem $\mathcal{P}_g$, we use the method of Lagrange partial relaxation. Specifically, we relax the constraint (3.16), which is the "hard" constraint of the problem as it couples the caching and routing variables. As a result of relaxing this constraint, the Lagrangian subproblem consists of two optimization problems that can be solved independently: one that involves only the cache related variables of the original problem and another that involves only the delivery variables of the problem.

Specifically, let us define the set of Lagrange multipliers for every $g \in \mathcal{G}$ as follows:

$$\boldsymbol{\lambda}_g = (\lambda^{nu}_{vglm} \geq 0, \forall n \in \mathcal{N}, u \in \mathcal{U}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M}) \tag{3.22}$$

By relaxing the constraint in (3.16) we obtain the Lagrangian function:

$$\begin{aligned} &L(\boldsymbol{\lambda}_g, \mathbf{x}_g, \mathbf{y}_g) = \\ &\max_{\mathbf{x}_g, \mathbf{y}_g} \sum_{n\in\mathcal{N}} \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} (\frac{1}{\Delta} z^u_{vglm} \delta_{vglm} \alpha_{nu} - \lambda^{nu}_{vglm}) y^{nu}_{vglm} \\ &\quad + \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} \frac{1}{\Delta} z^u_{vglm} \delta_{vglm} y^{(N+1)u}_{vglm} \\ &\quad + \sum_{n\in\mathcal{N}} \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} \lambda^{nu}_{vglm} \alpha_{nu} x^n_{vglm} \end{aligned} \tag{3.23}$$

Thus, the Lagrangian dual problem can be expressed as:

$$\min_{\boldsymbol{\lambda}_g \geq \mathbf{0}} L(\boldsymbol{\lambda}_g, \mathbf{x}_g, \mathbf{y}_g) \tag{3.24}$$

where $\mathbf{x}_g, \mathbf{y}_g$ satisfy (3.12)-(3.15), (3.17), (3.18). We can easily observe that the Lagrange function can be re-expressed as follows:

$$L(\boldsymbol{\lambda}_g, \mathbf{x}_g, \mathbf{y}_g) = f(\mathbf{x}_g) + k(\mathbf{y}_g) \tag{3.25}$$

where $f(\mathbf{x}_g)$ and $k(\mathbf{y}_g)$ correspond to two independent optimization sub-subproblems $P1_g$ and $P2_g$, respectively. These sub-subproblems are defined as:

$$P1_g : \max_{\mathbf{x}_g} \sum_{n\in\mathcal{N}} \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} \lambda^{nu}_{vglm} \alpha_{nu} x^n_{vglm} \tag{3.26}$$

s.t.:

$$\sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} o_{vglm} x^n_{vglm} \leq C_n(g), \forall n \in \mathcal{N}$$

$$\mathbf{x}_g = (x^n_{vglm} \in \{0,1\} : n \in \mathcal{N}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M})$$

and

$$P2_g : \max_{\mathbf{y}_g} \sum_{n\in\mathcal{N}} \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} (\frac{1}{\Delta} z^u_{vglm} \delta_{vglm} \alpha_{nu} - \lambda^{nu}_{vglm}) y^{nu}_{vglm} + \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} \frac{1}{\Delta} z^u_{vglm} \delta_{vglm} y^{(N+1)u}_{vglm} \tag{3.27}$$

s.t.:

$$\sum_{n\in\mathcal{N}_\mathcal{B}} y^{nu}_{vglm} \leq 1, \forall u \in \mathcal{U}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M}$$

$$\sum_{n\in\mathcal{N}_\mathcal{B}} y^{nu}_{vg(l+1)m} \leq \sum_{n\in\mathcal{N}_\mathcal{B}} y^{nu}_{vglm}, \forall u \in \mathcal{U}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M}$$

$$\sum_{n\in\mathcal{N}_\mathcal{B}} \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} o_{vglm} d_{nu} \alpha_{nu} y^{nu}_{vglm} \leq t(g), \forall u \in \mathcal{U}, v \in \mathcal{V}$$

$$\mathbf{y}_g = (y^{nu}_{vglm} \in \{0,1\} : n \in \mathcal{N}_\mathcal{B}, u \in \mathcal{U}, v \in \mathcal{V}, l \in \mathcal{L}, m \in \mathcal{M})$$

Since $P1_g$ involves only the caching variables $\mathbf{x}_g$, we refer to it as the *caching component*. In order to determine the optimal cache allocation $\mathbf{x}_g$ for $P1_g$, the problem is further decomposed into N $0-1$ knapsack problems, which can be solved independently using optimization methods such as Dynamic Programming [106]. In this case, we maximize the objective function of $P1_g$, considering that the cache space of each SBS is a knapsack.

The optimization subproblem $P2_g$ involves only the routing decision variables $\mathbf{y}_g$, hence it is called hereafter as the *routing component*. The subproblem $P2_g$ can be mapped to a multidimensional multiple choice knapsack problem. To find the optimal solution for the routing component we use CPLEX [107].

To solve the Lagrangian dual problem in (3.24), we apply the sub-gradient method to update the dual variables iteratively. Specifically, we start with non-negative values for the Lagrangian multiplier variables, and then based on the solution of the $P1_g$ and $P2_g$, in each iteration τ the dual variables are updated as follows:

$$\lambda^{nu}_{vglm}(\tau+1) = [\lambda^{nu}_{vglm}(\tau) - \sigma(\tau) d(\lambda^{nu}_{vglm}(\tau))]^{+} \tag{3.28}$$

In (3.28), $[s]^{+} = \max(0, s)$ is an operator that ensures that the dual variables cannot take negative values, and $\sigma(\tau)$ expresses the step size which controls the convergence properties of the sub-gradient algorithm at each iteration. In addition, $d(\lambda^{nu}_{vglm}(\tau))$ expresses the sub-gradient of the dual problem with respect to $\lambda^{nu}_{vglm}(\tau)$ and is given by:

$$d(\lambda^{nu}_{vglm}(\tau)) = -y^{nu}_{vglm}(\tau) + z^{u}_{vglm} \delta_{vglm} \alpha_{nu} x^{n}_{vglm}(\tau) \tag{3.29}$$

If we denote by LB the Lower Bound of the subproblem $\mathcal{P}_g$, UB the value of the Lagrange function at iteration τ, and $\boldsymbol{\phi}(\tau) = d(\boldsymbol{\lambda}(\tau))$ the subgradient, the

Algorithm 3.2 Primal-Dual Algorithm

1: **Require:** $\tau = 1$, $\tau_{max} = 1000$, $\boldsymbol{\lambda}_g(1) = 0.2$, $UB = +\infty$, $LB = -\infty$, $w \in (0, 2]$, $\epsilon = 0.01$
2: **while** $\left|\frac{UB-LB}{LB}\right| \geq \epsilon$ **and** $\tau \leq \tau_{\max}$ **do**
3: Determine $\mathbf{x}_g(\tau)$ by solving $P1_g$
4: Determine $\mathbf{y}_g(\tau)$ by solving $P2_g$
5: $UB = L(\boldsymbol{\lambda_g}, \mathbf{x}_g, \mathbf{y}_g)$ and $\sigma(\tau) = w\frac{UB-LB}{\left\|\boldsymbol{\phi}(\tau)\right\|^2}$
6: Update Lower Bound (LB)
7: Update the dual variables $\boldsymbol{\lambda}_g(\tau + 1)$ using (3.28)
8: Update $\tau = \tau + 1$
9: **end while**

step size can be easily calculated using the formula below:

$$\sigma(\tau) = w\frac{UB - LB}{\left\|\boldsymbol{\phi}(\tau)\right\|^2} \tag{3.30}$$

In the above equation, $w \in (0, 2]$ is a positive constant that scales the step size. The value of the Upper Bound is the value of the Lagrange function at each iteration, while the Lower Bound equals to the value of the objective function of the subproblem $\mathcal{P}_g$, by finding a feasible solution to it. The overall algorithm is summarized in Algorithm 3.2, where the variable ϵ defines the maximum acceptable distance between the UB and the LB.

3.5 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm exploiting cache collaboration and coding in multiple layers and tiles. First, we briefly describe all the schemes under comparison and give the simulation setup. Next, we provide experimental results that showcase the impact of the various system

parameters on the performance of the proposed scheme. Finally, we discuss the convergence of the proposed algorithm.

3.5.1 Simulation Setup

The schemes under comparison, including our proposed method, are described below:

1. *Independent Caching-No Tiles (ICNT):* In this scheme, each video is encoded into a number of versions, where each version is encoded in a single tile and represents the whole scene in low quality along with a viewport in high quality. We assume that each user is associated only with one SBS, which has the maximum signal-to-interference-plus-noise ratio (SINR). The caching and delivery policy is found by solving (3.10) for each SBS separately. In this case, the caching and delivery variables are per video per GOP.

2. *Joint Caching-No Tiles (JCNT):* In this scheme, each video is encoded into a number of versions, where each version is encoded in a single tile and represents the whole scene in low quality along with a viewport in high quality. Differently from ICNT, JCNT is a collaborative caching scheme that exploits the possible association of some users to more than one SBSs. The caching and routing decisions are jointly made using Algorithm 3.2. As previously, the caching and routing variables are per video per GOP as we do not use coding into multiple layers and tiles.

3. *Joint Caching-Multiple Layers (JCL):* In this scheme, each video is encoded in multiple quality layers and a single tile. In JCL, the base layer represents

the whole scene in low quality, while the enhancement layers improve the quality of the part of the scene that corresponds to the viewport. The caching and routing policy is determined by solving Algorithm 3.2 on a per quality-layer basis. In this scheme, the caching and routing variables are per video per layer per GOP.

4. *Independent Caching (IC):* In this scheme, each video is encoded in multiple quality layers and tiles. Similarly to ICNT we assume that each user is associated with only one SBS, i.e., the one with the maximum SINR. This scheme exploits the granularity of encoding the video into multiple tiles and quality layers. The caching and delivery policy is found by solving (3.10) for each SBS separately. In this case, the caching and delivery variables are per tile and per layer per GOP.

5. *Proposed Algorithm:* This is the proposed scheme where videos are encoded in multiple quality layers and tiles. This scheme is similar to IC in that it takes advantage of the granularity of encoding the video into multiple tiles and quality layers. It further advances IC by exploiting collaboration opportunities among SBSs. In this scheme, the caching and routing decisions are jointly made on per tile and per layer basis using Algorithm 3.2.

The constant ϵ which controls the convergence is set to 0.01. This ensures that UB and LB are close enough and the number of iterations required for the convergence of the algorithm is reasonable. The value of the weight w is set to 0.02, which was found by experimentation.

We assume for all conducted experiments a cellular network with 5 SBSs unless otherwise stated. The transmission radius of each SBS is set to p_n = 300m, while each SBS has cache capacity sufficient to store 10% of the size of

the content library. The coverage radius of the MBS is set to $p_{N+1} = 1000$m, which is large enough to allow communication between the MBS and all SBSs. We consider 30 users, who are randomly placed in the coverage area of the SBSs. The transmission delay from a SBS to a user is set to $d_{nu} = 1$ sec/Mbit $\forall n \in \mathcal{N}, \forall u \in \mathcal{U}$, while the transmission delay when the content has to be fetched from the backhaul to the user is $d_{(N+1)u} = 5$ sec/Mbit $\forall u \in \mathcal{U}$.

The content library contains $V = 10$ videos with resolution 1280×720. In our scheme the videos are encoded into $M = 12$ tiles per frame and $L = 2$ quality layers per tile. We assume that the size of each viewport is 2×2 tiles. Each video consists of $G = 30$ GOPs with duration of 1 sec per GOP. The playback delay t_{app} for each video is 1sec, and the display time of a GOP t_{disp} is 1 sec. The videos are encoded using the scalable extension (SHVC) [104] of H.265/HEVC [4] standard, which allows encoding in tiles and layers. Although the results we obtain are for video sequences encoded by SHVC, the derived conclusions are valid for videos encoded in tiles and layers by other codecs as well. Finally, we would like to note that we consider equirectangular projection, but our method is transparent to the employed projection, and hence is applicable when other projections such as cube or pyramid, etc. are used.

For the sake of simplicity, we assume that 40% of the videos in the content library have similar characteristics with the "Hog Rider" video sequence, 30% with the "Roller Coaster" video sequence, while the remaining 30% with the "Chariot Race" video sequence. These videos were obtained from YouTube. The average size and the distortion reduction per tile for both base and enhancement layers are given in Table 3.1.

The probability that a specific video is requested by a user follows the Zipfian distribution [108] with shape parameter $\eta_v = 1$. Hence, the probability that a

Table 3.1: Distortion reduction per tile and layer for the considered video sequences

Video	o_{vg1t} (in Mbits)	o_{vg2t} (in Mbits)	δ_{vg1t}	δ_{vg2t}
"Hog Rider"	0.010	0.125	118	125
"Roller Coaster"	0.016	0.167	292	298
"Chariot Race"	0.029	0.275	187	192

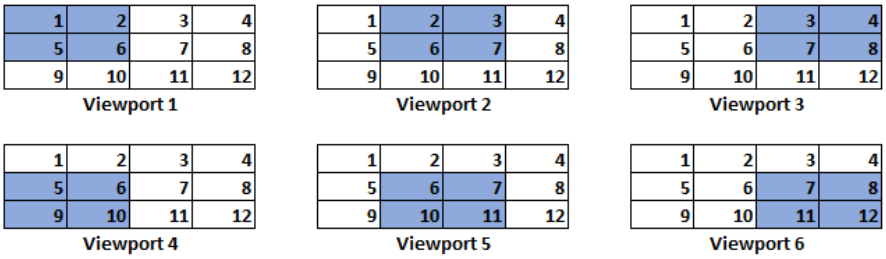

Figure 3.3: Illustration of viewports considered for the evaluation. Numbers indicate tiles indices, while highlighted areas denote the viewports.

user requests the video $v \in \mathcal{V}$ is calculated as:

$$p_v = \frac{1/v^{\eta_v}}{\sum_{v \in \mathcal{V}} 1/v^{\eta_v}} \tag{3.31}$$

To study the effect of non-uniform popularity of tiles we consider that only the six viewports depicted in Fig. 3.3 can be requested with non-zero probability by the users for each 360^o video. Assuming that the requests for these viewports are uniform, results in central tiles (which belong to multiple viewports) having higher popularity. We would like to emphasize that the set of viewports presented in Fig. 3.3 is not complete; under equirectangular projection there are other available viewports that we do not consider here for the sake of obtaining the desirable distribution over the tiles and deriving interpretable results. This, however, does not change the conclusions drawn in this section.

As JCL, ICNT and JCNT schemes do not exploit encoding of 360^o video

into tiles and/or layers, these schemes would be heavily penalized if a viewport was not cached in the SBSs caches as data chunks of large size would not be delivered on time through the backhaul. Thus, for the sake of fairness, we allow soft satisfaction of users' demands [109] for these two schemes. Specifically, when a viewport of a video requested by a user is not cached at an SBS, but another overlapping viewport of this video is cached, we assume that the cached viewport is delivered to the user. This way tiles that correspond to the overlap area of the requested and the delivered viewport are recovered at the highest quality, while the rest are recovered at the base quality. For example, if the requested and delivered viewports are viewports 1 and 2 in Fig. 3.3, respectively, tiles 2 and 6 will be recovered in high quality. In such a case, if the video was encoded in two quality layers, the soft cache hit ratio is computed by the following formula

$$\text{SoftCHR} = \frac{nd_t^b + nd_t^e}{nr_t^b + nr_r^e} \tag{3.32}$$

where the numerator corresponds to the number of tiles delivered to the user and the denominator to the number of requested tiles. Specifically, nd_t^b (nd_t^e) and nr_t^b (nr_t^e) are the number of delivered tiles at base (enhancement) quality and requested number of tiles at base (enhancement) quality, respectively. For the example above, the numerical values are $nd_t^b = nr_t^b = 12$, $nd_t^e = 2$, and $nd_r^e = 4$ and hence, the soft cache hit ratio is 0.875. We would like to note that soft satisfaction improves only the cache hit ratios of non-tiles based schemes (ICNT, JCNT, JCL). These schemes suffer from low cache hit ratio when the requested data cannot be delivered in time due to the use of larger chunks of data. In such case, the user can often obtain part of the required viewport through the delivery of another overlapping viewport. Through soft satisfaction cache hit ratio, we

aim at capturing the improved interactivity a user experiences when instead of a required viewport, another viewport is delivered to the user. This happens as the user can navigate in part of the requested viewport but at degraded quality. Soft satisfaction assumes that tiles from base and enhancement layers are of equal importance and, hence, it is not a fully optimized QoE metric. The consideration of different weights for the importance of base and enhancement layers could lead to a more accurate QoE metric that better captures the tradeoff between quality and interactivity, but this is out of the scope of this thesis.

3.5.2 Parameter Analysis

1) Cache Size

We first study the impact of the cache size of SBSs on the achieved distortion reduction D as computed in (3.9). We vary the cache capacity C_n in the range [5,25]% of the size of the content library and measure the distortion reduction as the percentage of the maximum achievable cumulative distortion reduction. The performance of the schemes under comparison is shown in Fig. 3.4. From the results, we can see that the proposed scheme outperforms all the other schemes, for the entire range of SBSs' cache sizes. Specifically, we can observe that when SBSs can cache 5% of the content library, which is common for networks handled by mobile network operators, the performance gap between the proposed algorithm and JCNT and ICNT, which do not assume encoding into multiple layers and tiles, is $\sim 30\%$ and $\sim 55\%$, respectively. This gap is due to the fact that the proposed algorithm takes advantage of the increased granularity in the caching and routing decisions offered by encoding the video in multiple tiles and layers, i.e., smaller chunks of data. This permits our algorithm to cache the most

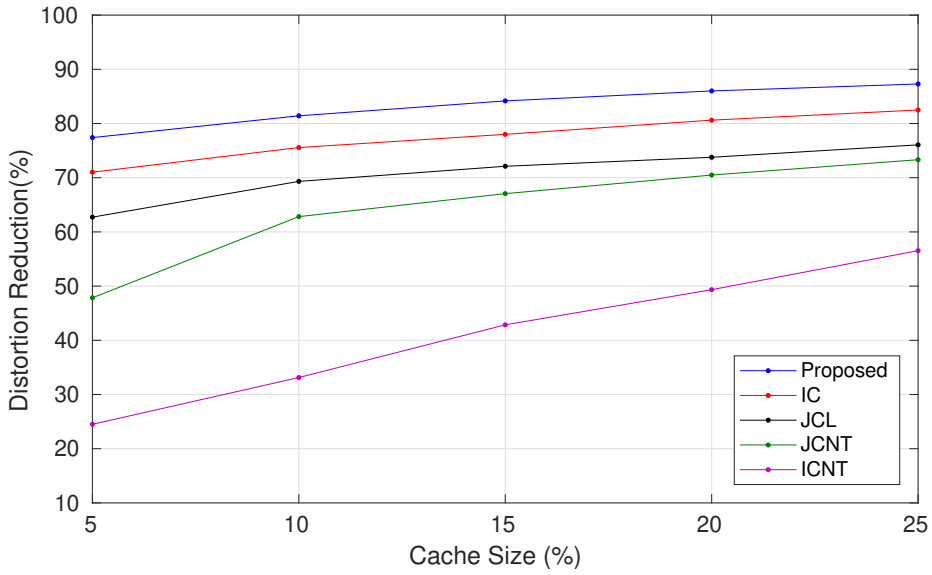

Figure 3.4: Distortion reduction with respect to the cache size for all schemes under comparison.

popular and significant chunks of data locally in the SBSs, while preserving high data diversity across the SBSs thanks to collaboration among SBSs.

Similar trends but smaller gains, $\sim 15\%$, can be observed when we compare the proposed algorithm with JCL, which assumes encoding into quality layers and a single tile. The observed gains are because the proposed algorithm benefits from the encoding in tiles, unlike JCL which only considers layers. Specifically, in the proposed scheme the cached tiles are obtained locally from the SBS, while the rest of the tiles are delivered from the backhaul within the delivery deadlines. This is often not possible when JCL is used, as the delay constraint may not allow timely delivery of the larger data chunks that correspond to quality layers. Finally, to understand the gains arising from collaborative caching we compare the proposed scheme with IC. We note that the proposed algorithm still outperforms the IC scheme despite the fact that the latter uses tile and layered encoding. This performance difference is attributed to the exploitation of

the collaborative caching opportunities among SBSs which leads to gains when users reside in the overlap area between the coverage area of multiple SBSs. As we will show later in this section, the gains grow when the coverage area of the SBSs increases, i.e. more users are in the overlap area of several SBSs.

As we can further observe from Fig. 3.4 when SBSs' cache capacity increases, the performance gap between the proposed algorithm and the comparison schemes becomes smaller in most of the cases. Specifically, when each SBS can cache 25% of the content catalogue, the proposed algorithm outperforms JCNT and ICNT by $\sim$ 15% and $\sim$ 30%, respectively. The performance gap is smaller for such cache sizes because larger cache capacity permits ICNT and JCNT to cache more videos in SBSs and less requests need to be served through the backhaul links, enabling the timely delivery of more data. The performance gap between the proposed algorithm and the JCL is $\sim$ 10%. JCL cannot close the gap further, as the data chunks corresponding to the layers are of larger size. When we compare the proposed scheme with IC we note that the performance gap stays constant because both schemes take advantage of the additional cache size, but the proposed scheme in addition exploits collaboration opportunities among SBSs. However, it is expected that both schemes will perform identically if the cache size is further increased, as most of the data will be cached locally and the backhaul link will not be used. These results are not presented as they are of no practical interest.

The above discussion regarding the benefits of making joint routing and caching decisions on per layer and tile basis are verified by Fig. 3.5, which depicts the achieved cache hit ratio for various cache sizes for all the schemes under comparison. From this figure, we can note that for all cache sizes the proposed scheme achieves higher cache hit ratio. This means that the majority

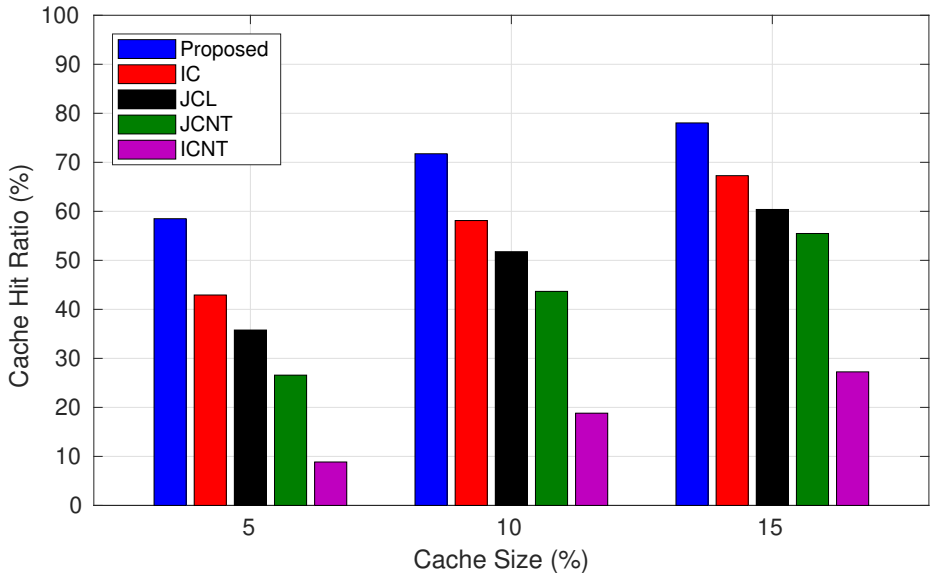

Figure 3.5: Cache hit ratio with respect to the cache size for all schemes under comparison.

of requests are served locally from the SBSs caches without the need to use the backhaul links. On the contrary, the low cache hit ratio in the ICNT and JCNT schemes indicates that user requests are not served locally and content is fetched from the remote servers, which due to the increased delivery delay and chunk size (compared to tile size in our scheme) may not be always delivered within the time constraints. This, in turn, results in a small distortion reduction. When we compare the proposed scheme with JCL and IC we can observe that for 10% cache size, it outperforms these schemes by $\sim 20\%$ and $\sim 14\%$, respectively. The performance difference is significant, however it is smaller than that between the proposed scheme and ICNT and JCNT. This is because JCL and IC encode the data in smaller chunks than ICNT and JCNT, i.e., JCL considers encoding in layers and IC encoding in layers and tiles. The gains of the proposed scheme over IC are due to the exploitation of collaborative caching opportunities.

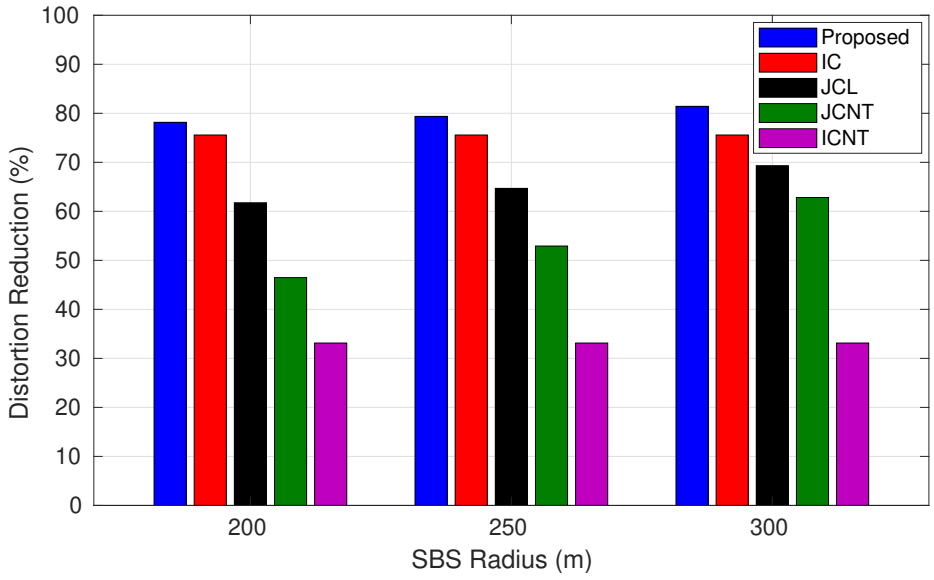

Figure 3.6: Distortion reduction with respect to the radius of the SBSs for all schemes under comparison.

2) SBS radius

To investigate the impact of the SBSs collaboration, we change the transmission radius (coverage area) of the SBSs on the distortion reduction from 200m up to 300m. The results are depicted in Fig. 3.6 from where we can note that an increase in the SBS radius affects the performance of the schemes performing collaborative caching, i.e., the proposed algorithm, JCL and the JCNT scheme, while the performance of non-collaborative caching schemes, i.e., ICNT and IC, stays invariant. This can be explained by the fact that for the collaborative caching schemes a higher overlap in the coverage area of the SBSs results in more users being in the overlap area of multiple SBSs. On the contrary, for the non-collaborative caching schemes an increase in the coverage area is not translated into performance gains because each user is associated with a single SBS, that with the maximum SINR. From the results, we can further see that the proposed scheme in all cases outperforms all other schemes. Additionally, the

performance gap between the proposed scheme and IC grows from $\sim 2\%$ to $\sim 6\%$, which makes clear the significance of exploiting SBSs collaboration opportunities. The scheme that profits the most from the increase in SBS transmission radius is JCNT, which considers the largest chunks of data (entire videos). For this scheme, cache collaboration partially compensates for the size of the data chunks.

3) SBS communication link delay

In Fig. 3.7, we illustrate the effect of the delay of the communication link between the users and the SBSs on the distortion reduction D. In particular, we assume that the value of the SBS delay d_{nu} varies in the range $[0.5, 2.5]$ sec/Mbit, while the backhaul delay remains constant at 5 sec/Mbit. We can observe that an increase in the SBS delay (i.e., the communication link becomes slower) results in a decrease in distortion reduction D for all the schemes. We can note that the performance of the proposed algorithm is only slightly affected by the increase in the SBS delay ($\sim 5\%$). Further, we can see that IC behaves similarly to the proposed algorithm. This behavior can be explained by the fact that both schemes exploit tile encoding; the superior performance of the proposed scheme is due to the exploitation of the collaboration opportunities between SBSs. We can see that JCNT and ICNT are more affected by the higher SBS communication link delays. As a result, the performance difference between the proposed algorithm and ICNT and JCNT grows from $\sim 48\%$ to $\sim 50\%$ and from $\sim 18\%$ to $\sim 38\%$, respectively, as the link delay increases from 0.5 to 2.5 sec/Mbit. This happens as ICNT and JCNT schemes do not employ encoding into tiles and layers. Hence, they are affected the most by the increase in the SBSs link delay, since this prevents the timely delivery of large data chunks, resulting in a significant decrease in the achieved distortion reduction.

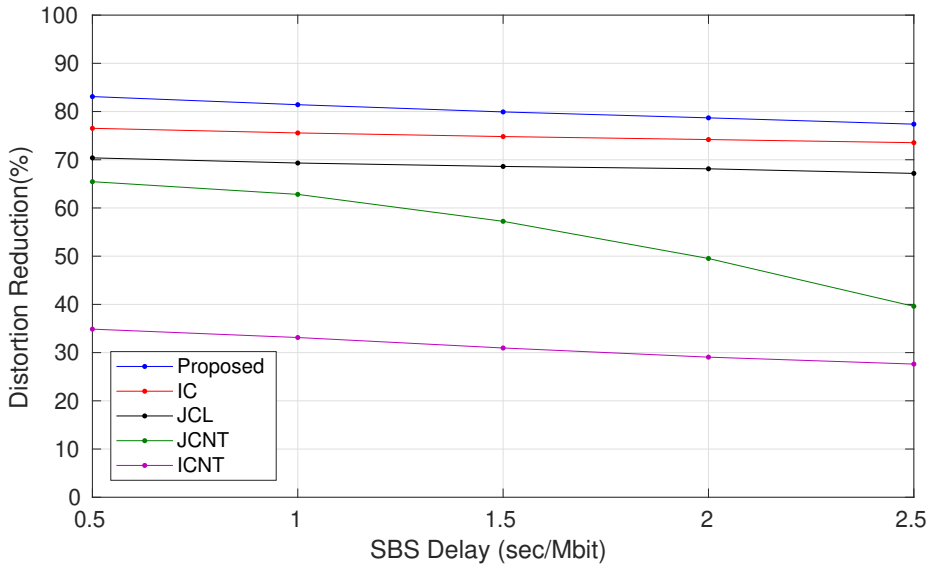

Figure 3.7: Distortion reduction with respect to the SBS Delay for all schemes under comparison.

JCL, which considers encoding in layers, performs better than ICNT and JCNT because the employed data chunks are of smaller size, which means that they are affected less from an increase in the SBSs communication link delay. From this comparison, it is evident that caching of smaller chunks of data is beneficial, and that collaborative caching and delivery decisions can bring additional gains.

4) Backhaul link delay

In Fig. 3.8, we compare the performance of all the schemes with respect to the backhaul delay. Specifically, we consider that the backhaul delay $d_{(N+1)u}$ takes values in the range [5, 15] sec/Mbit. From the results, we can conclude that an increase in the backhaul link delay results in users experiencing a lower distortion reduction. This happens because an increase in the backhaul delay results in less data being delivered through the backhaul link within the time constraint. The increase in the backhaul delay has a more significant impact on the JCNT and

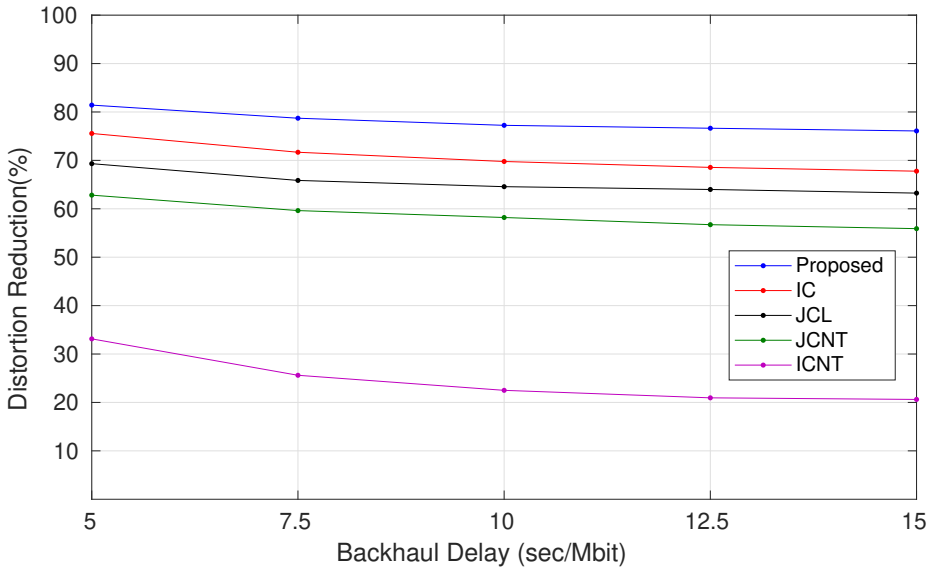

Figure 3.8: Distortion reduction with respect to the Backhaul Delay for all schemes under comparison.

ICNT, while there is a gentle decrease for all other schemes. In particular, the difference in the distortion reduction between the proposed method and JCL stays above 12%, while the difference between the proposed method and the IC varies from $\sim$ 6% to $\sim$ 9%. This is because of JCL encoding into layers, and the proposed scheme and IC encoding in both tiles and layers. By increasing the backhaul delay, the proposed scheme and IC are not affected significantly as the tiles (layers for JCL) of the requested videos are stored and delivered by the SBSs. Also, due to the smaller size of the tiles (layers for JCL) compared to videos, more tiles can be delivered in time to the users through the backhaul even when the backhaul delay increases. ICNT and JCNT are affected more by this delay increment, and the performance gap grows from $\sim$ 48% and $\sim$ 18% to $\sim$ 56% and $\sim$ 20%, respectively.

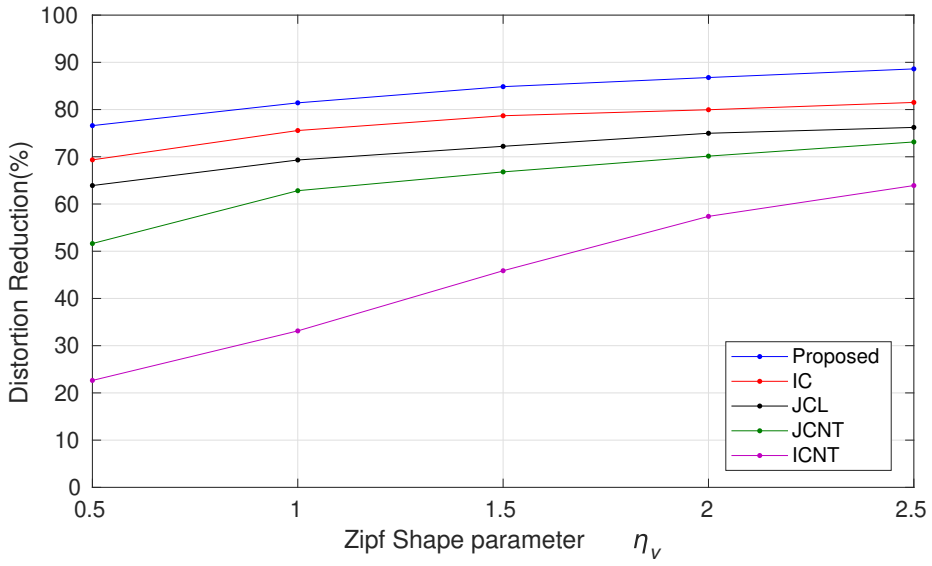

Figure 3.9: Distortion reduction with respect to Zipf shape parameter for all schemes under comparison.

5) Video popularity distribution

In Fig. 3.9, we examine the effect of the skewness parameter of the Zipfian distribution on the distortion reduction. To this end, we vary the shape parameter η_v from 0.5 up to 2.5. For small values of η_v, i.e., very diverse content demands, the difference between the proposed scheme and ICNT and JCNT is approximately 54% and 25%, respectively. This difference is due to the encoding in tiles which permits to cache the base layer tiles locally and hence satisfy most of the users' requests with a basic video quality. This is not the case in ICNT and JCNT, as without encoding in tiles and layers, the local caches are occupied by large data chunks which can satisfy only a small percentage of the diverse users' requests. Similarly to the previous comparisons, we can see that the performance difference between the proposed algorithm and JCL and IC is smaller than that with ICNT and JCNT, because the former schemes consider data chunks of smaller size, layers for JCL and tiles for IC. The proposed scheme outperforms IC because of the

exploitation of cache collaboration opportunities. The performance gap closes when the value of η_v becomes higher, as the majority of the user requests refer to a smaller number of videos. Higher values of η_v are more beneficial for ICNT and JCNT, as these schemes do not encode videos in tiles and layers, but even for such η_v values, ICNT and JCNT schemes performance remains inferior to that of the other schemes.

6) Viewports popularity distribution

We also study the effect of viewports popularity on the performance of the considered schemes. To this aim, we examine three viewport popularity distributions: (a) the viewport popularity distribution described in Section 3.5.1, which we term hereafter as "BiGauss", as according to this distribution the central tiles are more popular than the rest; (b) a uniform distribution where we consider all possible viewports (allowing viewports to fold around) with uniform popularity which results in uniform popularity of tiles; and (c) a selective viewport distribution where for each video all users request the same viewport of this video. We compare the performance of the proposed scheme and the JCNT scheme. The results are presented in Fig. 3.10. As expected the more concentrated are users' requests in fewer tiles, the higher is the distortion reduction for both schemes. Further, we can see that the proposed scheme outperforms significantly JCNT, as caching and delivery decisions are made per tile, and tiles are of small size which means they can be delivered to the users within the delivery deadlines. The performance gains become smaller when the cache size increases. This is attributed to the fact that when the cache space increases, JCNT can cache more than one viewports of popular videos assuming BiGauss or uniform distribution, and more videos for selective distribution. For the proposed scheme the

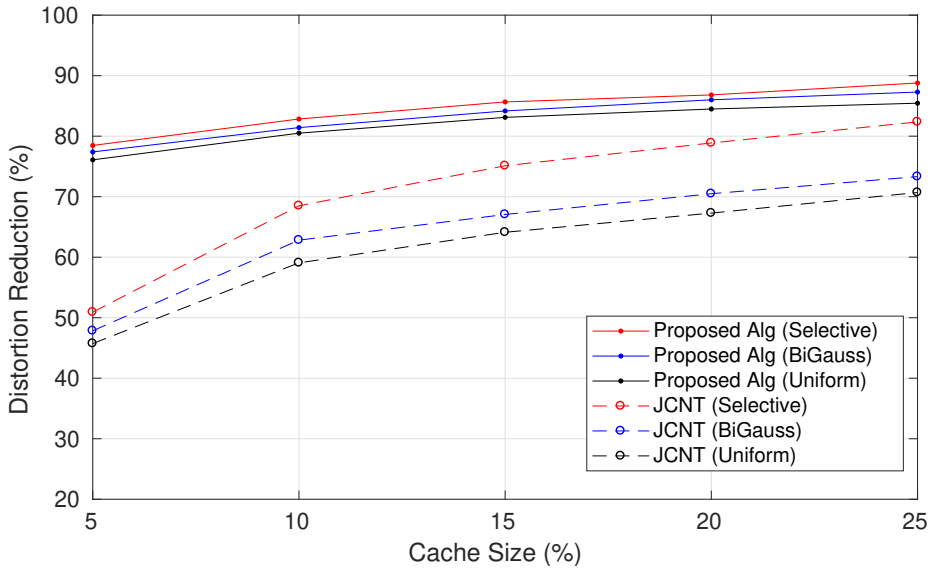

Figure 3.10: Distortion reduction with respect to the cache size for the proposed scheme and JCNT considering three viewport popularity distributions.

distortion reduction improvements are smaller, as already for small cache sizes our scheme is able to achieve distortion reductions that exceeds 75%. In the proposed scheme even when cache resources are scarce, thanks to tile and layer granularity, the most popular tiles are cached locally at the SBS and the rest can be delivered through the MBS.

3.5.3 Convergence

For the sake of completeness, we examine the convergence of the proposed algorithm. Recall that each subproblem $\mathcal{P}_g$ involves routing and caching decisions of gth GOP and is obtained by solving iteratively Algorithm 2. The convergence for subproblem $\mathcal{P}_1$, i.e., for the first GOP, is depicted in Fig. 3.11. Similar convergence behavior is noticed for all GOPs, but is omitted here for brevity reasons. From Fig. 3.11, we can see that the proposed algorithm requires only a few hundreds iterations to converge. This is in accordance with the work in [110], where

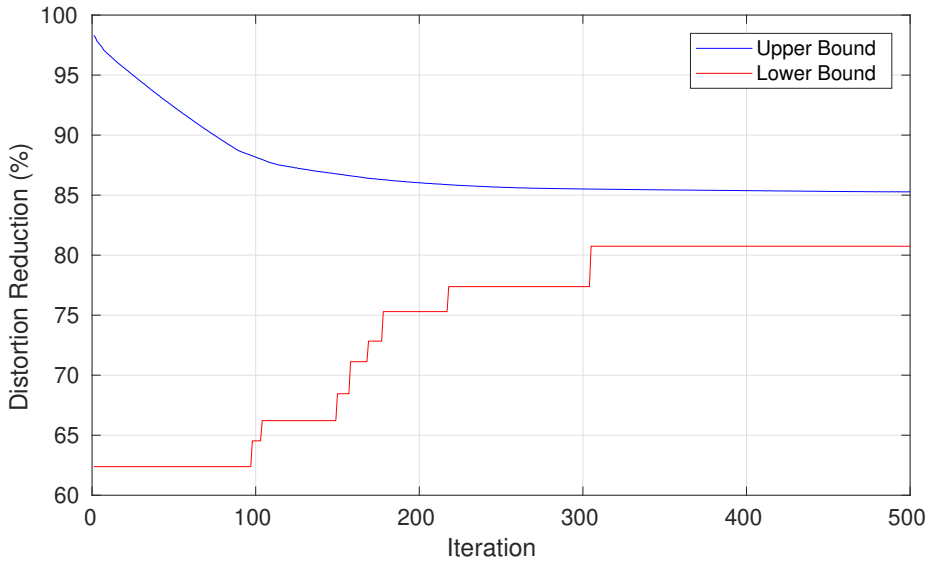

Figure 3.11: Convergence of the proposed algorithm.

it was shown that the Lagrange partial relaxation method provides an upper bound when applied to $\mathcal{P}_g$, which is guaranteed to converge to the optimal value of the $\mathcal{P}_g$ in a finite number of iterations. We would like to note that $\mathcal{P}1$ and $\mathcal{P}2$ problems were solved in approximately the same amount of time. Although $\mathcal{P}1$ is NP-hard, as it composed of multiple 0-1 Knapsack problems, it is possible to solve it in pseudo-polynomial time. Specifically, as the capacity of each SBS (knapsack) is C_n, and the maximum number of available items are M tiles of L quality layers of V videos, the theoretical run-time to obtain the caching decisions of one SBS, following a dynamic programming approach, is $\Theta(C_n VLM)$. Since the number of GOPs, which captures the length of the 360^o videos, affects directly the number of decision variables, the run-time to solve our problem increases linearly with the number of GOPs. The run-time to solve our problem can be reduced by solving each one of the subproblems $\mathcal{P}1$ and $\mathcal{P}2$ in parallel. Finally, since our scheme is an offline caching scheme, the caching decisions are proactively decided in off-peak hours, e.g., late hours at night, which allows our

scheme to be used for the caching of 360^o videos without the run-time of our problem being prohibiting.

3.6 Conclusion

In this chapter, we studied the problem of the joint caching and delivery of 360^o videos in cellular networks comprising an MBS and multiple SBSs that can collaborate. To maximize the quality of the video delivered to the end users, we exploited advanced video coding tools offered by video coding standards such as HEVC that permits encoding of video in multiple tiles and layers offering greater granularity of information. We also exploited SBSs collaboration opportunities to prevent neighboring SBSs from caching the same video data, which is an efficient technique to reduce the load of the backhaul link. The proposed algorithm decided the routing and caching policies by taking into consideration not only the content importance, but also video popularity at tile level. As the original problem is of high complexity, we decomposed it into a number of subproblems, one per each GOP, which were then solved sequentially. To further reduce the complexity of the proposed algorithm, we decoupled the subproblems into their routing and caching components and solved them using the Lagrange decomposition method. The experimental evaluation showed that collaborative caching and video encoding into tiles and quality layers allows to cache the most important data locally in the SBSs and deliver it in a timely manner to the users. Consequently, our method outperformed significantly its counterparts that did not use collaboration and/or encoding into multiple tiles and quality layers.

4

Viewport-Aware Deep Reinforcement Learning Approach for 360^o Video Caching

4.1 Introduction

In the previous chapter, we showed that caching 360^o videos at the edge servers on per quality layer and tile basis, instead of entire 360^o videos, leads to a better cache hit ratio, and improves the overall quality of the delivered video to the users. It was considered that the content popularity distribution is known at the SBSs in advance, hence our system was an offline caching system. How-

ever, the content popularity distribution may change dynamically, and is not always known *a priori*. Motivated by the above, in this chapter we propose a reactive caching scheme that does not assume that the videos and viewports' popularity are known. To this end, we formulate the online content cache placement/eviction of 360^o videos as a Markov Decision Process (MDP). To reduce the dimensionality of the cache optimization problem, we introduce the concept of virtual viewports. Virtual viewports have the same number of tiles with original viewports, while the tiles forming these viewports are the most popular ones (see Fig. 1.5). The online content cache placement/eviction problem is solved using the Deep Q-Network (DQN) algorithm, which can be efficiently used for large state and action spaces. We extensively evaluate the performance of the proposed system and compare it with that of known systems, i.e., LFU, LRU, and FIFO, over both synthetic and real 360^o video traces. The results reveal the large benefits coming from online caching of virtual viewports instead of the original ones in terms of the overall quality of the rendered viewports, the cache hit ratio and the backhaul usage.

The rest of the chapter is organized as follows. In Section 4.2, we describe our system setup. Next, in Section 4.3 we introduce the considered model of the users' requests. Then, we first formulate our problem as an MDP in Section 4.4, and right after in Section 4.5, we show how DQN can be used to solve the cache placement problem for 360^o videos. In Section 4.6, we thoroughly evaluate the performance of the proposed scheme and compare it with other methods in the literature. Finally, we draw conclusions in Section 4.7.

4.2 System Setup

In this section, we first introduce the network architecture, and the video library. Finally, we present the employed viewport prediction algorithm, and the considered end-to-end delay of the network.

1) Wireless cellular network

In this chapter, we consider a heterogeneous cellular network (HCN), like the one depicted in Fig. 4.1. Similarly to Chapter 3, this network consists of N SBSs with a cache capacity $C_n \geq 0,\ \forall n \in \mathcal{N}$, an MBS denoted by $N+1$, and U users. The communication range of the nth SBS is p_n, and the communication range of the MBS is p_{N+1}. Differently from Chapter 3, where SBSs could collaborate, in this chapter we assume that users located in the overlap of the coverage areas of multiple SBSs are associated with only one SBS, the one with the maximum SINR.

2) Video Library

We assume that users request 360^o video files from a content catalogue of V files, in a similar way as presented in Chapter 3. Specifically, each 360^o video is comprised of G GOPs, while each GOP is encoded into L quality layers and M tiles. However, compared to Chapter 3 where the caching decisions were made offline during the content placement phase, in this chapter the caching decisions are made online in a reactive manner.

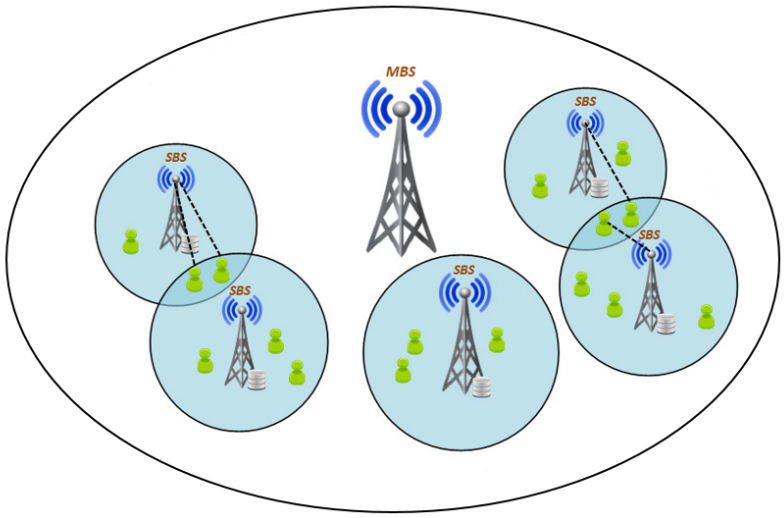

Figure 4.1: Considered network architecture. Users located in the overlap of the coverage areas of the SBSs are associated with the SBS with the maximum SINR, as shown with black dashed lines.

3) Viewport Prediction

A critical component of 360^o video streaming is the Viewport Prediction (VP) [111–113]. The aim of VP is to predict the requested viewport by a user in the *near* future (e.g., 1-2 sec), and prefetch it to the user. This is essential to provide smooth playback, as SBSs are not able to respond instantly to the user head movements due to the end-to-end delay.

VP can be done by observing the most recently requested frames by a user. These past requests are used to forecast the viewport that will be requested in the next few seconds. Such an approach is examined in [111, 112], where authors use variants of the linear regression algorithm to predict the users' head movements. A more naïve approach is presented in [113], where VP is performed assuming that the users' head orientation will not change in the next 3 seconds.

In our system, to perform viewport prediction, we use the Last Sample Replication (LSR) algorithm [112]. We have selected this algorithm because of its low

complexity. Based on the LSR, the predicted viewport of the GOP $g+1$ is assumed to be the same as the one that was requested in the GOP g. For the first GOP, without loss of generality, we assume that the predicted viewport is the requested viewport. Although the employment of advanced VP algorithms [50, 51] would further improve the accuracy of the predicted viewports, we do not adopt such algorithms as we aim to show the advantages coming from caching. Further, the employment of more advanced prediction algorithms would increase the complexity of our system. At the same time, the conclusions derived regarding the benefits of tile encoding and caching for 360^o videos would stay unaltered.

4) End-to-end-delay

As we already mentioned, for each GOP $g \in \mathcal{G}$, all the tiles encoded at the base quality along with all the enhancement layers up to the targeted quality for the tiles that form the output viewport of the VP algorithm, need to be prefetched to the users within a specific time window. Failing to deliver these tiles on time would lead to buffer underruns, as the tiles would not be available to the buffer at the time they should be displayed. This would lead to degraded QoE, as tiles that are not delivered on time are discarded. Considering the above, the timely delivery of the tiles of each GOP must respect the following equation:

$$\sum_{n \in \mathcal{N}_B} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} o_{vglm} \cdot d_{nu} \cdot q^{nu}_{vglm} \leq t_{disp}, \forall v \in \mathcal{V}, \forall u \in \mathcal{U}, g \in \mathcal{G} \tag{4.1}$$

where the variable q^{nu}_{vglm} takes the value 1 when the mth encoded tile of the lth quality layer of the gth GOP of the vth 360^o video is delivered to the uth user from the cache of the nth SBS ($n \in \mathcal{N}$) or the MBS ($n = N+1$), and 0 otherwise. Recall that d_{nu} is the time needed to transmit one Mbit from the nth SBS to the uth user, and $d_{(N+1)u}$ the time needed to transmit one Mbit from the backhaul

of the MBS to the uth user, as described in Chapter 3. In addition, o_{vglm} is the size of the mth tile encoded in the lth quality layer of the gth GOP of the vth 360^o video, and t_{disp} denotes the playback duration of each GOP. This constraint determines whether the tiles of the $(g+1)$th GOP can be prefetched to a user during the playback of the gth GOP.

4.3 Users' Requests Model and Cache Update Schedule

In this section, we present the considered users' requests model and the cache update schedule. We assume that for each cached 360^o video, our system caches all the tiles at the base quality layer for all the GOPs, as well as the tiles of a *virtual* viewport for each GOP in high quality. Recall that a viewport consists of k tiles that form a rectangular area, while a virtual viewport is comprised of the k most popular tiles, which do not necessarily form a rectangular area.

We assume that time is slotted in T time slots, and each time slot has the duration of one GOP. When a user is interested in watching a 360^o video with duration of G GOPs, they should send G consecutive requests,[1] as shown in Fig. 4.2. The first request is special and comprises a request w_0, which is used by our algorithm in Section 4.5 to predict the popularity of each 360^o video, and a request w_1 for obtaining the viewport for the first GOP. The request w_0 is to acquire the 360^o video at the base quality for all GOPs. As we will show in Section 4.5 this request is used to reduce the size of the optimization problem. Though all the tiles encoded at the base layer are requested at the first time slot,

[1]When a user wants to stop watching a video, they halt sending requests for the following GOPs.

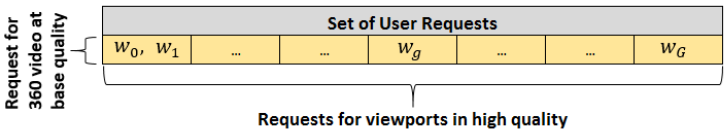

Figure 4.2: User requests for a 360^o video.

they may be delivered to the users along with the enhancement layer tiles. We assume that the request w_0 occurs at the same time slot with the request w_1. The w_gth request, $g \in \{1, \ldots, G\}$, is to obtain the tiles that comprise the requested viewport, and belong to the gth GOP, in high quality. These requests are used by our algorithm presented in the next sections to calculate the popularity of each tile per GOP. For notational convenience, we denote the ith set of requests $\{w_0^i, w_1^i, \ldots, w_G^i\}$ made by a user for a 360^o video by $\mathcal{W}^i$, while $\mathcal{W} = \cup_i \mathcal{W}^i$ contains all the sets of users' requests. Hereafter, we drop the index of the ith set of requests when it is not needed.

The decisions of which tiles of a 360^o video to cache at an SBS and in what quality are made online, i.e., when the content is requested. Specifically, when a user request w_0 arrives at an SBS in the time slot 1, if the tiles of the requested 360^o video at base quality are not cached in it, a decision has to be made regarding whether to cache them. This decision depends on the popularity of the video. If the decision is to cache these tiles, all the tiles of the base layer for all GOPs will start being fetched through the backhaul and cached at the SBS, replacing the tiles of another 360^o video that will be evicted. When the user issues further requests w_g for receiving tiles of the GOPs of the 360^o video in high quality, our system uses the viewport prediction algorithm described in Section 4.2 to decide which tiles of the predicted viewport will be fetched from the backhaul so that they can be delivered to the user on time. When the tiles

that form the predicted viewport arrive at the SBS, we identify two cases: (a) if the decision for w_0 was not to cache the 360^o video in base quality, the fetched tiles will be delivered to the user but these tiles will not be cached at the SBS, as the video is not popular enough, (b) if the decision for w_0 was to cache the 360^o video in base quality, then for each request $w_g, g \in G$, in case some (or none) of the tiles that form the predicted viewport are cached in high quality a soft cache hit [109] will occur. In such a case, the cached tiles of the predicted viewport will be served to the user directly from the SBS, while the tiles of the predicted viewport that are not cached at the SBS will be fetched to the SBS from a remote content server through the backhaul link of the MBS and be delivered to the user if the end-to-end constraint permits. Then, a decision is made about whether to cache some or all of the tiles that were fetched through the backhaul. The latter decisions reflects tiles' popularity in a 360^o video.

The proposed cache optimization algorithm regarding which tiles to cache is presented in the next sections.

4.4 MDP Formulation

In this section, we formulate the problem of caching 360^o videos in cellular networks as a Markov Decision Process [114]. Since in our setting users can download the requested content only from the SBS that they are connected to, each SBS optimizes the cache use and the content replacement strategy independently of each other. Hereafter, following reinforcement learning terminology, SBSs are also called agents. The formulated MDP consists of a number of states, where each state is comprised by features extracted from past requests. At any given state, the agents takes a specific action, i.e., to cache a requested content in place

of another, and receives a reward. This process is shown in Fig. 4.3

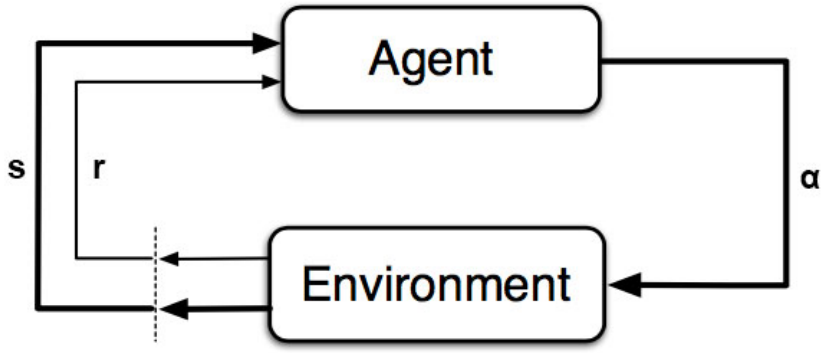

Figure 4.3: Markov Decision Process.

State Space: In the considered setting, the SBS $n \in \mathcal{N}$ can be in a state $s \in \mathcal{S}$, where $\mathcal{S}$ represents the set of all possible states. Each state s is a vector comprised of three features. Each feature is extracted from past users' requests, considering two time windows: a short one and a longer one. The aforementioned features as well as the considered time windows are discussed in detail below.

The *first feature* has two components that refer to the total number of requests for each cached 360^o video that occurred in: a) a short-term window of H_s sets of user requests (see Fig. 4.2), and b) a long-term window of H_l sets of user requests. This feature associated with the cache of SBS $n \in \mathcal{N}$ can be described by the vector $\mathbf{x}^n = [\mathbf{x}_s^n \ \mathbf{x}_l^n]$ with $\mathbf{x}_f^n = [x_{f,i}^n]$, $\forall f \in \{s, l\}$, $\forall i \in 1, \ldots, C$, and $x_{f,i}^n \in \{1, \ldots, H_f\}$. $x_{f,i}^n$ refers to the total number of times the video in the ith cache position was requested (either in short-term or long-term). Thus, the feature space $\mathcal{X}_f^n$ is given by $\{1, \ldots, H_f\}^C$ and the overall feature space is $\mathcal{X}^n = \mathcal{X}_s^n \times \mathcal{X}_l^n$. C is the cache capacity of the SBSs, where we dropped the subscript for notational convenience. It is worth noting that the above definition of features reduces the feature space drastically, as features are computed for all the tiles (cached videos) in base quality instead of each tile in base quality

independently.

Similarly, the *second feature* has two components that correspond to the total number of requests for tiles in high quality of the cached 360^o videos that happened during: a) the short-term window of H_s sets of user requests, and b) the long-term window of H_l sets of user requests. This feature is associated with the cache of SBS $n \in \mathcal{N}$ and is computed for each cached tile in high quality of GOP g, when request $w_g,\ g > 0$ is processed. Let the vector $\mathbf{y}^n = [\mathbf{y}_s^n\ \mathbf{y}_l^n]$ describe this feature, where $\mathbf{y}_f^n = [y_{f,i,j}^n]$, $\forall f \in \{s, l\}$, $\forall j \in \{1, \ldots, k\}$, $\forall i \in \{1, \ldots, C\}$ and $y_{f,i,j}^n \in \{1, \ldots, H_f\}$. $y_{f,i,j}^n$ denotes the number of times the jth tile of the ith cached 360^o video was requested at the nth SBS. Thus, the feature space $\mathcal{Y}_f^n$ is given by $\{1, \ldots, H_f\}^{kC}$ and the overall feature space for the cache space at the nth SBS is given by $\mathcal{Y}^n = \mathcal{Y}_s^n \times \mathcal{Y}_l^n$.

Finally, the *third feature* has two components that correspond to the number of times the examined item (tile in high quality or 360^o video in base quality) was requested at the nth SBS: a) in the short-term window of H_s sets of user requests, and b) in the long-term window of H_l sets of user requests. Specifically, when the examined item is a 360^o video in base quality, this feature refers to the total number of times this video was requested at the nth SBS. This is the case when a request w_0 is received. When the examined item is a tile of a 360^o video in high quality, i.e. for requests $w_g,\ g > 0$, the feature corresponds to the total number of times the examined tile was requested. The feature vector is defined as $\mathbf{z}^n = [\mathbf{z}_s^n\ \mathbf{z}_l^n]$ with $\mathbf{z}_f^n = [z_f^n]$, $\forall f \in \{s, l\}$, and $\forall z_f^n \in \{1, \ldots, H_f\}$. z_f^n stands for the total number of times the item (tile in high quality or 360^o video in base quality) was requested. Thus, the feature space $\mathcal{Z}_f^n$ is given by $\{1, \ldots, H_f\}$ and the overall feature space for the examined item is $\mathcal{Z}^n = \mathcal{Z}_s^n \times \mathcal{Z}_l^n$.

Following the above definitions of the features, the overall state space is given

by:

$$\mathcal{S}^n = \mathcal{X}^n \times \mathcal{Y}^n \times \mathcal{Z}^n. \tag{4.2}$$

Hereafter, we drop the superscript of the state space and use $\mathcal{S}$ as each SBS makes decisions independently of each other.

Action Space: As we mentioned in Section 4.3, users' requests w_0 correspond to a request for a 360^o video in base quality for all the GOPs of this video, while requests $w_g \in \mathcal{W}$ with $g \in 1, \ldots, G$ stand for a request for a viewport of the gth GOP encoded in high quality.

When an SBS receives a request from a user, given the current state s, it takes an action a which determines which content item to evict, to cache the requested content in that place. When a request is received at an SBS, there are three possible cases regarding the status of the cached data at that SBS: a) no data for the requested 360^o video is cached, b) the 360^o video is cached at the base quality and the predicted viewport is cached at high quality and, c) the 360^o video is cached at the base quality, but a different viewport is cached at high quality.

In case a user requests a 360^o video that is not cached at the SBS, this has to be fetched through the backhaul and be delivered to the user. Fetching content through the backhaul adds cost to the network operator and increases the delay experienced by the users. When no data of a 360^o video are cached at the SBS, a user request $w_g \in \mathcal{W}$ with $g \in \{0, \ldots, G\}$ is processed as follows. To accommodate a request w_0, all the tiles of the requested 360^o video encoded at the base quality will start being fetched through the backhaul and delivered to the user for all the GOPs. For each of the following requests w_g with $g \in \{1, \ldots, G\}$,

all the tiles of the viewport indicated by the prediction algorithm in Section 4.2 encoded in high quality will be fetched through the backhaul in high quality, and be delivered to the user. Therefore, when the requested 360^o video is not cached at the SBS, there are two types of possible actions: a) to leave the cached content at the SBS unchanged, or b) to evict the tiles of a cached 360^o video from the cache of the SBS and replace them with the tiles of the requested one. Thus, there are $C + 1$ possible actions. Let the set $\mathcal{A}_1 = \{A_{10}, A_{11} \ldots, A_{1i}, \ldots, A_{1C}\}$ denotes all the possible actions when a video is not cached at the SBS. A_{10} stands for the case the cached content at the SBS is left unchanged, and A_{1i} means that all the tiles of the ith cached video at the SBS will be replaced by the corresponding tiles of the requested 360^o video.

If both the requested 360^o video encoded in the base quality and the tiles that form the predicted viewport for the examined GOP, e.g., $g \in \{1, \ldots, G\}$ encoded in high quality are cached at the SBS, the request w_g will be served from the cache, and no action will be taken. Then, a decision regarding whether to cache the tiles of the predicted viewport for the next GOP, i.e., $g+1$, is made. This happens because our scheme employs the LSR algorithm, as we described in Section 4.2.

Finally, if the 360^o video is cached at the base quality, but a different viewport than the predicted one is cached at the SBS at high quality, the requested tiles that are not cached have to be fetched through the backhaul, and then be served to the user. In that case, the possible actions are the following: a) to leave the cached viewport unchanged, or b) to cache some of the tiles, which were not part of the predicted viewport, and were fetched through the backhaul. To limit the action space, we assume that each action concerns only one tile that may be updated at the SBS cache. In this way, the agent takes sequential actions for all

tiles that were fetched through the backhaul in terms of whether to cache them at the SBS or not. This process is repeated until a decision is made for all the fetched tiles. Since each viewport consists of k tiles, the possible actions for a tile form the set $\mathcal{A}_2 = \{A_{20}, A_{21} \ldots, A_{2j}, \ldots, A_{2k}\}$. The action A_{20} denotes the case where the cached content is left unchanged, while the action A_{2j} corresponds to the case where the candidate tile will replace the jth tile in high quality of the requested 360^o video that was cached at the SBS. We consider that a GOP is fully processed when a decision has been made for all the tiles that were fetched through the backhaul. After completing the sequential decisions, the cached virtual viewport for the considered video is updated. Next, the subsequent GOP is processed in a similar way. We would like to note that the use of virtual viewports and the decomposition of actions on a per tile basis permits to greatly reduce the action space as, otherwise the action space would have been comprised of all possible viewports.

Considering the above, the overall action space $\mathcal{A}$ is defined as:

$$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2. \tag{4.3}$$

Reward: We define the reward of each action to be the average distortion reduction the users will experience in the next H sets of users' requests. Thus, given a state $s \in \mathcal{S}$, the reward of taking action $a \in \mathcal{A}$ is calculated as:

$$r(s,a) = \frac{1}{H} \sum_{h \in \mathcal{H}} \sum_{v \in \mathcal{V}} \sum_{g \in \mathcal{G}} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} \mathbb{1}(\phi_{hvglm}) \cdot \delta_{vglm} \tag{4.4}$$

When we process the ith set of requests $\mathcal{W}^i$, the set $\mathcal{H}$ contains the next H sets of user requests. In our formulation, the reward in (4.4) is obtained after

the next H sets of user requests have occurred [115]. The term ϕ_{hvglm} represents the mth tile of the lth quality layer of the gth GOP of the vth 360^o video of the $\mathcal{W}^{i+h}$th set of user requests. Recall from Chapter 3 that δ_{vglm} denotes the distortion reduction achieved by obtaining the corresponding tile. The indicator function $\mathbb{1}(\phi_{hvglm})$ in (4.4) is defined as:

$$\mathbb{1}(\phi_{hvglm}) = \begin{cases} 1, & \text{if } \phi_{hvglm} \text{ can be delivered on time} \\ 0, & \text{if } \phi_{hvglm} \text{ cannot be delivered on time} \end{cases}$$

Optimization Problem:

In order to quantify how good a particular state s is, we estimate the value function. This function corresponds to the expected discounted reward of policy π when starting from a state s and then following this policy. The value function is formally expressed as:

$$V_\pi(s) = E_\pi[G_\tau | S_\tau = s] = E_\pi[\sum_{\kappa=0}^{\infty} \gamma^\tau R_{\tau+\kappa+1} | S_\tau = s] \tag{4.5}$$

where G_τ, R_τ, and S_τ are the expected reward, the immediate reward and the state at time τ, respectively. The parameter $0 \leq \gamma \leq 1$ is called discount rate and gradually discounts the effect of an action to future rewards. If $\gamma = 0$, the agent is "myopic" and maximizes the immediate reward. As γ approaches 1, the objective takes into account future rewards more strongly, and the agent becomes farsighted. The above equation can be rewritten as a Bellman equation [116] as follows:

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V_\pi(s')] \tag{4.6}$$

where $p(s', r|s, a)$ is the transition probability from the state s to the state s' by

taking the action a with a reward r.

4.5 DQN based cache optimization

The main challenge to solve (4.6) is the requirement to know the transition probabilities $p(s', r|s, a)$. For the studied problem, continuous computation of the transition probability matrix is necessary because of the dynamics of the non-stationary requests, which is computationally demanding. To overcome this problem, we can adopt the Q-learning algorithm [92], which learns the optimal policy through interaction with the environment. Q-learning uses the $Q(s, a)$ values instead of using the value function in (4.6). These values reflect how "good" is it to take action a when in state s. Similarly, $Q_\pi(s, a)$ represents how good it is to take action a when starting from state s, and thereafter follow the policy π. This is defined as follows:

$$Q_\pi(s, a) = E_\pi[\sum_{k=0}^{\infty} \gamma^\tau R_{\tau+k+1}|S_\tau = s, A_\tau = a] \tag{4.7}$$

where A_τ is the action at time τ.

The optimal policy is the one that maximizes the expected reward for all states and is given by:

$$\pi^\star(s) = \arg\max_{a\in\mathcal{A}}(Q(s, a)), s \in \mathcal{S} \tag{4.8}$$

To determine the optimal policy $\pi^\star(s)$, the Q-learning algorithm updates the $Q(s, a)$ values iteratively. Specifically, the $Q(s, a)$ values are updated according

to the formula:

$$Q(s_\tau, a_\tau) \leftarrow (1-\alpha_\tau)Q(s_\tau, a_\tau) + \alpha_\tau[R_\tau + \gamma \max_{a\in\mathcal{A}} Q(s_{\tau+1}, a)] \tag{4.9}$$

where α_τ is the learning rate at time τ. The learning rate corresponds to the rate at which newly acquired information overrides old information.

Q-learning can select actions using policies such as the ϵ-greedy, where $\epsilon \in [0, 1]$, which ensures that random actions are always explored and overfitting is avoided. According to the ϵ-greedy policy, the action resulting in the maximum $Q(s_\tau, a_\tau)$ value is selected with probability $1-\epsilon$, and a random action is selected with probability ϵ. The Q-learning algorithm is guaranteed to converge to the optimal solution [117] when all the state-action pairs are visited infinitely often, and the learning rate α_τ satisfies the following conditions:

$$\sum_{\tau=0}^{\infty} \alpha_\tau(s, a) = \infty \quad \text{and} \quad \sum_{\tau=0}^{\infty} \alpha_\tau^2(s, a) < \infty, \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A} \tag{4.10}$$

The Q-learning algorithm is an efficient method to determine the optimal policy when the state-action space is small. However, when the state-action space grows, the lookup table where the $Q(s, a)$ values are stored becomes prohibitively large. To overcome this drawback of Q-learning, we employ a Deep Reinforcement Learning (DRL) [118] approach. Using DLR the $Q(s, a)$ values are approximated by a Deep Neural Network (DNN). Specifically, the employed DNN takes as an input the current state s of the SBS, which is given as a vector comprised of the three features (both in the short term, and in the long term) described in Section 4.4. Then, it outputs a vector of $Q(s, a)$ values which indicate how "good" is to take each of the possible actions a, given the current state s. The DRL framework consists of two phases: a) the offline phase where the DNN is trained, and b) the online phase during which the actual caching

decisions are made.

During the offline phase, the DNN is initially built by selecting some random weights θ. Then, the DNN is trained with a number of historic transition profiles, as in [119]. These profiles correspond to request patterns experienced in the past. The training of the DNN is performed in a mini-batch manner. Specifically, at each training epoch, a sample of the transition profiles and their estimated Q values are obtained by randomly sampling the experience replay memory D, which has capacity N_D. This mechanism is used to remove the correlations between observations, while the transitions between the states become more independent and identically distributed.

To stabilize DNN training, apart from the experience replay, we use the mechanism of the fixed target network [117]. According to this mechanism, a second DNN is employed, which is called the fixed-target network. This network has the same architecture as the original DNN that is used for the function approximation (evaluation network). Not using a separate network to estimate the target Q values would lead to destabilization. This would happen because as the Q values (output of the evaluation network) are updated towards the target values (calculated by (4.9)), the target values will also be updated in the same direction. To overcome this problem, the weight parameters of the target network are kept fixed and are copied from the evaluation network only every N_T steps. Thus, using a second network to estimate the target Q-values leads to a more stable training, since the Q-values obtained from the evaluation network are updated towards a target that is kept fixed (for a number of steps).

When the offline phase is completed, the obtained weights θ are used to initialize the DNN in the online phase. During this phase, if the candidate item (360^o video in base quality or tile in high quality) is not cached at an SBS, the

agent takes an action according to the ϵ-greedy policy (i.e., it decides whether to cache the item or not and what content will be replaced), and then proceeds to the next state. In this way, new actions are always explored, and cached content whose popularity the algorithm overestimated in the past will not stay in the cache forever. After the execution of each action, the tuple $(s_\tau, a_\tau, r_\tau, s_{\tau+1})$ is stored in the experience replay buffer D, in order to be used later for the training of the DNN.

In the online phase, the DNN is trained in a similar way to the offline phase, where a batch of M_B transition profiles is randomly sampled from the experience replay memory D every N_B steps. The DNN is trained towards the target Q values using the back-propagation method, by minimizing the loss function $Loss(\theta)$. The loss function is given by:

$$Loss(\theta) = \frac{1}{M_B} \sum_{i \in \{1,\ldots,M_B\}} (y_i - Q(s_i, a_i, \theta))^2 \tag{4.11}$$

where $y_i = r_i + \max_{a'_i} Q(s'_i, a'_i, \theta_i^-)$ represents the target Q value of the ith sample, and $\theta_i^- = \theta_{i-N_B}$.

The overall DRL framework is presented in Algorithm 4.1.

Algorithm 4.1 DRL Framework

Offline Phase
Initialize the evaluation network with weights θ
Initialize the fixed target network with weights θ'
Initialize the experience buffer D with capacity N_D
Initialize a random exploration process
Train the DNN with features (s, a) and outcomes $Q(s, a)$ in a mini-batch manner
Online Phase
for each time slot **do**
  for each user request in a time slot **do**
    for each candidate item of a user request **do**
      Receive observation s_τ
      if the candidate item is not cached at the SBS **then**
        With probability $1 - \epsilon$ **select**
          $a_\tau = \arg\max_{a \in \mathcal{A}} Q(s_\tau, a, \theta)$
        Otherwise,
          $a_\tau \leftarrow$ random action
        Take action a_τ and observe r_τ, $s_{\tau+1}$
        Store the tuple $(s_\tau, a_\tau, r_\tau, s_{\tau+1})$ in the experience replay buffer D
      end if
      Update cache hit ratio
      Update Feature Space
      if Modulo(w, N_B)==0 **then**
        Sample M_B tuples from D
        Update DNN by minimizing $Loss(\theta)$ in (4.11)
        Update fixed target network weights
      end if
    end for
  end for
end for

4.6 Performance Evaluation

In this section, we examine the performance of the proposed DQN-based online caching algorithm for 360^o videos in cellular networks. First, we describe the schemes under comparison and provide the simulation setup. Next, we show the

convergence of the loss function during the training of the DNN. Then, we analyze the impact of various system parameters on the performance of the system. Finally, we demonstrate how the viewports' popularity shapes the popularity of each tile.

4.6.1 Simulation Setup

Let us describe the main characteristics of the schemes under comparison and the proposed scheme:

1. *Least Frequently Used (LFU):* In this scheme, the network operator keeps track of the number of requests that occurred for each cached 360^o video. When a user request arrives at an SBS, then: a) if the requested 360^o video is not cached at it, all the tiles of the 360^o video that was requested the least number of times will be evicted from the cache of the SBS. Then, for all the GOPs, all the tiles of the requested 360^o video encoded at the base layer along with the tiles of the predicted viewport in high quality will be cached at the SBS; b) if the 360^o video is already cached at the base quality for all the GOPs, but some of the cached tiles in high quality are different from the ones that belong to the predicted viewport, these tiles will be evicted and be replaced by the tiles of the predicted viewport.

2. *Least Recently Used (LRU):* In this scheme, the network operator keeps track of how recent are the requests that occurred for each cached 360^o video. When a user request happens at an SBS, then: a) if the requested 360^o video is not cached at the SBS, all the tiles of the 360^o video that were requested the least recently will be evicted from the SBS cache. Next, all the tiles of the requested 360^o video will be cached at the SBS at the

base quality for all GOPs along with the tiles of the predicted viewport in high quality; b) if the 360^o video is cached at the SBS, for each GOP, if some of the cached tiles in high quality are different from the ones of the predicted viewport, these tiles will be replaced by the corresponding tiles of the predicted viewport.

3. *First In First Out (FIFO):* In this scheme, the network operator keeps track of when the requests for each cached 360^o video occurred. When a user request arrives at an SBS, then: a) if the requested 360^o video is not cached at the SBS, all the tiles of the 360^o video that was cached the earliest will be evicted from the SBS. Then, for all GOPs, all the tiles of the requested 360^o video encoded at the base layer, along with, for each GOP, the tiles of the predicted viewport in high quality will be cached at the SBS in the place of the evicted tiles; b) if the 360^o video is cached at the SBS, then for each GOP, if some of the cached tiles in high quality are different from the ones forming the predicted viewport, these tiles will be evicted, and be replaced by the tiles of the predicted viewport.

4. *Proposed Scheme:* In the proposed scheme, the caching decisions are made exploiting observations derived from past users' requests. This scheme employs the DQN algorithm presented in Section 4.5 to decide on the cache updates. For each cached 360^o video, all the tiles at the base quality along with the most popular tiles in high quality that form a virtual viewport, are cached at the SBS for all the GOPs.

For the sake of simplicity, all the conducted experiments are done assuming a single SBS and an MBS. This does not affect the derived conclusions, as SBSs make caching decisions independently of each other. As we have already

mentioned in Section 4.2, although SBSs' coverage area may overlap, users are assigned to a single SBS, i.e., the one with the maximum SINR. The exploitation of opportunities arising because of the overlapped coverage areas is part of our future work. We would like to emphasize that our algorithm can be applied to networks with an arbitrary number of SBSs. This is because as each user is assigned to a single SBS, our algorithm can run in parallel for each SBS. The coverage range of the SBS is set to be $p_n = 300$m, while the coverage range of the MBS is $p_{N+1} = 2000$m, and is large enough to permit the communication with the SBS. The delay needed to obtain one Mbit from the SBS is $d_{nu} = 1/14$ sec/Mbit, while the delay to deliver one Mbit from the backhaul of the MBS to the user is $d_{(N+1)u} = 1/2.9$ sec/Mbit. The cache capacity of the SBS is set to be enough to store 10% of the 360^o videos of the content library. The number of users is $U = 200$ who are randomly placed in the coverage area of the SBSs. Recall that, when a 360^o video is cached at the SBS, this means that for each GOP, all the tiles are cached at the base quality, and the tiles that form a virtual viewport are cached in high quality.

The content library contains $V = 500$ videos, while each video is encoded in 30 GOPs. The duration of each GOP is assumed to be $t_{disp} = 1$ sec. Each GOP is encoded into $M = 12$ tiles, where each tile is encoded into $L = 2$ quality layers. The bitrate of the base layer is 2 Mbps, while the bitrate of the enhancement layer is 12 Mbps. The size of each viewport consists of 4 tiles, while the available viewports are the ones depicted in Fig. 4.4. The distortion reduction achieved by obtaining a tile at the base quality layer is 30 dB, while the distortion reduction achieved by receiving a tile at the enhancement quality layer is 10 dB. Similarly with Chapter 3, the probability of a 360^o video to be requested from a user follows the Zipfian distribution [108], as is common to the literature. The shape

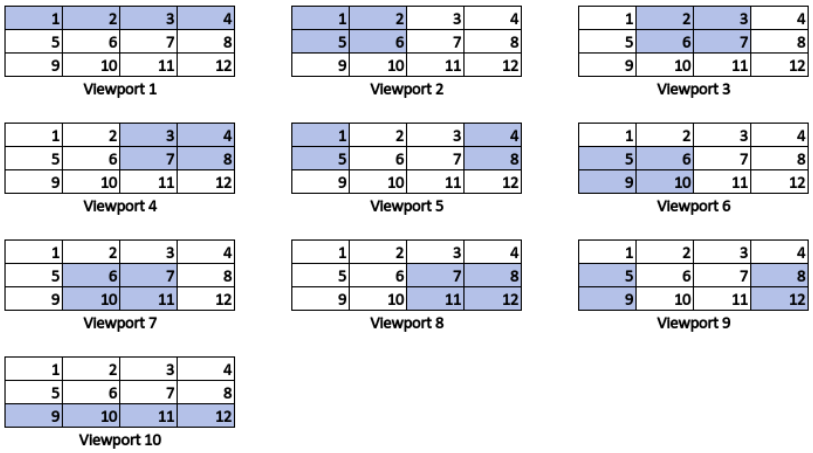

Figure 4.4: Considered set of viewports. The light blue area highlight the area covered by the viewport.

parameter of the Zipfian distribution is set to $\eta_v = 1$. Recall that the probability of a 360^o video $v \in \mathcal{V}$ to be selected under the Zipfian distribution is given by the Eq. (3.31).

We consider realistic navigation patterns, extracted from the dataset in [46], from which we sampled 200 trajectories of head movements. These trajectories are obtained from 10 different videos, where for each video, we sampled 20 different trajectories. With equal probability, we mapped the index of each one of the $V = 500$ videos from the content library to one of the 10 sampled videos of the dataset. Then, for each of the $V = 500$ videos of the content library, according to its mapped index, we selected one of the 20 available trajectories uniformly at random.

We assume that the total number of sets of users' requests is $W = 10000$. The short-term time window refers to $H_s = 300$ sets of user requests,[2] while the long-term time window corresponds to $H_l = 1000$ sets of user requests. The reward in (4.4) is calculated for the next $H = 1000$ sets of user requests.

[2]Each set of requests corresponds to the tiles of a single video demanded by a user.

4.6.2 Deep Neural Network Training

We consider a Deep Neural Network (DNN), which consists of four fully connected layers, i.e., the input layer, two hidden layers, and the output layer. As the cache capacity of our system is C, the input layer consists of $10C + 2$ nodes that reflect the vector size of each state. The hidden layers and the output layer consist of $5C+1$ nodes, as there are $5C+1$ total actions. The activation function of the hidden layers is the ReLu, while the activation function of the output layer is the linear function. The DNN is trained with the Adam optimizer. The DNN is trained for 100 epochs in order to become sufficiently accurate. The learning rate is set to be $\alpha = 0.001$, while the ϵ-greedy parameter is set to $\epsilon = 0.05$. The discount factor is set to be $\gamma = 0.6$. The experience replay buffer is set to be $D = 2000$, while the mini-batch size is set to $M_B = 32$. The mini-batch samples are obtained every $N_B = 200$ requests. The convergence of the loss function during the training phase for the basic scenario is presented in Fig. 4.5. When the DNN is trained with different system settings than the ones of the basic scenario, a similar convergence behavior is noticed. For more information regarding ANNs, we refer the interested reader in *Appendix B*.

4.6.3 System Parameter Analysis

1) Cache Size

First, we examine the impact of the cache size on the overall quality of the rendered viewports. To this aim, we vary the cache capacity C in the range [5,25]% of the size of the content library. As we can see in Fig. 4.6, the proposed scheme outperforms the LFU, LRU and FIFO schemes, for all cache sizes. In

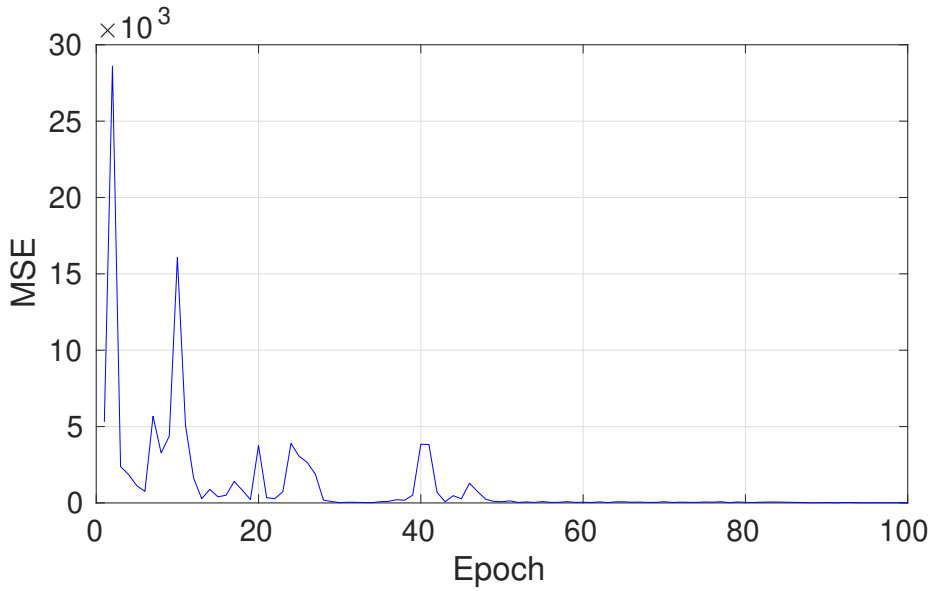

Figure 4.5: MSE of the loss function with respect to the training epochs.

particular, for typical cache capacity sizes, i.e., [5-10]%, the performance gap between the proposed scheme and the LFU, the LRU and the FIFO is about 1 dB, 1.5 dB, and 2 dB, respectively. This is because the proposed scheme achieves a better cache hit ratio, as shown in Fig. 4.7. The increased cache hit ratio of the proposed scheme is attributed to the use of the DQN that learns from the experience of the past observations, which content should be cached. In addition, unlike LFU, LRU and FIFO, where the cached tiles in high quality correspond to actual viewports, in the proposed algorithm, the tiles that will be cached for each 360^o video in high quality correspond to virtual viewports. This provides us with greater flexibility to decide the cached tiles. The effect of the increased cache hit ratio on the quality of the rendered viewports comes from the fact that the tiles that are delivered from the cache of the SBS to the users are delivered with a smaller delay. Hence, more tiles are delivered in total to the users under the considered tight time constraints. When the cache capacity is large, i.e., 25%, the performance gap between the proposed algorithm and the

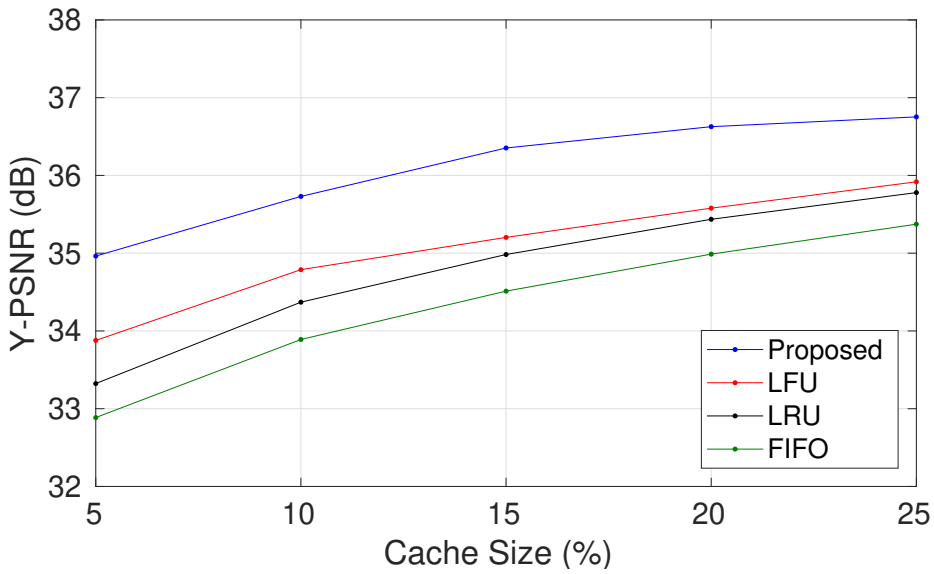

Figure 4.6: Y-PSNR with respect to the cache size for all the schemes under comparison.

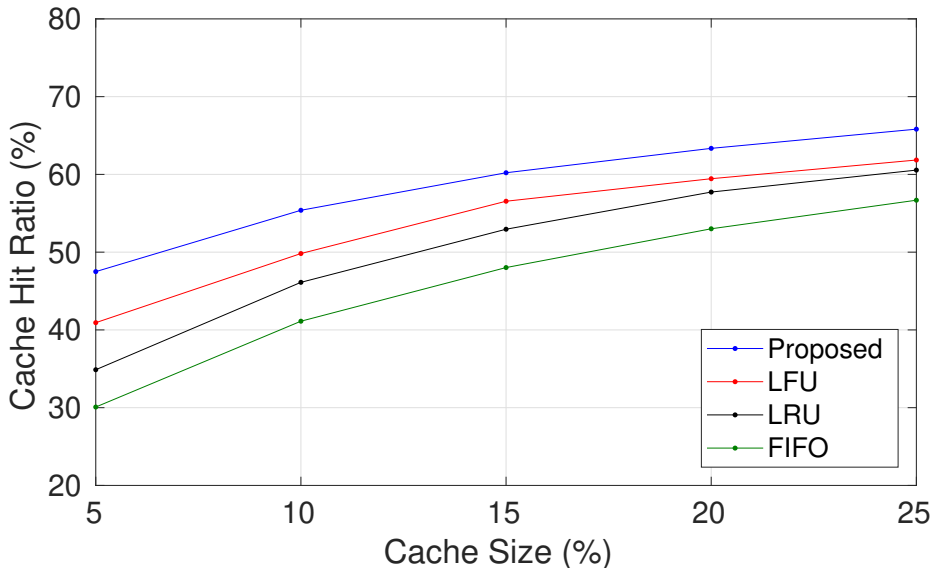

Figure 4.7: Cache Hit Ratio with respect to the cache size for all the schemes under comparison.

LFU, the LRU and the FIFO schemes closes to about 0.8 dB, 1 dB and 1.4 dB, respectively. This happens because as the cache capacity becomes larger, most of the popular content is stored in the SBS cache for all the schemes.

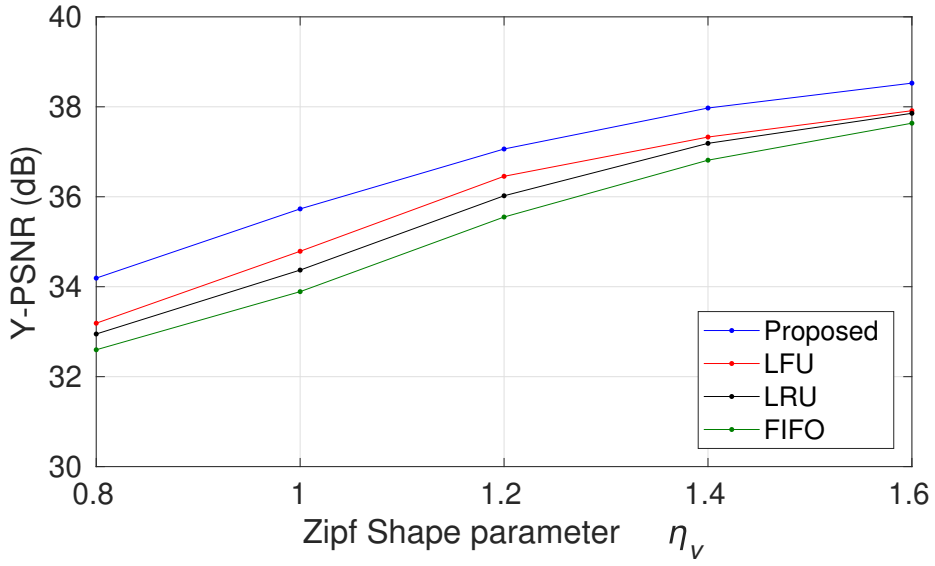

Figure 4.8: Y-PSNR with respect to the Zipf shape parameter of the 360^o videos for all schemes under comparison.

2) Video popularity distribution

In Fig. 4.8, we analyze the impact of the skewness parameter of the Zipfian distribution, which characterizes the 360^o video popularity. Specifically, we alter the shape parameter η_v in the range [0.8, 1.6] and measure the overall quality of the rendered viewports for all the schemes under comparison. We note that an increase in the value of the Zipf shape parameter η_v leads to an increase in the overall rendered quality for all the schemes. This is because bigger values of η_v mean that the video popularity distribution gets steeper, i.e., a smaller number of 360^o videos is popular, which increases the efficiency of the cache utilization. We can further observe that as the users' requests concern a smaller number of videos (big η_v values), the performance gap between the proposed algorithm and the LFU, the LRU, and the FIFO schemes decreases. For example, as the skewness parameter changes from 0.8 to 1.6, the performance gap between the proposed algorithm and the LFU decreases from $\sim$ 1 dB to $\sim$ 0.6 dB. This is

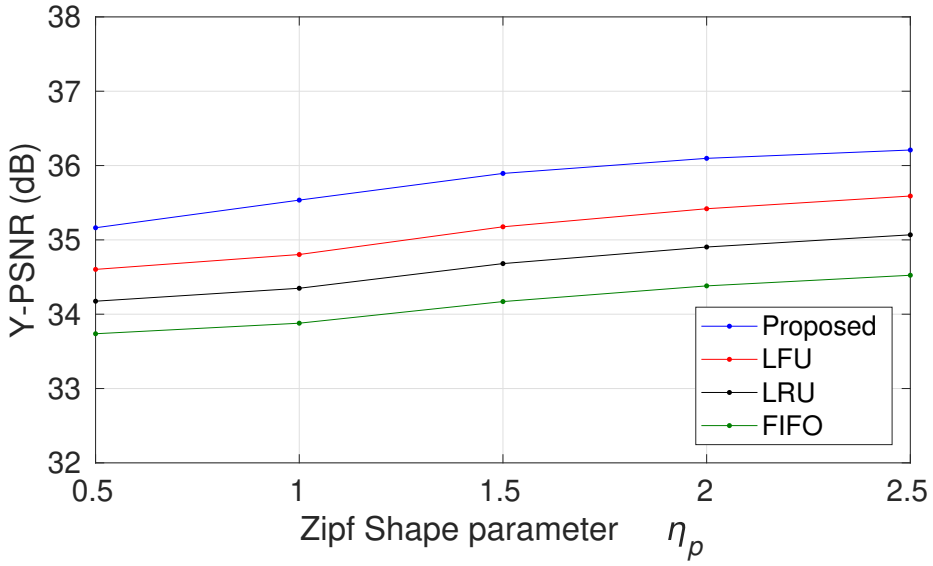

Figure 4.9: Y-PSNR with respect to the Zipf shape parameter of the viewports for all schemes under comparison.

attributed to the fact that as a smaller number of 360^o videos becomes popular, most of these videos will be cached at the SBS for all the schemes.

3) Viewports' popularity distribution

Besides video popularity, we examine the impact of viewports' popularity. We first assume that the viewports' popularity follows a Zipfian distribution with skewness parameter η_p. To analyze the impact of the skewness parameter on the quality of the rendered viewports, we vary the shape parameter η_p in the range [0.5, 2.5]. The performance of the schemes under comparison is depicted in Fig. 4.9. From the results, we can note that an increase in the skewness parameter η_p leads to an increase in the overall quality of the rendered viewports for all the examined schemes. This is because as the parameter η_p increases, the user requests for the various parts of the 360^o video scenes become less diverse. Thus, the cache effectiveness is improved. In addition, as the skewness parameter

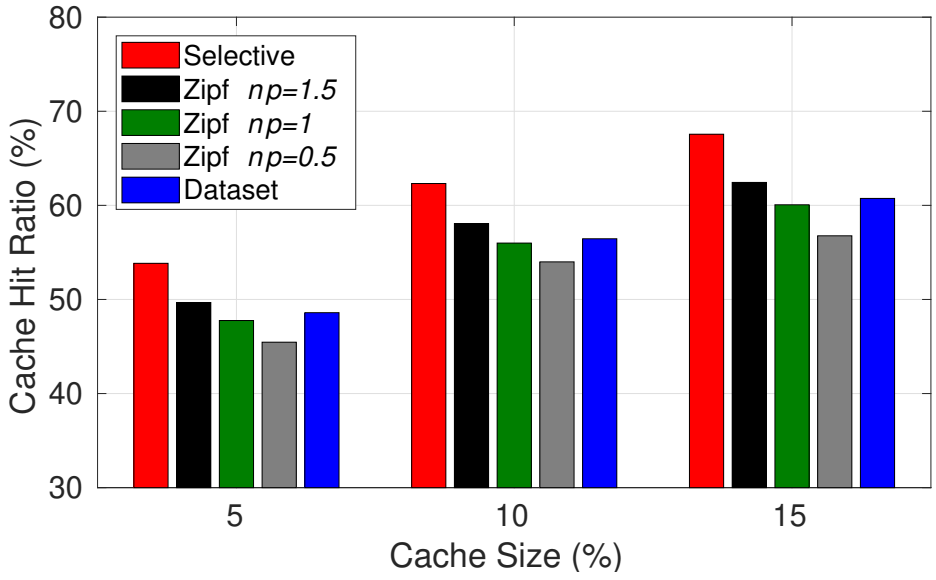

Figure 4.10: Cache hit ratio with respect to the cache size for the proposed scheme considering different viewport popularity distributions.

changes from 0.5 to 2.5, the performance gap between the proposed algorithm and the LFU increases from about 0.5 dB to about 0.65 dB, respectively. This is because unlike LFU, in the proposed algorithm, the caching decisions for the tiles that will be cached in high quality are made for virtual viewports, which offers increased flexibility in the caching decisions regarding which tiles to cache. Thus, as the requests for the various viewports become less diverse, the performance gains in the proposed algorithm increase. Similar conclusions can be drawn by comparing the proposed scheme with the LRU and FIFO schemes.

In Fig. 4.10, we evaluate the cache hit ratio of the proposed scheme for: a) our basic scenario where the requests for the viewports are according to the dataset [46], b) the case where the requests for the viewports follow the Zipfian distribution while the shape parameter η_p takes a value from the range [0.5, 1.5], and c) the case where all the user requests are for one viewport, which we term as "Selective". To this aim, we vary the cache size from 5% to 15% of the content

library. As we can observe, the "Selective" distribution achieves a better cache hit ratio in all cases. This is expected, as when the viewports follow either the dataset or the Zipfian distribution, the requests for the viewports are diverse, while in case of the Selective distribution, all requests are for one viewport. In addition, the cache hit ratio is better when the skewness parameter is higher as described above, while the performance of the dataset, is comparable with the case when the skewness parameter is $\eta_p = 1$.

4) Backhaul Usage

In Fig. 4.11, we compare the performance of all the schemes under comparison in terms of the backhaul usage. This is a very important performance indicator of the caching schemes since field trials [110] have shown that by reducing the backhaul usage, the network service cost is also reduced. To this end, we vary the cache size in the range [5, 25]% of the content library and measure the backhaul usage, in terms of the bandwidth that should be communicated to satisfy the demands. As expected, an increase in the cache size leads to a decrease in the backhaul usage for all cases. This is because as the cache size increases, more videos will be able to be stored at the SBS cache, thus, more content will be served locally to the users. In addition, we can note that as the cache size increases, the performance gap between the proposed method and the other schemes under comparison decreases. Specifically, as the cache size increases from 5% to 25%, the performance gap between the proposed method and the LFU decreases from about 15.6 GB to approximately 10.7 GB. This is because as the cache size increases, most of the requested content will be able to be cached at the SBS, and thus, the effectiveness of the caching improves for all schemes.

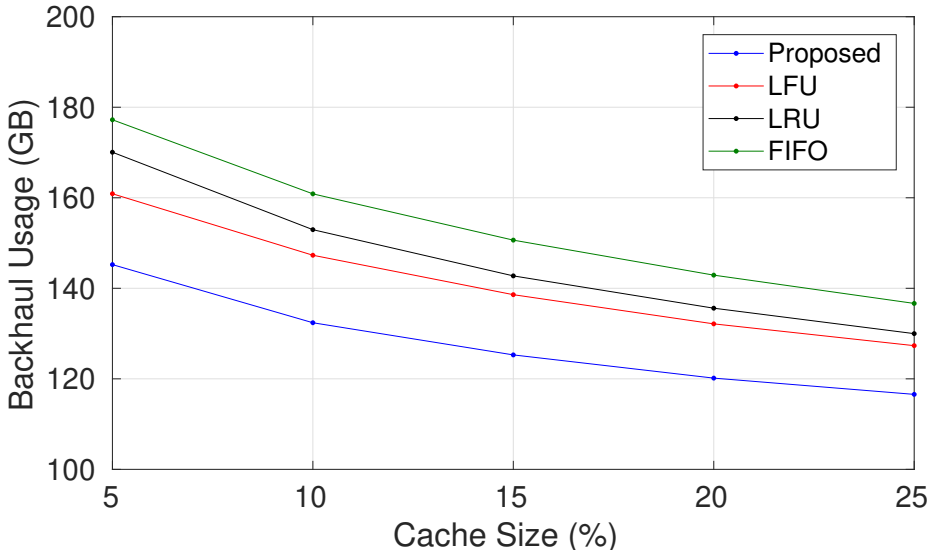

Figure 4.11: Backhaul usage with respect to the Cache Size for all schemes under comparison.

4.6.4 Overlap between Viewports

In this section, we present how the overlap between the various viewports shapes the popularity of each tile. To this aim, we examine the popularity of each viewport, along with the popularity of each tile. These popularities are computed by measuring the frequency of occurrence of a request w_g in a window of the previous $H_l = 1000$ sets of user requests. The popularity of each viewport is depicted in Fig. 4.12 and the popularity of each tile is depicted in Fig. 4.13. Although the most popular viewport is the viewport 8 (see the viewports illustrated in Fig. 4.4), by observing the Fig. 4.13, we can see that the most popular tiles do not correspond to the tiles of that viewport. The overlap between the diverse requests for the various viewports is what determines the popularity of each tile. Thus, by using *virtual* viewports, which consist of the most popular tiles, the most popular tiles can be cached at the SBS. This results in higher cache hit ratio and better quality for the rendered viewports.

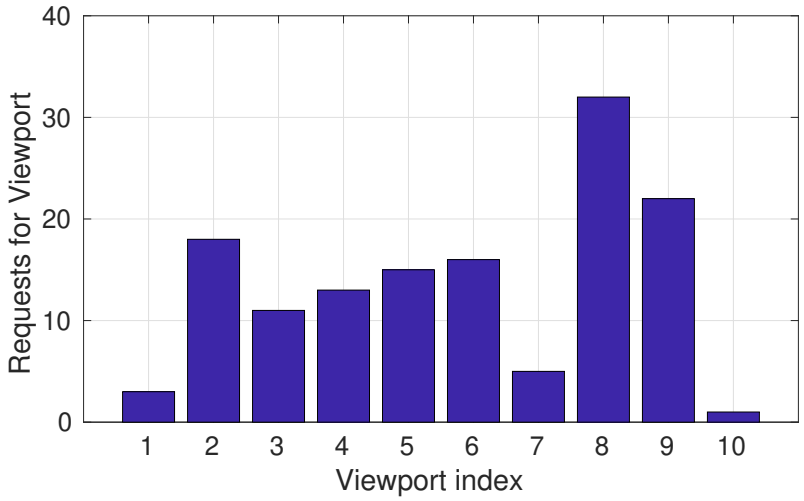

Figure 4.12: Total amount of requests for each one of the available viewports.

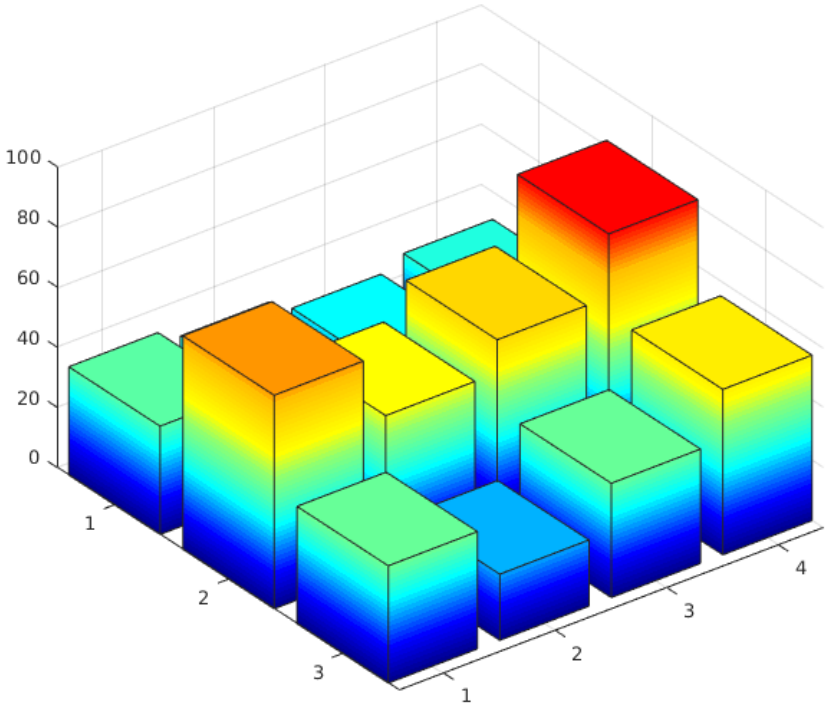

Figure 4.13: Total amount of requests for each one of the in high quality encoded tiles.

4.7 Conclusion

In this chapter, we studied the problem of delivering 360^o videos in mobile networks using edge caching, considering unknown 360^o video and viewport popularities. We formulated the caching placement/eviction problem as a Markov Decision Process that aimed at maximizing the overall quality of the videos delivered to the users. To deal with the dimensionality of this problem, we introduced the concept of virtual viewports, which have the same number of tiles with original viewports and are comprised by the most popular tiles. To solve the cache optimization problem for large state and action spaces, we employed a DQN solution that exploited the patterns obtained from the observations in the sequence of users' requests. These observations were used to train our system to learn for each state, which cache update action should be taken. In this way, we were able to cache the 360^o videos that were predicted to be the most popular, along with for each GOP, a virtual viewport. To evaluate our method, we used both real and synthetic navigation patterns. We compared our proposed method with the LFU, LRU, and FIFO schemes. The results showed that the proposed method outperformed its counterparts significantly. The improved performance of our method is attributed to the exploitation of the observations of the users' requests by the DQN, and the increased flexibility in the caching decisions that came from using virtual viewports.

5

A Tile-based caching framework for 360^o live video streaming

5.1 Introduction

In the previous two chapters, we showed the benefits of using edge caching for the delivery of 360^o videos. However, these systems are Video on Demand (VOD) solutions and cannot be trivially used for 360^o live video streaming. This is because the network traffic related to 360^o live video streaming differs from that of VOD systems, as in the former case multiple requests take place for the same content simultaneously. Motivated by the above, in this chapter we propose a novel caching framework for 360^o live video streaming. Our framework aims

to determine the optimal cache placement/evictions strategies to optimize the quality of the delivered video to the users. To this aim, SBSs update their cached content using LSTM networks, which have been be shown to be efficient for time series forecasting. To enhance the delivered video quality and reduce the service cost, users located in the overlap of the coverage areas of multiple SBSs may be associated with any of these SBSs at different SBS delays, from where they can receive their data. We evaluate and compare the performance of our algorithm with the LFU, LRU, and FIFO algorithms, which are commonly used in the literature. The results show the superiority of the proposed method against its counterparts, and make clear the benefits of users association with multiple SBSs in terms of the delivered quality.

The rest of this chapter is organized as follows. In Section 5.2 we describe the system setup. Next, in Section 5.3 we provide the system model. Afterwards, in Section 5.4 we evaluate the performance of the proposed scheme. Finally, we present the drawn conclusions in Section 5.5.

5.2 System Setup

In this section, we first introduce the considered live streaming architecture. Then, we discuss the transcoding of 360^o videos so that they have multiple quality layers and tiles. Finally, we describe the considered mobile edge network architecture, the users' requests model and the end-to-end delay of the mobile edge network.

1) Live Streaming Architecture

A high level representation of the considered live 360^o video streaming architecture is depicted in Fig. 5.1. We assume multiple users (broadcasters) that capture a 360^o FoV of a scene, using omnidirectional cameras. The captured 360^o videos are first transmitted to the Live Stream server using the Real-Time Messaging Protocol (RTMP) [120]. RTMP is selected as it can ensure low end-to-end latency between the broadcaster and viewers. The Live Stream server transcodes the 360^o videos so that they consist of multiple quality layers and tiles. This is because the captured 360^o videos from the broadcasters may not be necessarily encoded in that format. The transcoded video streams are transmitted to the Content Delivery Network (CDN) using HTTP. The mobile edge servers are populated with content from the CDN according to the proposed cache optimization algorithm, which will be presented in Section 5.3. We would like to note that in the considered live streaming architecture, the latency between the broadcasters and the end-users is in the order of few seconds, e.g., 1-2 secs, which is considered to be affordable by the users [121]. Considering the above, the focus of this work is on the cache optimization of the mobile edge caches.

2) Transcoding of 360^o videos

Similarly to the previous two chapters, we assume that the video library consists of $V = |\mathcal{V}|$ that are captured by the broadcasters, and $\mathcal{V} = \{1, \dots v, \dots, V\}$ is the set of 360^o videos that comprise the content library. The video transcoding at the Live Stream servers is performed using H.265/HEVC [4], but our scheme is compliant with other video codecs. Each video stream is encoded into a number

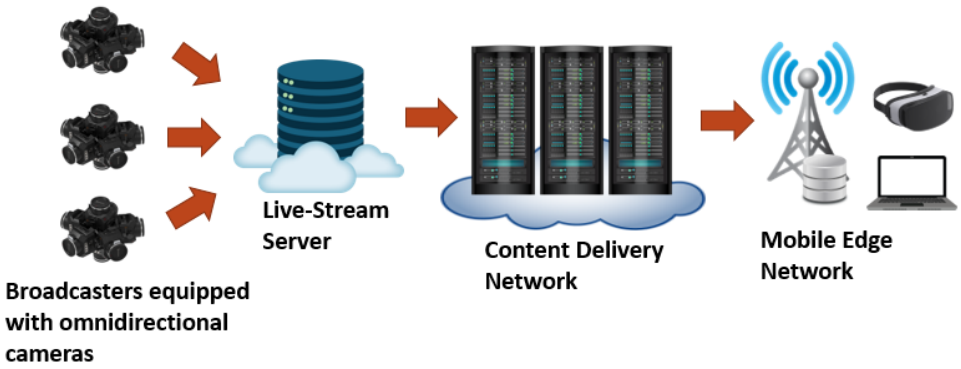

Figure 5.1: Considered live streaming architecture.

of L quality layers and M tiles, in the same way videos are encoded in Chapter 3. As the duration of each video may vary, each video consists of a variable number of GOPs, depending on the length of each video.

3) Content Delivery Network

CDN is comprised of geographically distributed servers that collaboratively cache and distribute popular content to a global reach, using high speed links. Each CDN server has the capacity to cache a number of video files. Using a CDN, the requested content in a geographic area may be served from the cache of the CDN server that is closer to the users. In this way, the demanded content is retrieved by the users with less latency, as their requests do not have to be routed to the Live Stream server. The use of CDNs reduces significantly the traffic reaching the Live Stream server.

4) Mobile Edge Network

We consider a mobile network architecture as the one depicted in Fig. 5.2. Similarly to the previous two chapters, this network consists of N SBSs that are

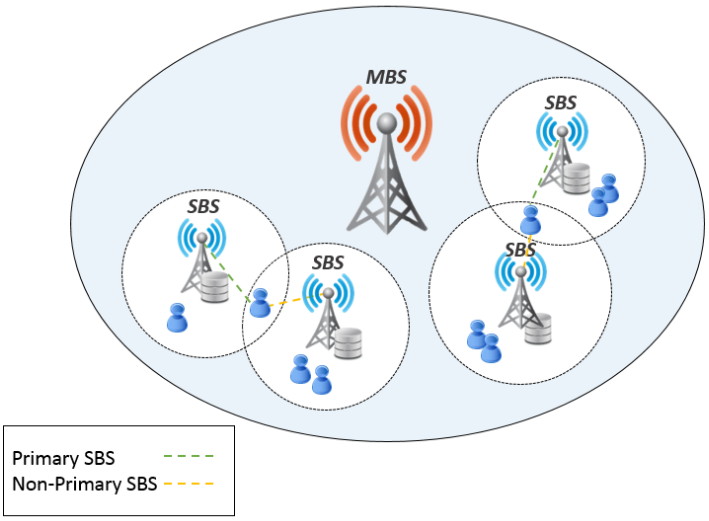

Figure 5.2: Considered mobile-edge architecture. The connection of users that reside in the coverage area of multiple SBSs is depicted with green dashed-lines for their primary SBS, and with yellow dashed-lines for their non-primary SBSs.

equipped with a cache capacity $C_n \geq 0,\ \forall n \in \mathcal{N}$, an MBS, and U users. The transmission ranges of the SBSs are given by the set $\mathcal{P} = \{p_1, \dots p_n, \dots, p_N\}$, and the communication range of the MBS is p_{N+1}. The MBS is connected with the CDN through a high capacity backhaul link, i.e., optical fiber, while the connection of the SBSs with the CDN is established through the MBS.

In contrast to the previous two chapters, users located in the overlap of the coverage areas of multiple SBSs can be associated with any of these SBSs as follows. The primary SBS for each user is the one that has the maximum SINR, while the rest of the SBSs are the non primary ones. When a user makes a request for a number of tiles, if the requested tiles are cached at the primary SBS, they will be served from that SBS to the user, However, when some of the requested tiles are not cached at the primary SBS, but are stored in the cache of one (or more) of the other SBSs the user resides, these tiles will be delivered to the user from these caches.

Similarly to Chapter 4, we assume that time is slotted in T time slots. In each time slot $t \in \mathcal{T}$, we denote the request of user $u \in \mathcal{U}$ for GOP $g \in \mathcal{G}$ of a 360^o video $v \in \mathcal{V}$ by w_u^t. Let $W_u = \{w_u^1, \ldots, w_u^t, \ldots, w_u^T\}$ be the set which has T consecutive requests from user $u \in \mathcal{U}$.

5) End-to-end delivery constraint

As we have already mentioned, the overall delay a user experiences from when they request the data until the data is delivered to them is captured by the end-to-end delay. This delay depends from where the data is retrieved. In order to guarantee the timely delivery of the tiles of each GOP to the users, the constraint given by the Eq. (4.1) should be met. Recall that the delay needed to transmit one Mbit from the cache of the nth SBS to the uth user is denoted by d_{nu}. In addition, the delay needed to transmit one Mbit that is fetched from the backhaul of the MBS to a user is $d_{(N+1)u}$.

5.3 System Model

1) Caching Entity (CE)

We consider that each SBS is equipped with a CE, as shown in Fig. 5.3. This entity is responsible for deciding which 360^o videos and tiles should be cached at each SBS. Each caching entity is composed of a number of modules, e.g., User Requests Processor, User Requests Forecasting, Feature Updater, etc., that their operation will be discussed later.

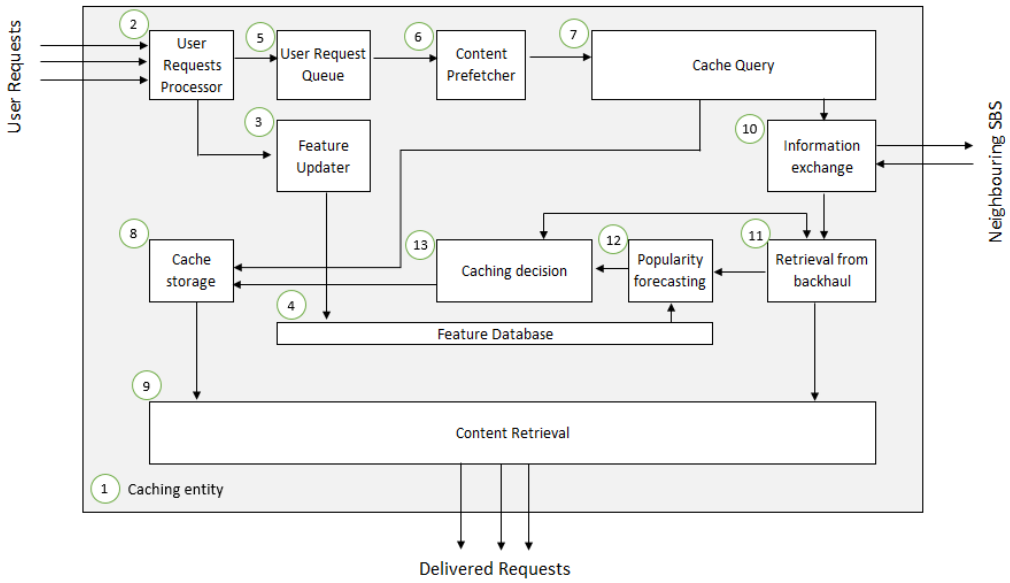

Figure 5.3: Flow of operations in a caching entity.

2) Users Request Processor (URP)

The URP module is responsible for decomposing the user requests $w_u^t, t \in \mathcal{T}, u \in \mathcal{U}$. Specifically, if a viewport consists of k tiles, each user request w_u^t is decomposed into $k+1$ requests $w_{u,i}^t$, as shown in Fig. 5.4. The first request $w_{u,0}^t$ is for receiving all the tiles of the requested 360^o video at the base quality. The rest k requests $\{w_{u,1}^t, \ldots, w_{u,i}^t, \ldots, w_{u,k}^t\}$ are for receiving each of the k tiles of the requested viewport in high quality. This decomposition gives our system the flexibility to decide which 360^o videos (all tiles at the base quality) should be cached to ensure interactivity, and which tiles of these videos should be cached in high quality. Let the set $\mathcal{W}_u^t = \{w_{u,0}^t, w_{u,1}^t, \ldots, w_{u,i}^t, \ldots, w_{u,k}^t\}$ describe these $k+1$ requests.

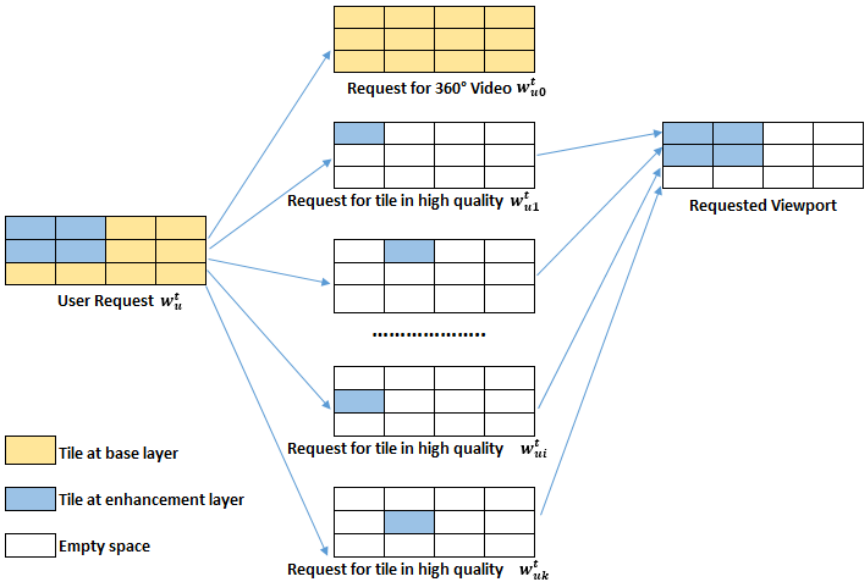

Figure 5.4: Decomposition of user request w_u^t into $k+1$ requests.

3) Feature Updater (FU)

The FU module is responsible for updating features regarding the requests for the various 360^o videos and tiles. Specifically, this module calculates in each time slot $t \in \mathcal{T}$, the number of times each request (for receiving either a 360^o video at the base quality, or a tile of a viewport in high quality) was encountered. The computed features are transferred to the Feature Database (FD) module, where this information is stored.

4) Feature Database (FD)

The FD module stores the features computed by the FU module regarding the number of times the various 360^o videos (all tiles at base quality) and tiles (in high quality) were requested at an SBS.

5) Users' Requests Queue (URQ)

After the decomposition of each user request into multiple requests by the URP module, the decomposed requests are directed to the URQ module. This module applies a technique called request coalescing [122]. According to this technique, when multiple requests for the same content arrive simultaneously at an SBS, the first request is prioritized for processing, while the rest of the user requests are held in a queue. This mechanism is needed because in live streaming scenarios, many people are watching the same content almost simultaneously. Without such mechanism, in case a requested content is not cached at the SBS, a cache miss will occur for all the users' requests for that content. This would cause all the traffic related to that content to be redirected to the origin CDN or even the Live Stream server, causing the crashing of these servers.

6) Content Prefetcher (CP)

Due to the end-to-end delay, SBSs are not able to respond instantly to the users' head movements and transmit the demanded tiles by the users. To overcome this problem, the CP module is used. Specifically, when the CP module receives requests $\mathcal{W}_u^t = \{w_{u,0}^t, w_{u,1}^t, \ldots, w_{u,i}^t, \ldots, w_{u,k}^t\}$ for the various 360^o videos (in base quality) and tiles (in high quality) from the URP at the time slot t, it decides what content should be prefetched for the various 360^o videos and tiles for the next time slot $t+1$. Let the set $\mathcal{Z}_u^{t+1} = \{z_{u,0}^{t+1}, z_{u,1}^{t+1}, \ldots, z_{u,i}^{t+1}, \ldots, z_{u,k}^{t+1}\}$ denote the content that will be prefetched to the users regarding the time slot $t+1$. Enabling prefetching allows the timely delivery of the content to the users. To decide which content should be prefetched, similarly to Chapter 4, we use the LSR [112] algorithm. According to this algorithm, when the CP module receives

a user request $w_{u,i}^t$ at time slot t, the content $z_{u,i}^{t+1}$ that is decided to be prefetched for the time slot $t+1$ is considered to be the same with the request $w_{u,i}^t$. For the sake of simplicity, we assume that for the first time slot, the content that will be prefetched to the users is the requested content.

7) Cache Query (CQ)

The CQ module examines whether the content indicated by the CP module is already cached at the SBS. When this content is cached at the SBS, it is served to the user locally from the Cache Storage (CS) of the SBS. When the content indicated by the CP module is not available at the SBS, the Information Exchange (IE) module is activated to check whether it can be served by a neighbouring SBS. This happens when the user is in the communication range of the neighbouring SBS. In case the content is cached at a neighbouring SBS and the user can be associated with that SBS, the content is delivered to the user from that SBS. Differently, when the content indicated by the CP module is not cached at any neighbouring SBS the user may be associated with, the content is fetched at the SBS through the backhaul of the MBS, and served from there to the user.

8) Cache Storage (CS)

The CS module is responsible for storing the 360^o videos at the base quality and the tiles of the cached videos in high quality. Each SBS has a separate cache storage module with capacity C_n, where n is the index of the SBS node.

9) Content Retrieval (CR)

The CR module is responsible for the delivery of the content to the users. This module retrieves the content either through: a) the CS module, when it is cached

at the SBS, or b) the Retrieval from Backhaul (RFB) module that retrieves the content from the backhaul, when it is not cached at the SBS.

10) Information Exchange (IE)

The IE module is responsible for the communication of an SBS with its neighbouring SBSs. Specifically, in case of a cache miss for a content at an SBS, the SBS checks whether that content may be served to the user from a neighbouring SBS. The communication between the SBSs is accomplished by millimeter wave links through the MBS. The delay needed for the above communication is captured by the delay parameter d_{nu}. Recall, that this parameter denotes the delay needed to deliver one Mbit from the nth SBS to a user, in the constraint (4.1).

11) Retrieval from Backhaul (RFB)

The RFB module is responsible for the retrieval of the content that will be prefetched to caches of the SBSs through the backhaul. After the retrieval of that content at the RFB module, its popularity is estimated by the Popularity Forecasting (PF) module, and a decision is made at the Caching Decision (CD) module about whether to cache that content.

12) Popularity Forecasting (PF)

The PF module forecasts at each time t, the popularity of the content that will be prefetched for the time slot $t+1$. Specifically, the PF module uses a window of h time slots, in order to capture the trends in the popularity of the content that will be prefetched to the users, and cache at the SBSs content that will be popular. To this aim, the features (number of requests for videos and tiles) stored at the FD module regarding the previous $h-1$ time slots along with the

current time slot $t \in \mathcal{T}$ are used. Let us denote by $\lambda^t_{n,v,0}$ the popularity of the 360^o video $v \in \mathcal{V}$ (base quality) at the SBS $n \in \mathcal{N}$ in the time slot $t \in \mathcal{T}$. Similarly, let $\lambda^t_{n,v,m}$ stand for the popularity of the tile $m \in \mathcal{M}$ of the video $v \in \mathcal{V}$ at the SBS $n \in \mathcal{N}$, where $\mathcal{M} = \{1, \ldots, m, \ldots M\}$.

One way to predict these popularities would be to use Recurrent Neural Networks (RNNs), as they have been shown to be effective for time series data forecasting [123]. However, simple RNNs cannot capture long-term dependencies, as they lack control structures, which causes the norm of gradients to decay or explode during training [124]. To overcome this problem, LSTM networks may be used [125]. LSTMs are a special type of RNNs able to learn long-term dependencies. Inspired by [126], we use an LSTM network for the forecasting of the popularity of the content retrieved by the RFB module. The LSTM network takes as input the features of content regarding the previous $h-1$ time slots along with the current time slot t, and outputs the estimated popularity for the content for the time slot $t+1$. The LSTM is initially pre-trained offline (warm-up phase) with historic data profiles using the backpropagation through time method, in order to find a good starting point for its weights. Then, these weights are used for the popularity prediction of the content retrieved by the RFB module. For more information regarding ANNs such as RNNs and LSTM networks, we refer the interested reader in *Appendix B*.

13) Caching Decision (CD)

The CD module makes decisions regarding whether to cache the retrieved content by the RFB module. To this aim, it uses the popularities predicted by the PF module.

Let us denote the total number of cached 360^o videos (at the SBS $n \in \mathcal{N}$)

at the base quality as b_n, and the total number of cached tiles in high quality as f_n. In addition, let the forecast popularities of the cached 360^o videos at the base quality at the time slot $t+1$ be described by the set $\mathcal{B}^{n,t+1} = \{B_1^{n,t+1}, \ldots, B_i^{n,t+1}, \ldots, B_{b_n}^{n,t+1}\}$, and the forecast popularities of the cached tiles in high quality by the set $\mathcal{F}^{n,t+1} = \{F_1^{n,t+1}, \ldots, F_j^{n,t+1}, \ldots, F_{f_n}^{n,t+1}\}$.

When the content that will be prefetched to a user is cached locally or at a neighbouring SBS the user resides, it is delivered to the user from the SBS it is cached. In such case, no decision is made at the CD module. In a different case, the content is retrieved at the RFB module, and a decision is made about whether to cache it. Specifically, a decision is first made about whether to cache at the SBS the request $z_{u,0}^{t+1}$ regarding the 360^o video indicated by the CP module (when it is not cached). If the predicted popularity by the PF module for the $z_{u,0}^{t+1}$ is $\lambda_{n,v,0}^{t+1}$, the $z_{u,0}^{t+1}$ will be cached at the SBS in the place of the ith cached 360^o video at the base quality if $\lambda_{n,v,0}^{t+1} > B_i^{n,t+1}$ and $\min(\mathcal{B}^{n,t+1}) = B_i^{n,t+1}$. Next, in case the 360^o video is cached at the base quality, a decision is made for each one of the tiles in high quality $\{z_{u,1}^{t+1}, \ldots, z_{u,i}^{t+1}, \ldots, z_{u,k}^{t+1}\}$ that are not cached about whether they should be stored at the CS module. Specifically, if the predicted popularity by the PF module for the tile $z_{u,i}^{t+1}$ is given by $\lambda_{n,v,m}^{t+1}$, the tile $z_{u,i}^{t+1}$ will be cached in the place of the jth cached tile in high quality if $\lambda_{n,v,m}^{t+1} > F_j^{n,t+1}$ and $\min(\mathcal{F}^{n,t+1}) = F_j^{n,t+1}$. However, if the initial decision regarding the request $z_{u,0}^{t+1}$ was not to cache it, no further decision is made, and none of the content requests $\{z_{u,1}^{t+1}, \ldots, z_{u,i}^{t+1}, \ldots, z_{u,k}^{t+1}\}$ regarding the tiles in high quality are cached.

Following the above workflow, the cache is populated with the most popular 360^o videos at the base quality. Tiles in high quality are cached only for the videos with cached base layer tiles. The number of cached tiles in high quality for each cached 360^o video depends on videos' popularity, i.e., the more popular

Algorithm 5.1 Caching decisions using forecast popularities

Offline Phase
Pre-train the LSTM network with historic transition profiles, using the back-propagation through time method
Online Phase
for each time slot t **do**
  for each user u **do**
    for each user request $w_{u,i}^{t}, i \in \{0, 1, \ldots, k\}$ **do**
      if $z_{u,0}^{t+1}$ (all tiles at base quality) are not cached **then**
        if $\lambda_{n,v,0}^{t+1} > B_i^{n,t+1}$ and $\min(\mathcal{B}^{n,t+1}) = B_i^{n,t+1}$ **then**
          Cache $z_{u,0}^{t+1}$ in place of the ith cached 360^o video at base quality
        end if
      end if
      if $z_{u,0}^{t+1}$ (all tiles at base quality) are cached **then**
        if $z_{u,i}^{t+1}$ (tile in high quality) is not cached **then**
          if $\lambda_{n,v,m}^{t+1} > F_j^{n,t+1}$ and $\min(\mathcal{F}^{n,t+1}) = F_j^{n,t+1}$ **then**
            Cache $z_{u,i}^{t+1}$ in place of the jth cached tile in high quality
          end if
        end if
      end if
    end for
  end for
end for

is a 360^o video, the more tiles are cached at the SBSs. Hence, for the least popular videos a small number of tiles or even no tiles may be cached at the SBSs. This provides our scheme greater flexibility in deciding how many tiles to cache per video, helps increase the cache hits, and enhance the quality of the displayed video, as we see in the next section. The aforementioned decision process (cache updates) made by the CD module using the forecast popularities by the PF module is summarized in Algorithm 5.1.

5.4 Performance Evaluation

In this section, we evaluate the performance of the proposed framework for enabling live 360^o video streaming. For all the schemes under comparison, users can be associated with multiple SBSs, when they reside in the transmission range of these SBSs. Hence, they can obtain their data from one of the neighboring SBSs when the data is not found in the primary SBS. We should note that when the users' requests arrive at an SBS, the cache update decisions are made at that SBS regardless if the users obtained their data from a neighboring SBS, or the backhaul.

5.4.1 Simulation Setup

Following a similar methodology with that in Chapter 4, we compare the performance of the proposed scheme with that of the LFU, LRU, and FIFO algorithms. However, differently from Chapter 4, in these chapter the comparison schemes are always applied on a per GOP basis. In addition, the cache capacity is split into two tiers. The first tier is used to store all tiles of each cached 360^o video at the base quality. The second tier is used only for 360^o videos that are already cached at the base quality, in order to store for these videos tiles in high quality according to the selected caching policy. Considering the above, the schemes under comparison, as well as the proposed scheme, are described below.

1. *Least Frequently Used (LFU):* In this scheme, the network operator keeps track of the number of requests that occurred for each cached 360^o video and tile in high quality of each GOP. When a request for the gth GOP of a 360^o video arrives at an SBS, the LSR algorithm is used to decide

which content should be prefetched as follows: a) if no tiles of the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the 360^o video that was requested the least frequently will be evicted from the SBS cache. Then, the tiles indicated by the LSR algorithm will be cached in the corresponding places of the evicted tiles; b) if the tiles of the GOP $g+1$ are cached at the first tier at the base quality, but some (or all) of the requested tiles are not cached in high quality, the cached tiles in high quality at the second tier that were requested the least will be evicted, and the requested tiles that were not cached will be stored in the place of the evicted tiles.

2. *Least Recently Used (LRU):* In this scheme, the network operator keeps track of how recent are the requests that occurred for each cached 360^o video and tile in high quality of each GOP. When an SBS receives a user request for the GOP g of a 360^o video, according to the LSR algorithm: a) if no tiles of the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the 360^o video that was requested the least recently will be evicted from the SBS cache. Then, the tiles indicated by the LSR algorithm will be cached in the corresponding places of the evicted tiles; b) if the tiles of the GOP $g+1$ are cached at the first tier at the base quality, but some (or all) of the requested tiles are not cached in high quality, the cached tiles in high quality at the second tier that were requested the least recently will be evicted, and the requested tiles that were not cached will be stored in these places.

3. *First In First Out (FIFO):* In this scheme, the network operator keeps track of when the requests for each cached 360^o video and tile in high

quality of each GOP happened. For a user request for the GOP g of a 360^o video, according to the LSR algorithm: a) if no tiles for the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the 360^o video that was requested the earliest will be evicted from the cache at the SBS. Next, all tiles of the viewport predicted by the LSR algorithm for the GOP $g+1$ will be cached at the SBS; b) if the tiles of the GOP $g+1$ are cached at the first tier at the base quality, but some (or all) of the requested tiles in high quality are different from the tiles of the viewport indicated by the LSR, the cached tiles in high quality at the second tier that were requested the earliest will be evicted, and the requested tiles that were not cached will replace these tiles.

4. *Proposed Scheme:* In the proposed scheme, the caching decisions are performed following the cache update framework described in Section 5.3. The proposed scheme uses popularity forecasting to decide with what content to populate the SBSs caches and how to update them. For each cached video, all the tiles in base quality are cached at the SBS. In addition, for each GOP, a number of tiles for the most popular videos are cached in high quality.

For all the conducted experiments, unless otherwise specified, we assume a cellular network that consists of $N = 3$ SBSs along with an MBS. The coverage range of each SBS is p_n = 200m, and the coverage range of the MBS is p_{N+1} = 2000m. The cache capacity of the SBSs is set to be enough to cache 10% of the 360^o videos content libary. This space is calculated assuming that for each GOP of each cached 360^o video, all the tiles at the base quality along with the tiles of one viewport in high quality are cached. However, as we have already

mentioned, for all schemes the number of tiles in high quality that will be cached for each 360^o video depends on the corresponding caching policy of each scheme. Furthermore, we consider that the total number of users is $U = 540$, who are randomly placed in the coverage area of the $N = 3$ SBSs. The delay at which data is delivered from the cache of the users' primary SBS is $d_{nu} = 1/14$ sec/Mbit. The delay at which data is delivered from the cache of a SBS that is not the primary one to a user equals to $d_{nu} = 1/13$ sec/Mbit. When a request for a 360^o video at base quality or tile in high quality is not cached at any of the SBSs the user resides, the delay needed to deliver that request from the backhaul equals to $d_{(N+1)u} = 1/2.9$ sec/Mbit.

The content library is comprised of $V = 100$ videos. Each 360^o video has a duration of 300 GOPs, with each GOP lasting $t_{disp} = 1$ sec. Each GOP of the 360^o videos is encoded in $M = 12$ tiles and $L = 2$ quality layers, while the size of each viewport is 4 tiles. The considered viewports are depicted in Fig. 4.4. The bitrate of the base layer is 2 Mbps, while the bitrate of the enhancement layer is 12 Mbps. The distortion reduction achieved by acquiring a tile at the base quality layer is $\delta_{vg1m} = 30$ dB, while the distortion reduction achieved obtaining a tile at the enhancement quality layer is $\delta_{vg2m} = 10$ dB.

We assume that the popularity of the 360^o videos through the GOPs is time-varying, and at any given moment, users may drop the 360^o video they are watching to view another one. To this aim, we assume that for the first GOP, the users' requests for the various 360^o videos follow the Zipfian distribution (see Eq. (3.31)), with shape parameter parameter $\eta_v = 1$.

To capture the evolution of the popularity of the 360^o videos with the time, inspired by [127], we assume that the probability of a user to stop watching a 360^o video at the GOP $g \in \{2, \ldots, G\}$ follows the Weibull distribution. Specifically,

we assume that each one of the 360^o videos that comprise our content library falls with equal probability to one of the 14 video categories, e.g., News, Sports, Education, etc., presented in [127]. Then, according to the video category each 360^o video falls into, we use the corresponding Weibull distribution parameters presented in [127], to estimate for each user, the probability of dropping a 360^o video at the GOP $g \in \{2, \ldots, G\}$. For each GOP, when users stop watching a 360^o video, the probability to select a different one to watch follows again the Zipfian distribution.

In terms of the viewports' requests, we simulated them with realistic navigation patterns obtained from the dataset described in [46], following a similar methodology with the one discussed in Chapter 4. To this aim, we initially sampled 10 different videos from this dataset, where for each sampled video, we obtained 30 trajectories. To assign these trajectories to the considered user requests, we mapped with equal probability, each of the $V = 100$ videos that comprise the content library to one of the 10 sampled videos. Then, for each user request for a specific 360^o video, according to its mapped index, we assigned with equal probability one of the 30 available trajectories for that video.

5.4.2 LSTM Neural Network training

We consider a Deep LSTM network comprised of four layers. The input layer gets as input a 3D tensor with shape (samples, time-steps, 1). Each of the two hidden layers is an LSTM layer with 100 LSTM cells. The output layer is a Dense layer comprised by 1 unit. The Deep LSTM network is implemented with the open source library Keras in Python. The hidden layers have as activation function the ReLu function. The optimizer used is Adam. The LSTM network is initially pre-trained (warm-up phase) with 2000 samples of historic data profiles, as explained

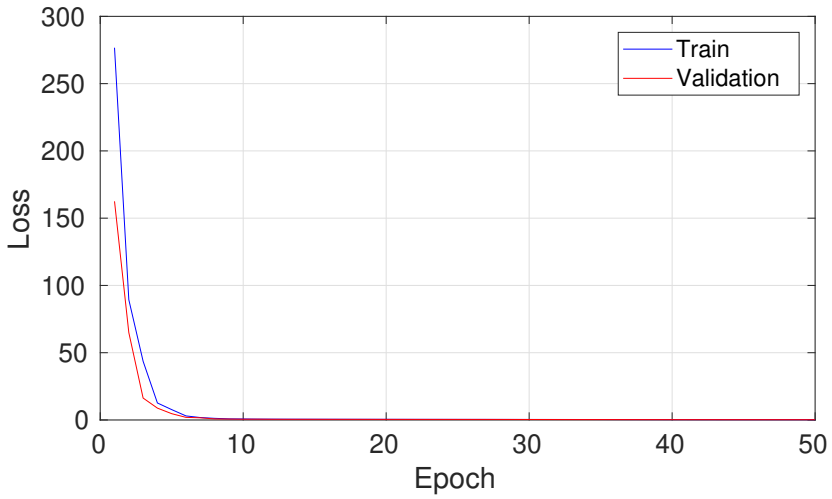

Figure 5.5: Popularity estimation of a 360^o video, using LSTM network.

in Section 5.3. Each sample represents the evolution in the popularity (of a 360^o video or tile in high quality) over $h = 10$ consecutive time slots (time-steps). The LSTM is pre-trained with a batch size of 300 for 50 epochs, using the MSE loss function between the outputs of the LSTM and the actual popularities. The historic data profiles are split into a training set and validation set, where the training set accounts for the 90% of the historic data profiles, and the validation set the rest 10%. After the pre-training of the LSTM, the trained weights are used by the PF module for the forecasting of the future content popularities (online phase). Specifically, during the time slot t, for each content retrieved by the RFB module, the features regarding the $h - 1 = 9$ previous time slots along with the current time slot t are provided as an input to the LSTM network, while its output is the predicted popularity at the time slot $t + 1$. During the online phase, the weights of the LSTM network are updated every 20 time slots by randomly sampling 100 samples from the FD module, and training the LSTM network with that samples for 20 epochs. As we can note from Fig. 5.5, the loss decreases with the number of epochs for both training and validation sets. We

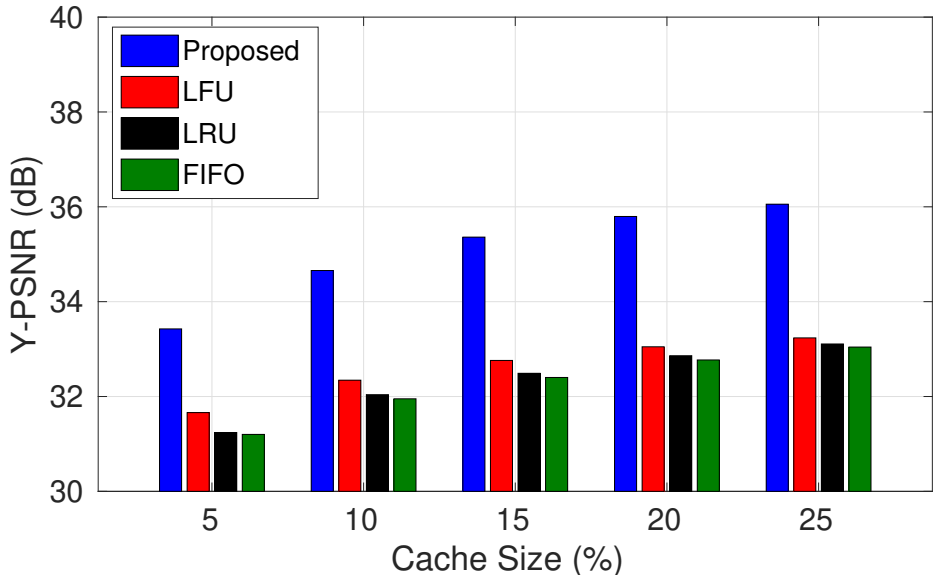

Figure 5.6: Y-PSNR of the rendered viewports with respect to the cache size for all the schemes under comparison.

can also see that the loss function converges to zero which means that the LSTM network is able to predict the popularity of the content that is prefetched to the caches of the SBSs with a small error.

5.4.3 Parameter Analysis

1) Cache Size

We first study the impact of the cache size on the overall quality of the rendered viewports. To this end, we vary the cache capacity C_n in the range [5, 25]% of the content's library size. We can note from Fig. 5.6 that the proposed scheme outperforms the schemes under comparison significantly in terms of the overall quality of the rendered viewports in all the range of cache sizes. Specifically, for small cache sizes, i.e., 5%, the performance gap between the proposed scheme and the LFU, LRU, and FIFO is approximately 1.7 dB, 2.1 dB, and 2.2 dB, respectively. For large cache size values, i.e., 25%, this gap grows to about 2.8

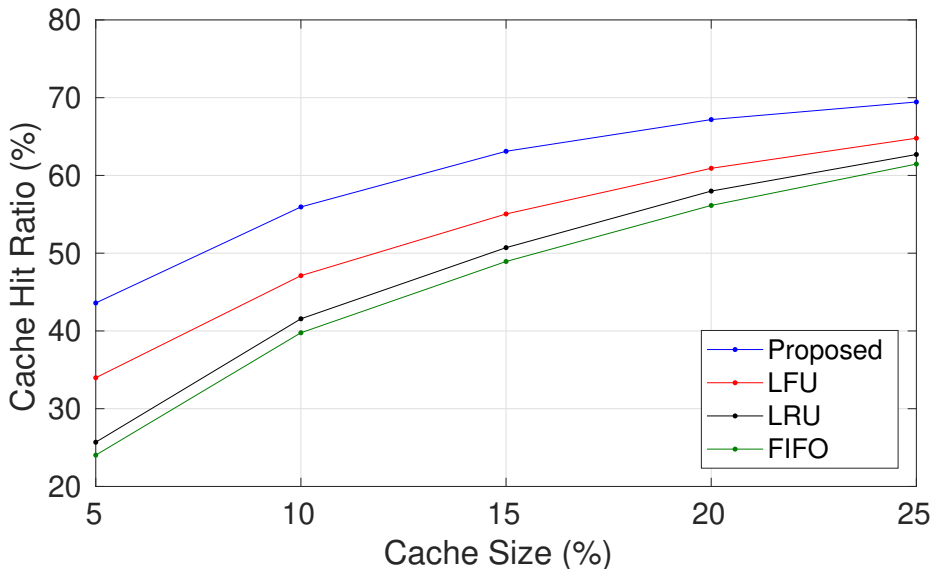

Figure 5.7: Cache Hit Ratio with respect to the cache size for all the schemes under comparison.

dB, 2.9 dB, and 3 dB, respectively. This performance gap is attributed to the fact that the proposed scheme achieves a better cache hit ratio compared to its counterparts, as is evident from Fig. 5.7. Specifically, the performance gap between the proposed scheme and the LFU, which is the second best performing scheme in terms of the cache hit ratio for small cache sizes (e.g., 5-10%) is about 9%. When the cache size takes large values, i.e., 25%, this gap closes to about 5%. This is because as the cache size increases, more content can be cached at the SBSs, and all schemes benefit from the additional cache space. The proposed scheme achieves the highest cache hit ratio due to the use of the LSTM network, which can help accurately predict the popularity evolution of the content that will be prefetched to the SBSs caches. Due to the increased cache hit ratio in the proposed scheme, more tiles are delivered in high quality to the users from the caches of the SBSs at a small delay. Thus, it is more likely for a greater number of tiles in high quality to be delivered in total from the caches of the SBSs along

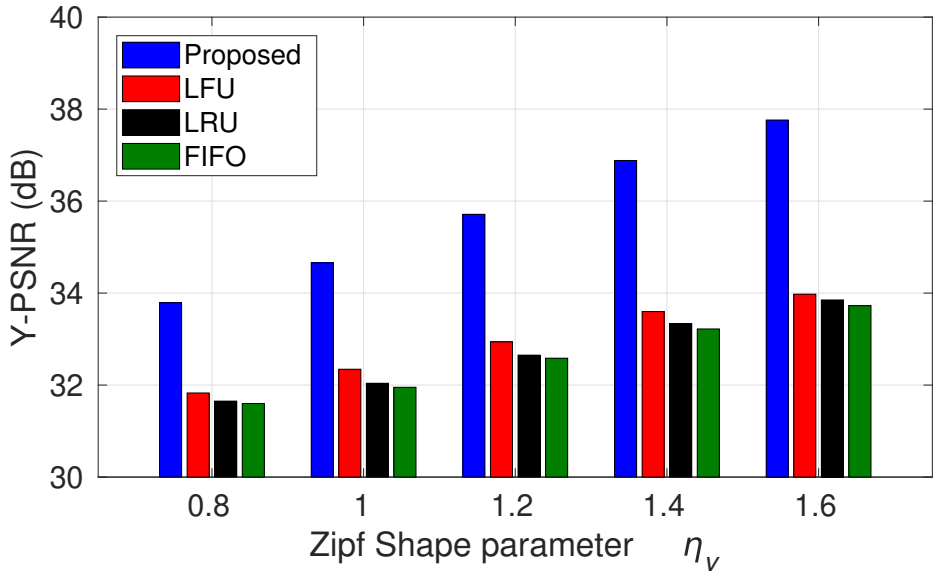

Figure 5.8: Y-PSNR of the rendered viewports with respect to the Zipf shape parameter of the 360^o videos for all schemes under comparison.

with the backhaul of the MBS to the users under the tight end-to-end delivery constraint.

2) Video popularity distribution

In Fig. 5.8, we investigate the impact of the users' requests for the various 360^o videos on the quality of the rendered viewports. To this aim, we vary the Zipf shape parameter η_v in the range $[0.8, 1.6]$. As we can see, an increase in the Zipf shape parameter η_v leads to an increase in the overall quality of the rendered viewports for all the schemes. This is because as the parameter η_v increases, the video popularity distribution gets steeper, and a smaller number of 360^o videos becomes more popular. As a result, the overall cache efficiency increases, as shown in Fig. 5.9. Thus, more tiles will be served directly from the caches of the SBSs at a small delay, allowing more tiles to be served in total to the users under the end-to-end time constraint. Similarly to the previous comparison,

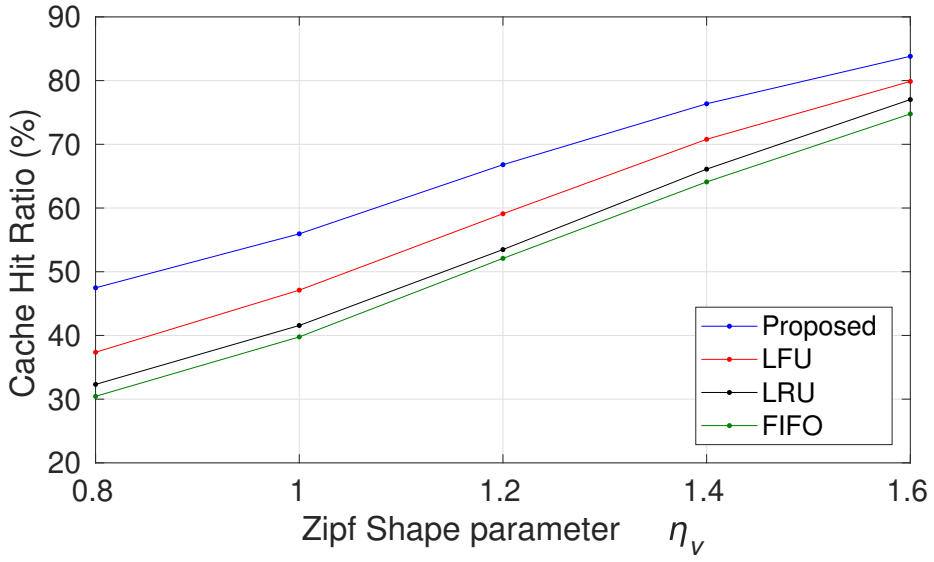

Figure 5.9: Cache Hit Ratio with respect to the Zipf shape parameter of the 360^o videos for all schemes under comparison.

the superiority of the proposed scheme regarding the overall cache hit ratio is attributed to the accurate prediction of the popularities of the content that is prefetched to the SBSs caches, using LSTM networks. As the value of the shape parameter increases from 0.8 to 1.6, the performance gap between the proposed scheme and the LFU widens from 1.9 dB to about 3.7 dB. Similar observations can be made from the comparison of the proposed scheme with the LRU and FIFO schemes.

3) Viewports popularity distribution

To examine the impact of the viewports' popularity on the overall quality of the rendered viewports, we assume that the viewports' popularity follows a Zipf distribution with shape parameter η_p. We vary the shape parameter η_p in the range $[0.5, 2.5]$, while we keep the cache capacity constant at $C_n = 10\%$. The performance of all the schemes under comparison is shown in Fig. 5.10. We can note that an increase in the value of the shape parameter η_p leads to an increase in

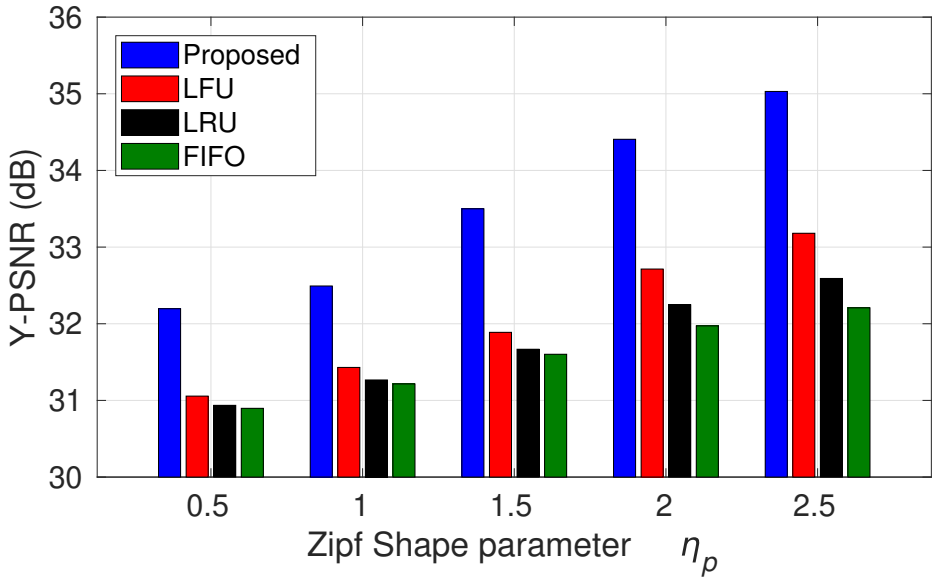

Figure 5.10: Y-PSNR of the rendered viewports with respect to the Zipf shape parameter of the viewports for all schemes under comparison.

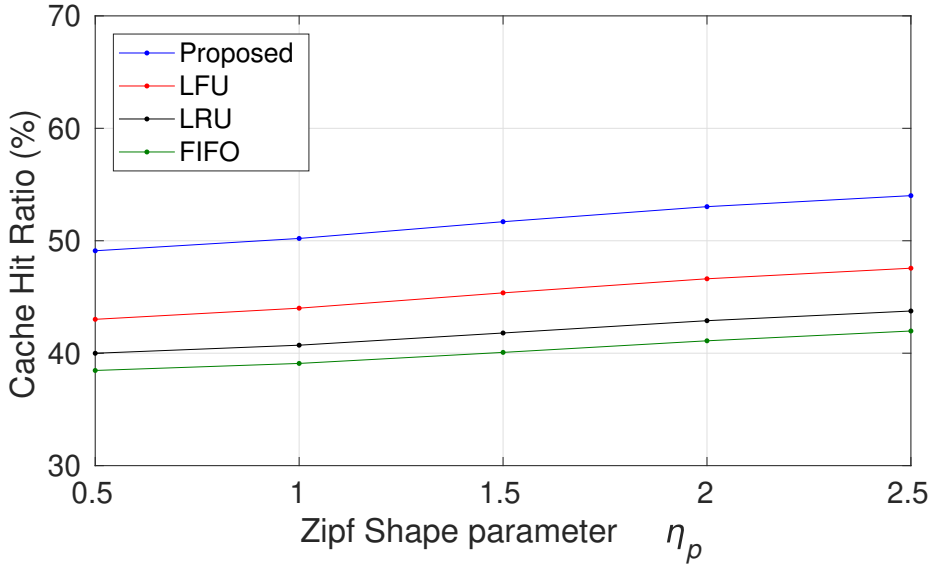

Figure 5.11: Cache Hit Ratio with respect to the Zipf shape parameter of the viewports for all schemes under comparison.

the overall quality of the rendered viewports for all the examined schemes. This is attributed to the fact that as the shape parameter η_p increases, the requests for the viewports become less diverse, and a smaller number of viewports is more

popular. Thus, most of the requests for tiles in high quality are served from the SBSs while respecting the end-to-end constraint, and overall the cache space is better used, as is evident from Fig. 5.11. For small values of η_p, i.e., $\eta_p = 0.5$, the performance gap between the proposed scheme and the LFU, LRU, and FIFO schemes is about 1.1 dB, 1.2 dB, and 1.3 dB, respectively. When the shape parameter η_p is large, i.e., $\eta_p = 2.5$, the performance gap between the proposed scheme and the LFU, LRU, and FIFO grows to about 1.8 dB, 2.4 dB, and 2.8 dB, respectively.

4) SBS Radius

To understand the impact of users' association with multiple SBSs on the overall quality of the rendered viewports, we vary the SBSs radius p_n in the range $[200, 300]$m. As the radius of the SBSs increases, the overlap between the coverage areas of the SBSs also increases. This results in more users being within the transmission range of multiple SBSs, and hence being able to be associated with multiple SBS. For the sake of completeness, we also examine the case where users are assigned only to the SBS with the maximum SINR. In such a case, the increase of the transmission range of the SBSs from 200m to 300m does not affect the overall rendered quality of the viewports because users are always assigned to the same SBSs regardless of the increase in the SBSs' transmission range. The simulation results are depicted in Fig. 5.12. As we can see, an increase in the radius of the SBSs leads to an increase in the overall quality of the rendered viewports due to the increase of the cache hit ratio achieved because users will be associated with multiple SBSs (see Fig. 5.13) for all cache sizes. As expected, when users are assigned to the SBS with the maximum SINR, the overall quality of the rendered viewports is lower compared to its counterparts. This is due

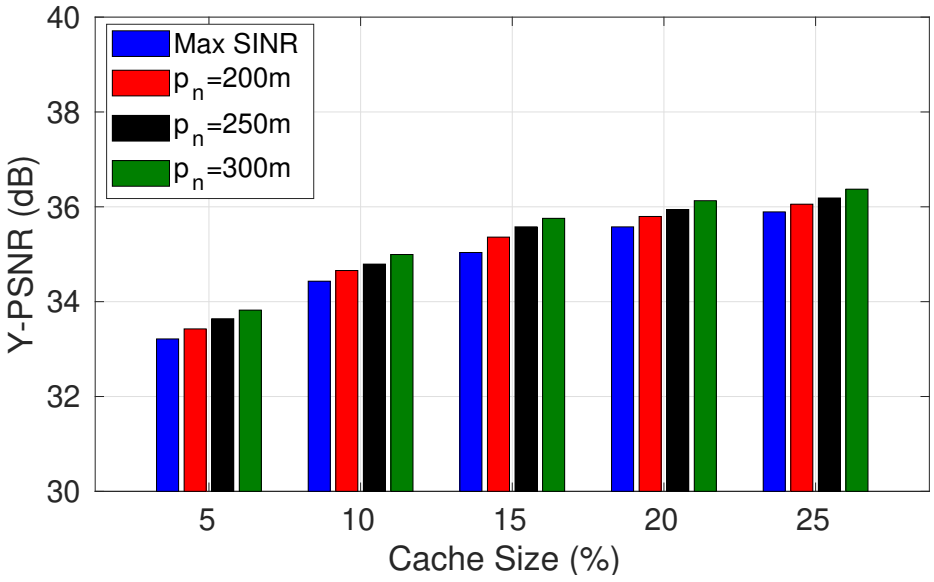

Figure 5.12: Y-PSNR of the rendered viewports with respect to the cache size for all schemes under comparison.

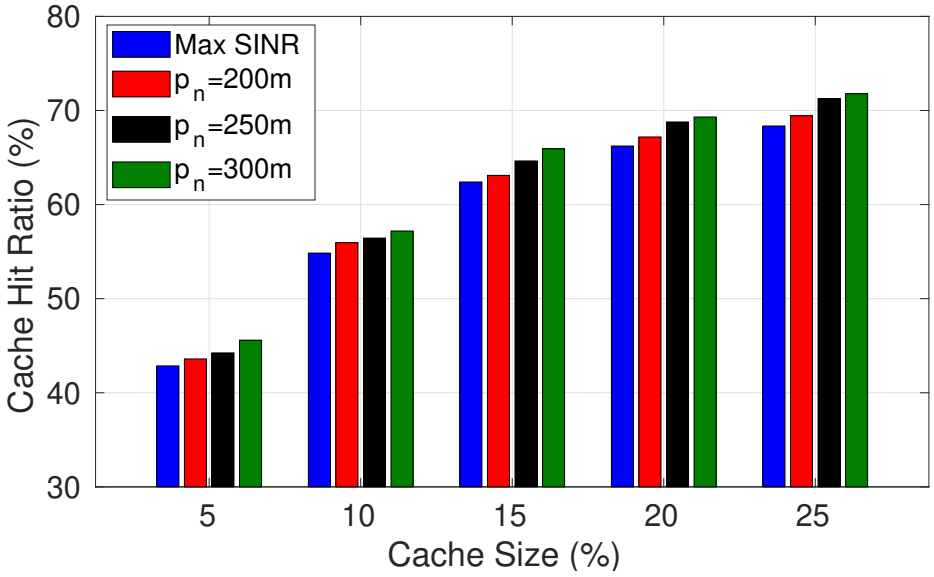

Figure 5.13: Cache Hit Ratio with respect to the cache size for all schemes under comparison.

to the fact that users are associated with only one SBS from which they can download their data.

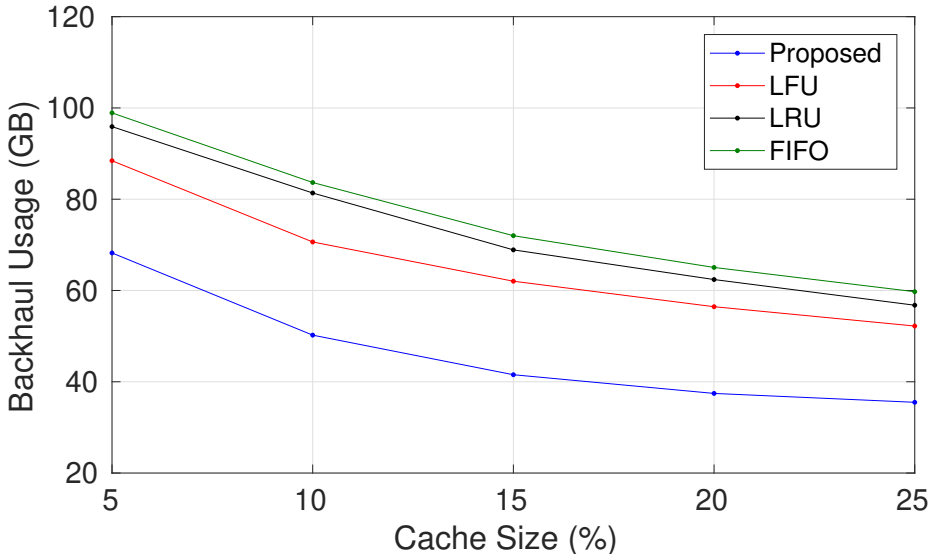

Figure 5.14: Backhaul usage with respect to the Cache Size for all schemes under comparison.

5) Backhaul Usage

In Fig. 5.14, we examine the backhaul usage of the MBS with respect to the cache size of the SBSs for all the schemes under comparison. To this aim, we vary the cache capacity of the SBSs in the range $[5, 25]\%$ of the content library. We can observe that an increase in the cache size of the SBSs leads to a decrease in the backhaul usage of the MBS for all schemes. This is because as the cache size increases, most of the demanded content is cached at the SBSs. As a result, more user requests will be served directly from the SBSs caches with no need to use the MSB's backhaul to fetch content from the core network. We can further note that as the cache size of the SBSs increases from 5% to 25%, the performance gap in terms of the backhaul usage between the proposed method and the LFU scheme closes from about 20.2 GB to about 16.7 GB. This is attributed to the fact that as the cache size increases, more content will be delivered to the users from the caches of the SBSs in both cases. Similar conclusions can be drawn

when comparing the proposed scheme with the LRU and FIFO schemes.

5.5 Conclusion

In this chapter, we studied the problem of online caching to support live streaming of 360^o videos in mobile networks. The proposed framework used LSTM networks to predict the evolution of the popularity of video tiles in future GOPs. These estimates were used to update the cached content at the SBSs, and allowed prefetching of it to the users on time. In addition, it permitted the backhaul usage to be minimized, and the overall quality of the rendered videos to be increased. To further enhance the performance of our proposed method, we exploited the potential association of users with multiple SBSs. Hence, users located in the overlapping coverage areas of the SBSs had access to all the caches of these SBSs. We tested our scheme for both real and synthetic navigation patterns and compared it with the LFU, LRU, and FIFO schemes. From the results, it was showed that our method outperformed its counterparts significantly, in terms of the overall quality of the rendered viewports, the cache hit ratio and the backhaul usage. In addition, it made apparent the benefits of using LSTM networks to predict the evolution of the content popularity, and make the most from content prefetching and caching at the SBSs.

6

Discussion and Future Work

6.1 Summary

360^o video is an essential component of VR/AR systems that provides users an immersive experience. However, 360^o video is associated with high bandwidth requirements, which makes streaming of such content on mobile networks challenging. Edge caching can be utilized as an enabling technology to facilitate the delivery of video content in wireless cellular networks. Through edge caching, users receive content with less latency, and the usage of the backhaul links is limited. However, existing edge caching schemes proposed for regular videos perform suboptimally for the delivery of 360^o videos. This is because 360^o videos' size is significantly larger than that of standard videos, which limits the number of

360^o video files that can be cached at the SBSs. In addition, caching entire 360^o videos is not an efficient strategy, as users are viewing only a portion of the 360^o video scenes, i.e., the viewport. Clearly, caching 360^o videos at the wireless edge without considering the aforementioned 360^o video special characteristics leads to significant bandwidth and storage resources waste, and may be unnecessary as parts of the 360^o videos may not be displayed.

6.2 Main Contributions

In this dissertation, we researched the optimization of edge caching for the delivery of 360^o videos. To this aim, we exploited advanced coding tools, i.e., encoding in multiple layers and tiles, to cache 360^o videos at the SBSs in a more fine-grained way. Tile encoding creates a flexible video stream structure that allows intelligent caching schemes to store only the parts of the 360^o video scenes that are viewed by the users. The main contributions of this thesis are summarized as follows:

- In Chapter 3, we examined the problem of joint caching and delivery of 360^o videos on a per-tile basis, given the content popularity distribution for the various 360^o videos and viewports (offline caching). To this end, we proposed a distortion-aware proactive caching scheme, which aimed to maximize the overall distortion reduction to the users' population. Our scheme exploited SBSs collaboration opportunities for users located in the coverage area of more than one SBSs. To determine the optimal caching placement, we took into account both the popularity of the 360^o videos and viewports, in order to cache only the parts of the scenes that are viewed by most of the users in high quality. Since the formulated problem is of high complexity, we decomposed it into a num-

ber of subproblems on per GOP basis, which were solved sequentially. Each of these subproblems was decoupled into its caching and routing components with the Lagrange relaxation method, while the solution of these subproblems was combined to determine the solution to the original problem. We compared our method with other baseline schemes that did not exploit SBSs collaboration and/or encoding into multiple quality layers and tiles. The results made clear that encoding 360^o videos into multiple quality layers and tiles allows their caching in a more fine grained way, and leads to an increase in the overall delivered quality compared to storing entire 360^o videos.

- In Chapter 4, we proposed a reactive caching scheme for caching 360^o videos at the SBSs, considering that the content popularity distribution is not known *a priori* (online caching). To determine which 360^o videos and tiles should be cached at the SBSs, we formulated the problem of 360^o videos caching as a Markov Decision Process. To reduce the dimensionality of the cache optimization problem, we introduced the concept of virtual viewports, which have the same number of tiles with standard viewports, but consist of the most popular tiles. To solve the formulated MDP, we used the DQN algorithm. We evaluated our method using both real and synthetic navigation patterns, and compared our solution with that of the LFU, LRU, and FIFO schemes. The results revealed that our method achieved a better performance compared to its counterparts in terms of the delivered quality of the rendered viewports, the overall cache hit ratio, and the backhaul usage.

- In Chapter 5, we proposed a novel caching scheme for live streaming of 360^o videos, considering unknown video and viewport popularities across the GOPs of the 360^o videos. Through the optimization of caching, we were able to

maximize the overall quality of the rendered viewports and reduce the usage of the backhual links. To determine which 360^o videos (base quality) and tiles (high quality) would be cached at the SBSs, we deployed an LSTM network. Using LSTM, we were able to predict the popularity of each 360^o video and tile for the next GOP, cache them at the SBSs, and facilitate the delivery of them to the users. To enhance the delivered quality of the 360^o videos to the users, we allowed the association of users with multiple SBSs (provided they were located in the coverage area of these SBSs). We compared our method with the LFU, LRU, and FIFO schemes, using both real and synthetic navigation patterns. From the results, it was shown that our method achieved a better performance compared to its counterparts in terms of the delivered quality of the rendered viewports, the overall cache hit ratio, and the backhaul usage. In addition, it was shown how the association of users with more than one SBSs leads to better caching and delivery strategies.

6.3 Future Work

We conclude this dissertation by presenting some research directions for future investigation. Some possible extensions of our work are:

- In the proposed caching scheme in Chapter 3, we considered that users' location are fixed. However, an interesting extension of that work would be to take into account users' mobility. This would make our formulated problem in Chapter 3 even more challenging, as the users association with SBSs would be more complicated, due to the uncertainty in the users' loacation. One way to model users' mobility would be with the use of Markov chains [128], where the uncertainty in the users' locations would be modelled with the transition

probabilities of the Markov chains. Another interesting extension of our work in Chapter 3, could be to consider that apart from SBSs caches, users may exploit local cache space in their devices, as in device-to-device (D2D) [129] communication systems. This approach would make the formulated problem in Chapter 3 even more challenging, as it would add one more layer of complexity due to the D2D communications. However, the use of D2D communications could allow users in proximity to share their cached contents, increase their experienced QoE, and further decrease the backhaul usage.

- In Chapter 4, the caching decisions of the proposed scheme were designed independently at each SBS. However, an interesting way to extend that work could be to design the caching decisions among the SBSs in a collaborative way. This may be accomplished by describing the problem of caching 360^o videos using multiagent MDPs [130], where each agent would be an SBS. However, compared to our approach in Chapter 4, using multiagent MDPs would further increase the overall complexity for the problem of caching 360^o videos. This is because the actions of each SBS would affect the state of the other SBSs. Another possible extension of our work in Chapter 4, would be to examine the effect of various encoding settings, e.g., number of quality layers, number of tiles, etc., on the overall quality of the rendered viewports. This is because different encoding settings, e.g., increased number of tiles, would increase the flexibility in the caching decisions in terms of which parts of the 360^o scenes to cache, but at the expense of the increased dimensionality of the state space and action space.

- A topic of significant interest for extending our work in Chapter 5, would be to organize a number of subjective tests, where users will verify our findings

regarding 360^o live video streaming and caching. Such an approach has also the potential for creating a dataset regarding the users' navigation patterns, for live streaming 360^o videos. The main challenge of this extension is to schedule a large number of users to watch a number of 360^o videos at the same time. This is due to the fact that in live streaming scenarios, users watch the broadcasted videos "live", at the time these videos are created. The examination of live streaming 360^o videos and caching from the perspective of reducing the total energy consumption would be another interesting extension of the work presented in Chapter 5. This is because as the distances between the users and the SBSs they may be associated with are different, this leads to different transmission energies required to transmit data to the users.

Overall, 360^o video is an emerging technology, which will have a tremendous impact in fields such as medical training, education, and entertainment [131]. 360^o videos offer an immersive way to share our stories, by allowing users to interact in a virtual environment. Major content providers, e.g., YouTube, Facebook, etc, have already available a great number of 360^o videos for streaming. Besides recent advances of communications systems and the emergence of 5G networks, due to the high bandwidth requirements of 360^o videos, the delivery of them in mobile networks is still challenging. Since edge caching will be an instrumental component of 5G and beyond networks, it can be utilized with advanced coding tools, i.e., encoding into multiple layers and tiles, to satisfy the forthcoming high expectations of the users' demands. In addition, the use of machine learning techniques will be instrumental for facilitating the caching of 360^o videos when the content popularity distribution is unknown, and will enable the caching of the 360^o videos at the SBSs in a large scale.

Appendix A

Video Coding

A.1) Overview

Video compression is a key technology that allows videos to be communicated through the Internet, stored, and consumed by the users [3]. Video compression is the process of encoding the visual information of video files, which in turns helps to reduce the enormous size of the raw video streams. Video files are compressed by minimizing the redundancy in the video frames. There are two main types of redundancy, namely *spatial* and *temporal*.

To remove spatial redundancy, intra-frame compression is used. Each frame is compressed individually [132], and is treated as a still image. The techniques used for spatial compression of videos are similar to the ones used for image files, e.g., JPEG, PNG, etc. In spatial compression, various algorithms are employed to look for similarities though a frame, e.g., pixels in proximity of a blue sky, in order to exploit these similarities and describe large areas of an image in a more compact way.

To remove temporal redundancy, inter-frame compression is used. Specifically, sets of neighboring frames, also known as Group of Pictures (GOPs) are

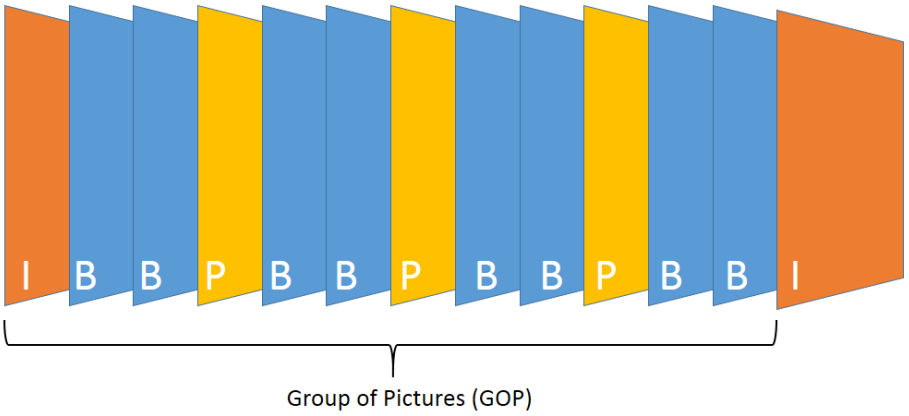

Figure A.1: Group of Pictures (GOP), comprised from I, B, and P frames

compressed together to exploit the similarities in a scene through the time. This is because consecutive frames may be correlated to a great extend, e.g., static background in the news. Each GOP my consist of a number of I, P, and B frames, as shown in Fig. A.1. Each type of frame is described in detail as follows:

- I-frame (intra-coded). This type of frame is encoded independently of the others, using only intra-frame compression. Because of that, it requires the most space compared to the P, and B frames. The advantage of I-frames is that they can be decoded independently. This offers flexibility to seek through a video and start its playback at any point an I-frame is present.

- P-frame (predictive-coded). This type of frame is encoded by means of both intra-frame compression, and inter-frame compression that encodes the differences that took place compared to the previous decoded frame, which may be an I-frame, or a P-frame. For this reason, this type of rame is smaller in size compared to an I-frame, however, it cannot be decoded independently.

- B-frame (bipredictive-coded). This type of frame is encoded by using both intra-frame compression, and inter-frame compression that encodes the differ-

ences that took place between both the preceding and the following decoded frames, which may be either an I-frame, or a P-frame. It requires even less space than an I-frame or a P-frame, however, similarly to the P-frame, it cannot be decoded independently.

A.2) Video Quality Evaluation

Although lossy video compression reduces significantly the size of the video files [133], it induces distortion to the original video files that may affect the QoE users enjoy. The perception of the experienced QoE by the users is subjective [134], and can be influenced by many factors such as the viewing environment. To avoid the induction of biases when evaluating video quality, subjective quality tests may be performed in controlled environments, where users are watching a number of videos and rate their perceived QoE. However, this process is not always practical and requires resources that are not always available, e.g., equipment for the playback of multiple videos, scheduling of users to watch them, etc. To address these limitations, objective quality quality metrics [135] are widely used to evaluate video quality in an algorithmic way. Some popular quality metrics are the Mean Squared Error (MSE), and the Peak Signal to Noise Ratio (PSNR).

MSE measures the average squared error between the pixels of the luminance component between the decoded frame and the reference frame. The MSE between two frames with resolution $M \times N$ is calculated as follows:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (\hat{Y}_{ij} - Y_{ij}) \tag{A.1}$$

where $\hat{Y}_{ij}$, (Y_{ij}) represents the values of the pixels $i \in \{1, \ldots, M\}, j \in \{1, \ldots, N\}$ of the luminance component of the decoded (reference) frame.

PSNR measures on a logarithmic scale the ratio between the maximum possible signal, relative to the MSE. Specifically, PSNR is calculated as follows:

$$PSNR = 10 \log_{10}(\frac{MAX_I^2}{MSE}) \tag{A.2}$$

where MAX_I is the maximum value a pixel can take in an image, e.g., for an 8-bit image $MAX_I = 255$.

Appendix B

Artificial Neural Networks

B.1) Introduction

An Artificial Neural Network (ANN) is a computational model consisting of a large number of interconnected units, inspired by biological neural networks such as the human brain [136]. ANNs are an essential part of multiple applications such as computer vision [137], speech recognition [138], and text interpretation [139].

The basic unit of a neural network is the neuron (see Fig. B.1), also known as node. A node receives several inputs $\{X_1, \ldots, X_N\}$, where each input is associated with a weight $\{W_1, \ldots, W_N\}$. In addition, a special input is provided to each node, which is called bias. This input has the value 1, and is associated with a weight b. The role of bias is to provide a node with more flexibility in terms of its computed output, given its inputs. The output of a node is defined from its activation function f, e.g., sigmoid, linear, etc. Specifically, the activation function takes as input the weighted sum $\sum_{i=1}^{N} X_i \cdot W_i + b$ of all the inputs of the node, and calculates the $f(\sum_{i=1}^{N} X_i \cdot W_i + b)$, which is the output Y of the node.

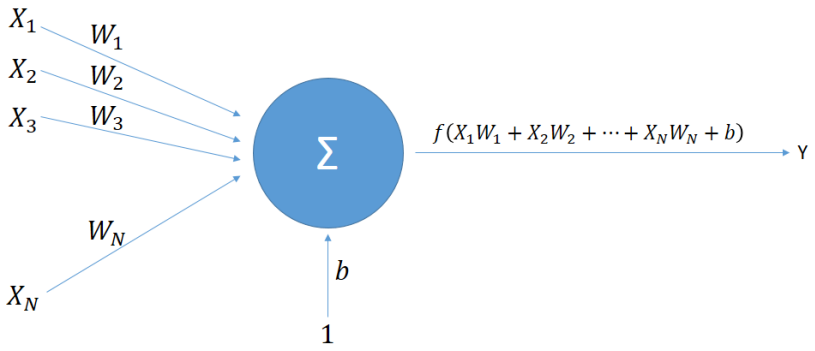

Figure B.1: Basic computation unit of neural networks

B.2) Feedforward Neural Network

A Feedforward Neural Network (FNN) consists of multiple layers, i.e., input layer, hidden layers, output layer, where each layer is comprised by a number of nodes, as shown in Fig. B.2. The nodes of adjacent layers are interconnected [140], with each connection being associated with a weight. The computations of a neural network are determined by the set of the weights associated with the interconnections of the various nodes of different layers, as well as the activation function of each node.

FNNs may be used for regression or classification. Both problems belong to the family of supervised machine learning. In both cases, a training dataset that contains historic data with known inputs along with their corresponding outputs is initially used for the training of an FNN, in order to initially estimate its weights. Afterwards, the performance, e.g., MSE, accuracy, of the FNN is tested on the test dataset.

Two problems that may occur during the training of ANNs such as FNNs are the underfitting, and the overfitting. In the former case, the trained model performs poorly both on the training and the test set. In the latter case, the

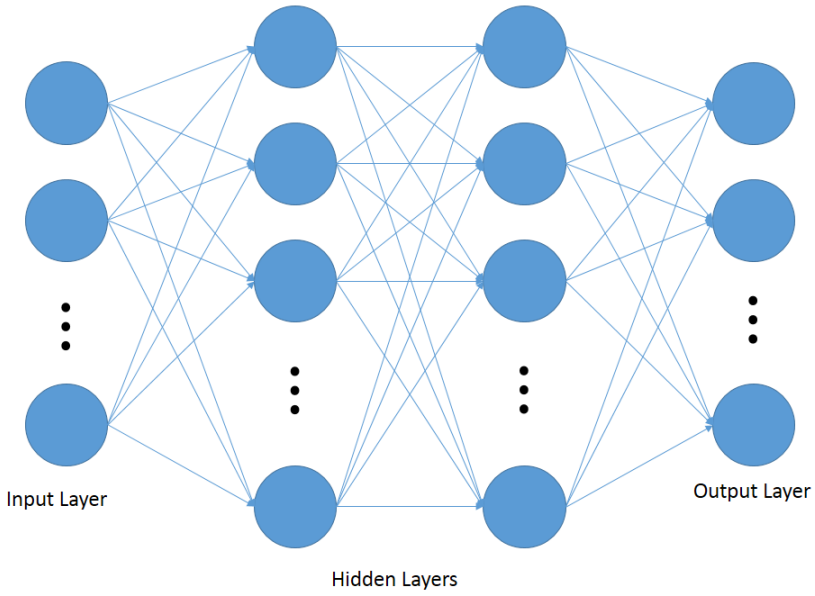

Figure B.2: Feedforward Neural Network

trained model has a very good performance on the training set, but performs poorly on the test set. Thus, the model fails to generalize on unseen data. Therefore, in order to have confidence in the ability of an FNN to make predictions, its training and testing must be performed on different datasets. In addition, a high performance should be observed both on the training, and the test dataset.

B.3) Recurrent Neural Network

Unlike FNNs where the information can move only in one direction, in Recurrent Neural Networks (RNNs) there are loops in the nodes of the hidden layers that do not restrict the flow of information in one way, as shown in Fig. B.3. Due to these loops, the output of the neurons in the hidden layers depends not only on their immediate inputs, but also on their previous state. This helps RNNs to maintain information over time, and makes them suitable for modeling tasks where the sequence of data is important, e.g., time-series, natural language processing, etc.

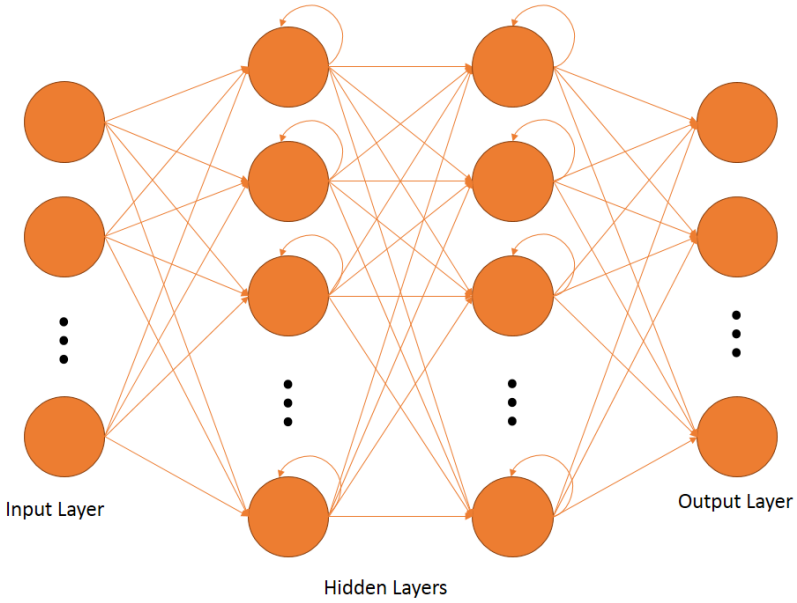

Figure B.3: Recurrent Neural Network

However, training RNNs to capture long-term dependencies can be difficult, as the norm of the gradients may vanish or explode.

B.4) Long Short-Term Memory Cell

To overcome the challenges in capturing long-term dependencies in RNNs, Long Short-Term Memory (LSTM) networks were proposed. LSTMs are a special type of RNNs, which have control structures that regulate the flow of information. Specifically, these control structures are called gates, and control the update and removal of information to the cell (memory) of the LSTM, as well as the output (see Fig. B.4). These gates are the forget gate, input gate, and output gate. The operation of these is described as follows:

- Forget gate: This gate decides what information should be kept and what information should be thrown away from the cell state. Specifically, it takes as input information from the previous hidden state h_{t-1} and the current

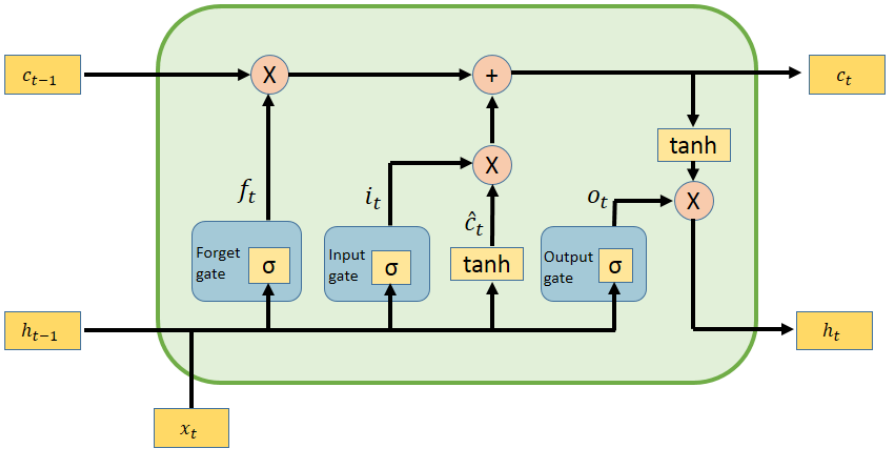

Figure B.4: Long Short-Term Memory (LSTM) cell

input x_t, and passes that information through a *sigmoid* function. The output of the *sigmoid* function is the output of the forget gate f_t, with values between 0 and 1. Values close to 0 are forgotten from the cell state c_{t-1}, while values close to 1 are kept.

- Input gate: This gate is responsible for the update of the cell state. Specifically, the previous hidden state h_{t-1} and the current input x_t are initially passed through a *sigmoid* function that outputs values between 0 and 1, which are the output i_t of the input gate. Values close to 0 are considered as not significant, while values close to 1 are important. The hidden state h_{t-1} and the current input x_t are also passed through the *tanh* function, which provides the output $\hat{c}_t$ with values between -1 and 1. Then, the output of the *sigmoid* i_t and the output of the *tanh* function $\hat{c}_t$ are multiplied, and the outcome of that multiplication, is added to the cell state. After that addition, the cell state is updated to c_t.

- Output gate: The output gate along with the updated cell state c_t deter-

mine what would be the next hidden state h_t. Specifically, the previous hidden state h_{t-1} and the current input x_t are initially passed through the *sigmoid* function. The output of the *sigmoid* function is denoted by o_t. Simultaneously, the updated cell state c_t is passed through the *tanh* function. The outcome of the *tanh* function is multiplied with the output of the *sigmoid* function, and their multiplication provides the next state h_t.

The above computations are summarized in the next equations.

$$\begin{aligned} f(t) &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i(t) &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o(t) &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ \hat{c}(t) &= tanh(W_c x_t + U_c h_{t-1} + b_c) \\ c(t) &= f_t \odot c_{t-1} + i_t \odot c_t \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{B.1}$$

where $W_i, U_i, b_i, i \in \{f, i, o, c\}$ are the weights and bias associated with the current input x_t and the previous hidden state h_{t-1} of the forget, input, output gate, and *tanh* function, respectively. In addition, $\odot$ stands for element-wise multiplication.

List of Publications

Journal Publications

J1. P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360^o videos in heterogeneous networks," *IEEE Trans.on Multimedia*, in press

Conference Publications

C1. P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360^o videos in heterogeneous networks," in *Proc. of IEEE 21st Int. Workshop on Multimedia Signal Processing (MMSP'19)*, Kuala Lumpur, Malaysia, Sep. 2019

C2. P. Maniotis and N. Thomos, "Smart caching for live 360^o video streaming in mobile networks," in *Proc. of IEEE 22st Int. Workshop on Multimedia Signal Processing (MMSP'20)*, Tampere, Finland, Sep. 2020

Preprints

P1. P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360^o videos in heterogeneous networks," *CoRR*, vol. abs/1902.09581, 2019.

P2. P. Maniotis and N. Thomos, "Viewport-aware deep reinforcement learning approach for 360^o video caching," *CoRR*, vol. abs/2003.08473, 2020.

Bibliography

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016—2021," White paper, Mar. 2017.

[2] K. Huang, P. Chien, C. Chien, H. Chang, and J. Guo, "A 360-degree panoramic video system design," in *Proc. of Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*, Hsinchu, Taiwan, Apr. 2014, pp. 1–4.

[3] G. J. Sullivan and T. Wiegand, "Video compression - from concepts to the h.264/avc standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, Jan. 2005.

[4] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[5] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, C. Chiang, Y. Wang, P. Wilkins, J. Bankoski, L. Trudeau, N. Egge, J. Valin, T. Davies, S. Midtskogen, A. Norkin, and P. de Rivaz, "An overview of core coding tools in the av1 video codec," in *Proc. of Picture Coding Symposium (PCS'18)*, San Francisco, CA, USA, Jun. 2018, pp. 41–45.

[6] D. Mukherjee, J. Han, J. Bankoski, R. Bultje, A. Grange, J. Koleszar, P. Wilkins, and Y. Xu, "A technical overview of vp9 – the latest open-source video codec," in *Proc. of Annual Technical Conference Exhibition*, Hollywood, CA, USA, Oct. 2013, pp. 1–17.

[7] Y. Zhang and R. Wang, "A study on the effects of head mounted displays movement and image movement on virtual reality sickness," in *Proc. of IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW'20)*, Atlanta, GA, USA, USA, Mar. 2020, pp. 631–632.

[8] M. Qadri, M. S. Hussain, S. Jawed, and S. A. Iftikhar, "Virtual tourism using samsung gear vr headset," in *Proc. of International Conference on Information Science and Communication Technology (ICISCT'19)*, Karachi, Pakistan, Pakistan, Mar. 2019, pp. 1–10.

[9] X. Xie and X. Zhang, "Poi360: Panoramic mobile video telephony over lte cellular networks," in *Proc. of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: ACM, Nov. 2017, pp. 336–349.

[10] A. TaghaviNasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, "Adaptive 360-degree video streaming using layered video coding," in *Proc. of IEEE Virtual Reality (VR'17)*, Los Angeles, CA, USA, Mar. 2017, pp. 347–348.

[11] A. Ghosh, V. Aggarwal, and F. Qian, "A rate adaptation algorithm for tile-based 360-degree video streaming," *CoRR*, vol. abs/1704.08215, 2017. [Online]. Available: http://arxiv.org/abs/1704.08215

[12] J. Tan, G. Cheung, and R. Ma, "360-degree virtual-reality cameras for the masses," *IEEE MultiMedia*, vol. 25, no. 1, pp. 87–94, Jan. 2018.

[13] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. of IEEE International Conference on Communications (ICC'17)*, Paris, France, May 2017, pp. 1–7.

[14] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, Sept. 2016.

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

[16] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Edge-caching wireless networks: Performance analysis and optimization," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2827–2839, Apr. 2018.

[17] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3665–3677, Oct. 2014.

[18] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, Apr. 2013.

[19] G. S. Paschos, A. Destounis, and G. Iosifidis, "Online convex optimization for caching networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 625–638, Apr. 2020.

[20] F. Gabry, V. Bioglio, and I. Land, "Online caching in heterogeneous networks," in *Proc. of IEEE International Conference on Communications (ICC'17)*, Paris, France, May 2017, pp. 1–5.

[21] N. Garg, M. Sellathurai, V. Bhatia, B. N. Bharath, and T. Ratnarajah, "Online content popularity prediction and learning in wireless edge caching," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1087–1100, Feb. 2020.

[22] S. Rahman, M. G. R. Alam, and M. M. Rahman, "Deep learning-based predictive caching in the edge of a network," in *Proc. of International Conference on Information Networking (ICOIN'20)*, Barcelona, Spain, Spain, Jan. 2020, pp. 797–801.

[23] M. Z. Shafiq, A. X. Liu, and A. R. Khakpour, "Revisiting caching in content delivery networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 1, p. 567–568, Jun. 2014.

[24] Y. Wang, X. Tao, X. Zhang, and G. Mao, "Joint caching placement and user association for minimizing user download delay," *IEEE Access*, vol. 4, pp. 8625–8633, Dec. 2016.

[25] P. Ostovari, J. Wu, and A. Khreishah, "Efficient online collaborative caching in cellular networks with multiple base stations," in *Proc. of IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Brasilia, Brazil, Oct. 2016, pp. 136–144.

[26] Q. Gong, J. W. Woods, K. Kar, and J. Chakareski, "Fine-grained scalable video caching," in *Proc. of IEEE International Symposium on Multimedia (ISM'15)*, Miami, FL, USA, Dec. 2015, pp. 101–106.

[27] B. Zhang, Z. Liu, S. H. G. Chan, and G. Cheung, "Collaborative wireless freeview video streaming with network coding," *IEEE Transactions on Multimedia*, vol. 18, no. 3, pp. 521–536, Mar. 2016.

[28] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Caching and operator cooperation policies for layered video content delivery," in *Proc. of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM'16)*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[29] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Distributed caching algorithms in the realm of layered video streaming," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 757–770, Apr. 2019.

[30] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile based hevc video for head mounted displays," in *Proc. of IEEE International Symposium on Multimedia (ISM'16)*, San Jose, CA, USA, Dec. 2016, pp. 399–400.

[31] M. Hosseini, "View-aware tile-based adaptations in 360 virtual reality video streaming," in *Proc. of IEEE Virtual Reality (VR'17)*, Los Angeles, CA, USA, Mar. 2017, pp. 423–424.

[32] J. Le Feuvre and C. Concolato, "Tiled-based adaptive streaming using mpeg-dash," in *Proc. of the 7th International Conference on Multimedia Systems*, Klagenfurt, Austria, May 2016, pp. 41–43.

[33] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Hevc-compliant tile-based streaming of panoramic video for virtual reality applications," in *Proc. of the 2016 ACM on Multimedia Conference*, Amsterdam, Netherlands, Oct. 2016, pp. 601–605.

[34] W. C. Lo, C. L. Fan, S. C. Yen, and C. H. Hsu, "Performance measurements of 360^o video streaming to head-mounted displays over live 4g cellular networks," in *Proc. of 19th Asia-Pacific Network Operations and Management Symposium (APNOMS'17)*, Seoul, South Korea, Sep. 2017, pp. 205–210.

[35] S. Rossi and L. Toni, "Navigation-aware adaptive streaming strategies for omnidirectional video," in *Proc of. IEEE 19th International Workshop on Multimedia Signal Processing (MMSP'17)*, Luton, UK, Oct. 2017, pp. 1–6.

[36] A. Ghosh, V. Aggarwal, and F. Qian, "A robust algorithm for tile-based 360-degree video streaming with uncertain fov estimation," vol. abs/1812.00816, 2018. [Online]. Available: http://arxiv.org/abs/1812.00816

[37] M. Jamali, F. Golaghazadeh, S. Coulombe, A. Vakili, and C. Vazquez, "Comparison of 3d 360-degree video compression performance using different projections," in *Proc. of IEEE Canadian Conference of Electrical and Computer Engineering (CCECE'19)*, Edmonton, AB, Canada, Canada, May 2019, pp. 1–6.

[38] "Encoder settings for live 360-degree videos." [Online]. Available: https://support.google.com/youtube/answer/6396222

[39] "Under the hood: Building 360 video." [Online]. Available: https://engineering.fb.com/video-engineering/under-the-hood-building-360-video

[40] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications," in *Proc. of IEEE International Symposium on Multimedia (ISM'16)*, San Jose, CA, USA, Dec. 2016, pp. 583–586.

[41] J. Tang and X. Zhang, "Hybrid projection for encoding 360 vr videos," in *Proc. of IEEE Conference on Virtual Reality and 3D User Interfaces (VR'19)*, Osaka, Japan, Japan, Mar. 2019, pp. 440–447.

[42] A. De Abreu, C. Ozcinar, and A. Smolic, "Look around you: Saliency maps for omnidirectional images in vr applications," in *Proc. of Ninth International Conference on Quality of Multimedia Experience (QoMEX'17)*, Erfurt, Germany, May 2017, pp. 1–6.

[43] M. Almquist, V. Almquist, V. Krishnamoorthi, N. Carlsson, and D. Eager, "The prefetch aggressiveness tradeoff in 360 video streaming," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, p. 258–269.

[44] Y. Rai, J. Gutiérrez, and P. Le Callet, "A dataset of head and eye movements for 360 degree images," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ser. MMSys'17. New York, NY, USA: Association for Computing Machinery, Jun. 2017, p. 205–210.

[45] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "360^o video viewing dataset in head-mounted virtual reality," in *Proc. of the 8th ACM on Multimedia Systems Conference (MMSys'17)*, Taipei, Taiwan, Jun. 2017, pp. 211–216.

[46] F. Duanmu, Y. Mao, S. Liu, S. Srinivasan, and Y. Wang, "A subjective study of viewer navigation behaviors when watching 360-degree videos on computers," in *Proc. of IEEE International Conference on Multimedia and Expo (ICME'18)*, San Diego, CA, USA, Jul. 2018, pp. 1–6.

[47] C. Ozcinar and A. Smolic, "Visual attention in omnidirectional video for virtual reality applications," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, Cagliari, Italy, May 2018, pp. 1–6.

[48] S. Fremerey, A. Singla, K. Meseberg, and A. Raake, "Avtrack360: An open dataset and software recording people's head rotations watching 360 videos

on an hmd," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, p. 403–408.

[49] C. Wu, Z. Tan, Z. Wang, and S. Yang, "A dataset for exploring user behaviors in vr spherical video streaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ser. MMSys'17. New York, NY, USA: Association for Computing Machinery, Jun. 2017, p. 193–198.

[50] M. Assens, X. Giro-i-Nieto, K. McGuinness, and N. E. O'Connor, "Saltinet: Scan-path prediction on 360 degree images using saliency volumes," in *Proc. of IEEE International Conference on Computer Vision Workshops (ICCVW'17)*, Venice, Italy, Oct. 2017, pp. 2331–2338.

[51] A. D. Aladagli, E. Ekmekcioglu, D. Jarnikov, and A. Kondoz, "Predicting head trajectories in 360 virtual reality videos," in *Proc. of International Conference on 3D Immersion (IC3D'17)*, Brussels, Belgium, Dec. 2017, pp. 1–6.

[52] J. Ling, K. Zhang, Y. Zhang, D. Yang, and Z. Chen, "A saliency prediction model on 360 degree images using color dictionary based sparse representation," *Signal Processing: Image Communication*, vol. 69, Mar. 2018.

[53] Y. Kavak, E. Erdem, and A. Erdem, "A comparative study for feature integration strategies in dynamic saliency estimation," *Image Commun.*, vol. 51, no. C, p. 13–25, Feb. 2017.

[54] S. Rossi, F. De Simone, P. Frossard, and L. Toni, "Spherical clustering of users navigating 360 content," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'19)*, Brighton, United Kingdom, United Kingdom, May 2019, pp. 4020–4024.

[55] W. Lin, X. Zhang, Z. Guo, and W. Hu, "Opv: Bias correction based optimal probabilistic viewport-adaptive streaming for 360-degree video," in *Proc. of IEEE International Conference on Multimedia Expo Workshops (ICMEW'19)*, Shanghai, China, China, Jul. 2019, pp. 384–389.

[56] E. Jeong, D. You, C. Hyun, B. Seo, N. Kim, D. H. Kim, and Y. H. Lee, "Viewport prediction method of 360 vr video using sound localization information," in *Proc. of Tenth International Conference on Ubiquitous and Future Networks (ICUFN'18)*, Prague, Czech Republic, Jul. 2018, pp. 679–681.

[57] H. Hu, Z. Xu, X. Zhang, and Z. Guo, "Optimal viewport-adaptive 360-degree video streaming against random head movement," in *Proc. of IEEE*

International Conference on Communications (ICC'19), Shanghai, China, China, May 2019, pp. 1–6.

[58] Q. Yang, J. Zou, K. Tang, C. Li, and H. Xiong, "Single and sequential viewports prediction for 360-degree video streaming," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS'19)*, Sapporo, Japan, Japan, May 2019, pp. 1–5.

[59] J. Heyse, M. T. Vega, F. de Backere, and F. de Turck, "Contextual bandit learning-based viewport prediction for 360 video," in *Proc. of IEEE Conference on Virtual Reality and 3D User Interfaces (VR'19)*, Osaka, Japan, Japan, Mar. 2019, pp. 972–973.

[60] X. Chen, A. T. Z. Kasgari, and W. Saad, "Deep learning for content-based personalized viewport prediction of 360-degree vr videos," *IEEE Networking Letters*, pp. 1–1, Jun. 2020.

[61] X. Feng, Z. Bao, and S. Wei, "Exploring cnn-based viewport prediction for live virtual reality streaming," in *Proc. of IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR'19)*, San Diego, CA, USA, USA, Dec. 2019, pp. 183–1833.

[62] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proc. of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, ser. ATC '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, p. 1–6.

[63] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang, "Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming," in *Proc. of IEEE International Conference on Multimedia and Expo (ICME'18)*, San Diego, CA, USA, Jul. 2018, pp. 1–6.

[64] S. Petrangeli, G. Simon, and V. Swaminathan, "Trajectory-based viewport prediction for 360-degree virtual reality videos," in *Proc. of IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR'18)*, Taichung, Taiwan, Taiwan, Dec. 2018, pp. 157–160.

[65] M. Jeppsson, H. Espeland, C. Griwodz, T. Kupka, R. Langseth, A. Petlund, P. Qiaoqiao, C. Xue, K. Pogorelov, M. Riegler, D. Johansen, and P. Halvorsen, "Efficient live and on-demand tiled hevc 360 vr video streaming," in *Proc. of IEEE International Symposium on Multimedia (ISM'18)*, Taichung, Taiwan, Dec. 2018, pp. 81–88.

[66] R. Aksu, J. Chakareski, and V. Swaminathan, "Viewport-driven rate-distortion optimized scalable live 360 video network multicast," in

Proc. of IEEE International Conference on Multimedia Expo Workshops (ICMEW'18), San Diego, CA, USA, Jul. 2018, pp. 1–6.

[67] X. Liu, B. Han, F. Qian, and M. Varvello, "Lime: Understanding commercial 360° live video streaming services," in *Proc. of the 10th ACM Multimedia Systems Conference*, ser. MMSys '19. New York, NY, USA: ACM, Jun. 2019, pp. 154–164.

[68] Y. Hu, S. Xie, Y. Xu, and J. Sun, "Dynamic vr live streaming over mmt," in *Proc. of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB'17)*, Cagliari, Italy, Jun. 2017, pp. 1–4.

[69] J. Li, R. Feng, Z. Liu, W. Sun, and Q. Li, "Modeling qoe of virtual reality video transmission over wireless networks," in *Proc. of IEEE Global Communications Conference (GLOBECOM'18)*, Abu Dhabi, United Arab Emirates, United Arab Emirates, Dec. 2018, pp. 1–7.

[70] W. Sun, K. Gu, G. Zhai, S. Ma, W. Lin, and P. Le Calle, "Cviqd: Subjective quality evaluation of compressed virtual reality images," in *Proc. of IEEE International Conference on Image Processing (ICIP'17)*, Beijing, China, Sep. 2017, pp. 3450–3454.

[71] Z. Sinno and A. C. Bovik, "Large scale subjective video quality study," in *Proc. of 25th IEEE International Conference on Image Processing (ICIP'18)*, Athens, Greece, Oct. 2018, pp. 276–280.

[72] M. Vranjes, S. Rimac-Drlje, and D. Zagar, "Objective video quality metrics," in *ELMAR'07*, Zadar, Croatia, Sep. 2007, pp. 45–49.

[73] H. T. T. Tran, N. P. Ngoc, C. T. Pham, Y. J. Jung, and T. C. Thang, "A subjective study on qoe of 360 video for vr communication," in *Proc. of IEEE 19th International Workshop on Multimedia Signal Processing (MMSP'17)*, Luton, UK, Oct. 2017, pp. 1–6.

[74] S. Chen, Y. Zhang, Y. Li, Z. Chen, and Z. Wang, "Spherical structural similarity index for objective omnidirectional video quality assessment," in *Proc. of IEEE International Conference on Multimedia and Expo (ICME'18)*, San Diego, CA, USA, Jul. 2018, pp. 1–6.

[75] H. T. T. Tran, N. P. Ngoc, C. M. Bui, M. H. Pham, and T. C. Thang, "An evaluation of quality metrics for 360 videos," in *Proc. of the Ninth International Conference on Ubiquitous and Future Networks (ICUFN'17)*, Milan, Italy, Jul. 2017, pp. 7–11.

[76] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network - measurements, models, and implications," *Comput. Netw.*, vol. 53, no. 4, pp. 501–514, Mar. 2009.

[77] C. Ge, N. Wang, W. K. Chai, and H. Hellwagner, "Qoe-assured 4k http live streaming via transient segment holding at mobile edge," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1816–1830, Aug. 2018.

[78] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "Qoe-driven mobile edge caching placement for adaptive video streaming," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 965–984, Apr. 2018.

[79] A. Mehrabi, M. Siekkinen, and A. Ylä-Jaaski, "Qoe-traffic optimization through collaborative edge caching in adaptive mobile video streaming," *IEEE Access*, vol. 6, pp. 52 261–52 276, Sep. 2018.

[80] M. F. Tuysuz and M. E. Aydin, "Qoe-based mobility-aware collaborative video streaming on the edge of 5g," *IEEE Transactions on Industrial Informatics*, pp. 1–1, Feb. 2020.

[81] S. Zhang, N. Zhang, P. Yang, and X. Shen, "Cost-effective cache deployment in mobile heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 11 264–11 276, Dec. 2017.

[82] Z. Su, Q. Xu, F. Hou, Q. Yang, and Q. Qi, "Edge caching for layered video contents in mobile social networks," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2210–2221, Oct. 2017.

[83] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding performance of edge content caching for mobile video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1076–1089, May 2017.

[84] N. Carlsson and D. Eager, "Ephemeral content popularity at the edge and implications for on-demand caching," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1621–1634, Jun. 2017.

[85] S. Vakili, Q. Zhao, and Y. Zhou, "Time-varying stochastic multi-armed bandit problems," in *Proc. of 48th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, Nov. 2014, pp. 2103–2107.

[86] P. Blasco and D. Gündüz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *Proc. of IEEE International Symposium on Information Theory*, Honolulu, HI, USA, Jun. 2014, pp. 51–55.

[87] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Proc. of IEEE International Conference on Communications (ICC)*, Sydney, NSW, Australia, Jun. 2014, pp. 1897–1903.

[88] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in *Proc. of IEEE International Conference on Communications (ICC'16)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–7.

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

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

[91] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5g using reinforcement learning of space-time popularities," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, Feb. 2018.

[92] C. J. C. H. Watkins and P. Dayan, "Q-learning," pp. 279–292, 1992.

[93] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503–2516, Dec. 2016.

[94] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. of the 52nd Annual Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, USA, Mar. 2018, pp. 1–6.

[95] Y. Wei, Z. Zhang, F. R. Yu, and Z. Han, "Joint user scheduling and content caching strategy for mobile edge networks using deep reinforcement learning," in *Proc. of IEEE International Conference on Communications Workshops (ICC Workshops)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[96] J. Wang and I. C. Paschalidis, "An actor-critic algorithm with second-order actor and critic," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2689–2703, Jun. 2017.

[97] G. Dulac-Arnold, R. Evans, P. Sunehag, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *CoRR*, vol. abs/1512.07679, 2015. [Online]. Available: http://arxiv.org/abs/1512.07679

[98] V. N. Vasyukov and A. A. Spector, "Gauss markov sequences with gibbs distribution," in *Proc. of Third Russian-Korean International Symposium*

on Science and Technology. KORUS'99 (Cat. No.99EX362), vol. 1, Novosibirsk, Russia, Russia, Jun. 1999, pp. 253–256 vol.1.

[99] B. Dai and W. Yu, "Joint user association and content placement for cache-enabled wireless access networks," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'16)*, Shanghai, China, Mar. 2016, pp. 3521–3525.

[100] A. Khreishah and J. Chakareski, "Collaborative caching for multicell-coordinated systems," in *Proc. of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'15)*, Hong Kong, China, Apr. 2015, pp. 257–262.

[101] J. George and S. Sebastian, "Cooperative caching strategy for video streaming in mobile networks," in *Proc. of International Conference on Emerging Technological Trends (ICETT'16)*, Kollam, India, Oct. 2016, pp. 1–7.

[102] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 836–845, Apr. 2016.

[103] G. Van der Auwera, P. T. David, and M. Reisslein, "Traffic and quality characterization of single-layer video streams encoded with the h.264/mpeg-4 advanced video coding standard and scalable video coding extension," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 698–718, Sep. 2008.

[104] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian, "Overview of shvc: Scalable extensions of the high efficiency video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 20–34, Jan. 2016.

[105] Z. Ye, F. D. Pellegrini, R. El-Azouzi, L. Maggi, and T. Jimenez, "Quality-aware dash video caching schemes at mobile edge," in *Proc. of 29th International Teletraffic Congress (ITC 29)*, vol. 1, Genoa, Italy, Sep. 2017, pp. 205–213.

[106] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

[107] ILOG CPLEX optimization studio. http://is.gd/3GGOFp.

[108] L. Alexander, R. Johnson, and J. Weiss, "Exploring zipf's law," *Teaching Mathematics and Its Applications: International Journal of the IMA*, vol. 17, no. 4, pp. 155–158, Dec. 1998.

[109] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1300–1313, Jun. 2018.

[110] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *Proc. of IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Toronto, ON, Canada, Apr. 2014, pp. 1078–1086.

[111] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proc. of the 24th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '18. New York, NY, USA: ACM, Oct. 2018, pp. 99–114.

[112] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai, "A two-tier system for on-demand streaming of 360 degree video over dynamic networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 43–57, Mar. 2019.

[113] M. Xiao, C. Zhou, Y. Liu, and S. Chen, "Optile: Toward optimal tiling in 360-degree video streaming," in *Proc. of the 25th ACM International Conference on Multimedia*, ser. MM '17. New York, NY, USA: ACM, Oct. 2017, pp. 708–716.

[114] D. P. Bertsekas, *Dynamic Programming and Optimal Control.* Athena Scientific, 1995.

[115] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[116] R. E. Bellman, *Dynamic Programming.* Dover Publications, 2003.

[117] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[118] H. Li, T. Wei, A. Ren, Q. Zhu, and Y. Wang, "Deep reinforcement learning: Framework, applications, and embedded implementations," *CoRR*, vol. abs/1710.03792, 2017. [Online]. Available: http://arxiv.org/abs/1710.03792

[119] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam *et al.*, "Mastering

the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[120] X. Lei, X. Jiang, and C. Wang, "Design and implementation of streaming media processing software based on rtmp," in *Proc. of 5th Int. Congress on Image and Signal Processing*, Chongqing, China, Oct. 2012, pp. 192–196.

[121] G. Gualdi, R. Cucchiara, and A. Prati, "Low-latency live video streaming over low-capacity networks," in *Proc. of the 8th IEEE International Symposium on Multimedia (ISM'06)*, San Diego, CA, USA, Dec. 2006, pp. 449–456.

[122] [Online]. Available: https://engineering.fb.com/ios/under-the-hood-broadcasting-live-video-to-millions/

[123] T. Ergen and S. S. Kozat, "Online training of lstm networks in distributed systems for variable length data sequences," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 5159–5165, Oct. 2018.

[124] N. M. Vural, S. Ergut, and S. S. Kozat, "An efficient and effective second-order training algorithm for lstm-based adaptive learning," *CoRR*, vol. abs/1910.09857, 2019. [Online]. Available: https://arxiv.org/abs/1910.09857

[125] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[126] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "Making content caching policies "smart" using the deepcache framework," *SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 5, p. 64–69, Jan. 2019.

[127] E. Baccour, A. Erbad, A. Mohamed, K. Bilal, and M. Guizani, "Proactive video chunks caching and processing for latency and cost minimization in edge networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'19)*, Marrakesh, Morocco, Morocco, Apr. 2019, pp. 1–7.

[128] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 190–10 203, Nov. 2018.

[129] Z. Chen and M. Kountouris, "D2d caching vs. small cell caching: Where to cache content in a wireless network?" in *Proc. of IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC'16)*, Edinburgh, UK, Jul. 2016, pp. 1–6.

[130] S. Mukhopadhyay and Bindu Jain, "Multi-agent markov decision processes with limited agent communication," in *Proc. of the 2001 IEEE International Symposium on Intelligent Control (ISIC '01) (Cat. No.01CH37206)*, Mexico City, Mexico, Mexico, Sep. 2001, pp. 7–12.

[131] V. Akre, A. Abdulla, A. Rajan, A. Rashed, J. Ahamed, M. Mohammed, and O. Abdulla, "Virtual reality for medical science in the uae," in *Proc. of the Fifth HCT Information Technology Trends (ITT'18)*, Dubai, United Arab Emirates, Nov. 2018, pp. 325–330.

[132] Geheng Chen and Xiaowei Wang, "Intra-frame prediction algorithm based on the h.264/avc research and improvement," in *Proc. of International Conference on Computer, Mechatronics, Control and Electronic Engineering*, vol. 5, Changchun, China, Aug. 2010, pp. 338–340.

[133] J. Zhang, A. Moon, X. Zhuo, and S. W. Son, "Towards improving rate-distortion performance of transform-based lossy compression for hpc datasets," in *Proc. of IEEE High Performance Extreme Computing Conference (HPEC'19)*, Waltham, MA, USA, USA, Sep. 2019, pp. 1–7.

[134] P. Spachos, T. Lin, W. Li, M. Chignell, A. Leon-Garcia, J. Jiang, and L. Zucherman, "Subjective qoe assessment on video service: Laboratory controllable approach," in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Macau, China, Jun. 2017, pp. 1–9.

[135] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack, "Study of subjective and objective quality assessment of video," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1427–1441, Jun. 2010.

[136] R. E. Uhrig, "Introduction to artificial neural networks," in *Proc. of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, vol. 1, Orlando, FL, USA, USA, Nov. 1995, pp. 33–37 vol.1.

[137] E. Nishani and B. Çiço, "Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation," in *Proc. of 6th Mediterranean Conference on Embedded Computing (MECO'17)*, Bar, Montenegro, Jun. 2017, pp. 1–4.

[138] J. Wang and Z. Han, "Research on speech emotion recognition technology based on deep and shallow neural network," in *Proc. of Chinese Control Conference (CCC'19)*, Guangzhou, China, China, Jul. 2019, pp. 3555–3558.

[139] S. Marinai, M. Gori, and G. Soda, "Artificial neural networks for document analysis and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 23–35, Jan. 2005.

[140] A. Galindo-Serrano and L. Giupponi, "Distributed q-learning for aggregated interference control in cognitive radio networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 4, pp. 1823–1834, May 2010.