# COLLABORATIVE COMMUNICATIONS, CACHING, AND COMPUTING FOR CLOUD-ASSISTED 5G WIRELESS NETWORKS

**by**

**TUYEN X. TRAN**

**A dissertation submitted to the**

**School of Graduate Studies**

**Rutgers, The State University of New Jersey**

**In partial fulfillment of the requirements**

**For the degree of**

**Doctor of Philosophy**

**Graduate Program in Electrical and Computer Engineering**

**Written under the direction of**

**Professor Dario Pompili**

**And approved by**

_____

_____

_____

_____

**New Brunswick, New Jersey**

**MAY, 2018**

## ABSTRACT OF THE DISSERTATION

# Collaborative Communications, Caching, and Computing for Cloud-assisted 5G Wireless Networks

**By TUYEN X. TRAN**

**Dissertation Director:**
**Professor Dario Pompili**

Over the last few years, the proliferation of data-intensive applications on mobile devices has contributed to the overwhelming mobile traffic volume that is pushing against the boundary of the current communication networks' capacity. Additionally, the rapidly growing popularity of computation-intensive and latency-sensitive mobile services has placed severe demands on cloud infrastructures and wireless access networks such as ultra-low latency, user experience continuity, and high reliability. To keep up with these surging demands, network operators have to spend enormous efforts to improve users' experience while maintaining healthy revenue growth. While several solutions have been proposed to improve network capacity such as the deployment of ultra-dense small cells and massive antenna arrays as well as the utilization of millimeter wave spectrum bands, these approaches are fundamentally constrained by the limited spectrum resources, inter-cell interference, and control signaling overheads. Therefore, in order to support the foreseen massive demands from data- and computation-hungry users in the upcoming Fifth Generation (5G) of wireless systems in an affordable way, improving network capacity alone is not sufficient and has to be accompanied by innovations at higher layers.

To overcome the limitations of current connection-centric Radio Access Networks (RANs), cloud-assisted wireless networks are promising solutions that unite wireless networks and

cloud-computing to deliver cloud services directly from the network edges. The two emerging paradigms for cloud-assisted wireless networks are Cloud Radio Access Network (C-RAN), which aims at the centralization of base station (BS) functionalities via network virtualization and optical fronthaul technologies, and Mobile-Edge Computing (MEC), which proposes to empower the network edge by providing computing, storage, and networking resources within the edge of the mobile RAN. These two paradigms are complementary and have unique justifications within the 5G ecosystem: the centralized nature of C-RAN provides higher degree of cooperation in the network to address the capacity fluctuation and to increase the spectral and energy efficiency; on the other hand, the MEC paradigm is useful in reducing service latency and improving localized user experience.

The goal of this research is to leverage the emerging C-RAN and MEC paradigms to design disruptive innovations for the wireless access network that always make best use of the resources available to satisfy service requests from the users. To this end, novel cooperative frameworks are proposed to make optimized decisions for communications, caching, and computation in 5G wireless systems. The proposed innovative solutions include: (i) a joint user-centric radio clustering and beamforming scheme that maximizes the downlink sum throughput of a C-RAN system, (ii) a cooperative hierarchical caching framework that aims at minimizing the network cost of content delivery and at improving users' Quality of Experience (QoE) in a C-RAN, (iii) a joint collaborative caching and processing framework that enhances Adaptive Bitrate (ABR)-video streaming in a MEC network, and (iv) a joint computation offloading and resource allocation framework that helps improve users' computation experience by offloading their computation tasks to the MEC servers. The proposed innovations in this research can benefit a wide range of mobile applications and services such as video streaming, augmented reality (AR)/virtual reality (VR), Internet-of-Things (IoTs), public safety operations and real-time healthcare data analytics.

# Acknowledgements

I would like to express my deepest gratitude to my Ph.D. advisor, Dr. Dario Pompili, for his constant support, guidance, and encouragement throughout the course of my doctoral study and research. I have always been inspired by his in-depth interdisciplinary knowledge as well as by his vision and aspiration for high-quality research. From Dr. Pompili, I have learned how to always keep the bar high and to strive for the best possible achievement. I am honored to have been working under his supervision and I simply can't thank him enough for his constant advice on many aspects of a Ph.D. student's life and research career.

I would like to extend my gratitude to Drs. Emina Soljanin, Zoran Gajic, and Guosen Yue for serving as committee members in my qualifying exam first, in my thesis proposal defense later, and ultimately in my dissertation defense. Thank you very much for your valuable comments and suggestions on various aspects of my research since the early days of my Ph.D program. Special thanks goes to Guosen also for being my collaborator and mentor during my three summer internships with Huawei Technologies R&D in Bridgewater, NJ. His invaluable industrial experience has helped me make strong connections between theoretical research and real-world problems.

I am grateful to Dr. Predrag Spasojevic for his help and encouragement during my first semester at Rutgers as a Teaching Assistant in his class. I also thank him for being so generous to me with his time and expertise to discuss various research problems.

I would like to thank all the CPS Lab members and collaborators, Abolfazl Hajisami, Parul Pandey, Parsa Hosseini, Ayman Younis, Mehdi Rahmati, Vidyasagar Sadhu, and Xueyuan Zhao for the discussions and encouragement. My sincerest thanks to all the co-authors of my publications. I would like to also thank all my wonderful friends at Rutgers University, especially Viet Nguyen, Hai Nguyen, Cuong Tran, Long Le, Binh Pham, Trung Duong, Kien Dinh, and Hai Pham for making my Ph.D student life enjoyable.

I am grateful for the financial support from the Department of Electrical and Computer Engineering, Rutgers University and from the National Science Foundation (NSF) (Grant No. CNS-1319945), which have provided me with the necessary resources to carry on research and finish this dissertation.

Last, but not least, I would like to thank my parents, Mrs. Mai Nguyen and Mr. Vuong Tran, my wife Thao Nguyen, and my son Andrew Tran for the continued understanding and encouragement. Their unconditional love and support have given me the strength to chase my dreams and aspirations. To them, I dedicate this dissertation.

# Dedication

*To my parents,*

*my wife Thao, and our son Andrew*

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 5G Systems: Key Requirements and Technologies

Over the last few years, our daily lifestyle is increasingly exposed to a plethora of mobile services and applications for entertainment, business, education, healthcare, social networking, and so on. The proliferation mobile solutions in various domains with increasing requirement for rich computation and low-latency response has placed severe demands on cloud infrastructure and wireless access networks such as ultra-low latency, user experience continuity, and high reliability. At the same time, mobile data traffic is predicted to continue doubling each year [1] while fundamentally shifting from traditional connection-centric communications, such as phone calls and text messages, to user- and content-centric communications such as video streaming and content sharing. To keep up with these surging demands, network operators have to spend enormous efforts to improve users' experience while maintaining healthy revenue growth. The development of the upcoming Fifth Generation (5G) wireless systems is a cornerstone for realizing breakthroughs in the transformation of Information and Communications Technology (ICT) network infrastructure [2]. Overtime, any mobile app and any mobile service will be given the potential to connect to anything at anytime – from people and communities to physical things, processes, content, working knowledge, timely pertinent information and goods of all sorts in entirely flexible, reliable and secure ways.

**5G Requirements:** The three fundamental requirements for building 5G wireless networks include: (i) capabilities for supporting massive capacity and massive connectivity; (ii) support for an increasingly diverse set of services, application and users; and (iii) flexible and efficient use of all available spectrum, energy, and computation resources for wildly different network deployment scenarios. 5G is expected to enable connectivity for a wide

range of new use cases, including wireless connectivity for remote control of machinery, wireless connectivity for traffic safety and control, and monitor/control of infrastructure, etc. Furthermore, 5G should be flexible enough to enable connectivity also for future applications and use cases that may not even be partly anticipated. The very wide range of use cases to be covered by 5G implies that the capabilities of 5G wireless access have to extend far beyond that of previous generations. Some of the important performance target in 5G are highlighted below.

- *Data Rates:* Providing the possibility for much higher data rates will be important requirement in the 5G era, primarily as part of a quest for further enhanced mobile-broadband experience. In general, data rates of several 100 Mbit/s is expected in urban and suburban environments, with extremely high peak data rates of 10 Gbit/s.

- *Latency:* In terms of latency requirement, the possibility to provide an end-to-end latency in the order of 1 ms is often mentioned in the context of 5G. Such capability will enable new latency-critical wireless applications such as remote control with haptic feedback and wireless connectivity for traffic safety. The end-to-end latency requirement depends not only on the radio-access solution, but also on the physical distance between the end points.

- *Extreme Reliability:* Another important characteristic expected in the context of 5G is the possibility to enable connectivity with extremely high reliability. This might imply (i) the ability of the wireless-access solution to provide connectivity with extremely low error rate, for example, an error rate below $10^{-9}$; and/or (ii) the ability to retain connectivity even in case of unexpected events including natural disasters.

- *Low-cost Devices with Very Long Battery Life:* With the emergence of IoT, some applications such as the collection of data from a very large number of sensors, require the possibility for devices of much lower cost compared to the devices of today. In many cases, such applications also require the possibility for devices with extremely low energy consumption enabling battery life of several years. At the same time, these applications typically require only very modest data rates and can accept long latency.

- *Network Energy Efficiency:* Another important requirement emerging during the last few years has been the aim for significantly higher network energy efficiency. First, increased network energy efficiency is one important factor to reduce the operational cost of a network. Secondly, in many places where there is no easy access to the electrical grid, improving the energy efficiency of the base stations by means of using power from solar panels becomes a viable solution.

**5G Technologies:** The extremely higher aggregate data rates and the much lower latencies required by 5G is extremely difficult to achieved with a mere evolution of the status quo. This necessitates disruptive technologies that could lead to both architectural and component design chances, as outlined in the following [3, 4].

- *Cloud-assisted Wireless Networks:* To support the expected massive growth of mobile data, a large number of small cells are expected to be deployed indoors and outdoors, giving rise to ultra-dense network (UDN), which are considered to be the key path toward 5G. With such large-scale UDNs, network operators face many serious challenges in terms of operation and management, cost-effective small cell deployment, and inter-cell interference mitigation. To deal with those issues, cloud-based platforms are introduced to simplify the deployment, operation and management, and facilitate around-the-clock optimization of the network.

- *Millimeter Wave (mmWave):* While spectrum has become scarce at microwave frequencies, it is plentiful in the mmWave realm, ranging from 3 to 300 GHz. Many bands therein seem promising, including most immediately the local multipoint distribution service at 28–30 GHz, the license-free band at 60 GHz, and the E-band at 71–76 GHz, 81–86 GHz, and 92–95 GHz. Foreseeably, several tens of gigahertz could become available for 5G, offering well over an order of magnitude increase over what is available at present. Needless to say, work needs to be done on spectrum policy to render these bands available for mobile cellular.

- *Massive MIMO:* Massive multiple-input multiple-output (MIMO) proposes to utilize a very high number of antennas to multiplex messages for several devices on each time-frequency resource, focusing the radiated energy toward the intended directions

while minimizing intra- and inter-cell interference. Massive MIMO may require major architectural changes, particularly in the design of macro BSs, and it may also lead to new types of deployments.

- *Smarter devices:* Previous generations of cellular networks were designed with the premise of having complete control at the infrastructure side. However, it is expected that 5G systems should exploit intelligence at the device side within different layers of the protocol stacks, for example, by allowing device-to-device (D2D) connectivity or exploiting smart caching at the mobile terminals.

- *Native support for machine-to-machine (M2M) communication:* A native inclusion of M2M communication in 5G involves satisfying three fundamentally different requirements associated with different classes of low-data-rate services: support of a massive number of low-rate devices, sustaining a minimal data rate in virtually all circumstances, and very-low-latency data transfer. Addressing these requirements in 5G requires new methods and ideas at both the component and architectural levels.

## 1.2 Cloud-assisted Wireless Networks

5G systems will imply major changes in the implementation and deployment of networking infrastructure, based on software-defined networking (SDN) and network functions virtualization (NFV). Network operations and services are becoming cloud-enabled in almost every industry and it creates an apparent opportunity to generate value for the telecommunications industry from exploiting distributed storage and cloud computing towards specific clients and services. To overcome the limitations of current connection-centric Radio Access Networks (RANs), cloud-assisted wireless networks are promising solutions that unite wireless networks and cloud-computing to deliver cloud services directly from the network edges. The two emerging paradigms for cloud-assisted wireless networks are Cloud Radio Access Network (C-RAN) which aims at the centralization of base station (BS) functionalities via network virtualization and optical fronthaul technologies; and Mobile-Edge Computing (MEC) which proposes to empower the network edge by providing computing, storage, and networking resources within the edge of the mobile RAN.

### 1.2.1 Cloud Radio Access Network (C-RAN)

C-RAN was introduced as a revolutionary redesign of the cellular architecture to address the increase in data traffic and to reduce the capital expenditure (CAPEX) and operating expenditure (OPEX) [5, 6]. The idea of C-RAN is to decouple the computational functionalities from the distributed BS (a.k.a. eNodeB in LTE) and to consolidate them in a centralized processing center. Its main characteristics are: (i) centralized management of computing resources, (ii) reconfigurability of spectrum resources, (iii) collaborative communications, and (iv) real-time cloud computing on generic platforms.

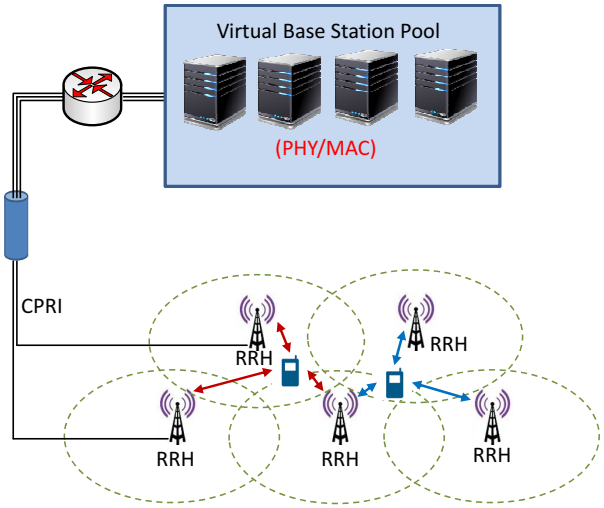



Figure 1.1: Illustration of a C-RAN.

**C-RAN Architecture.** As shown in Fig. 1.1, a typical C-RAN is composed of: (i) lightweight, distributed Radio Remote Heads (RRHs) plus antennae, which are located at the remote site and are controlled by a centralized virtual base station pool, (ii) the Base Band Unit (BBU) (also known as the VBS pool) composed of high-speed programmable processors and real-time virtualization technology to carry out the digital processing tasks, and (iii) low-latency high-bandwidth optical fibers, which connect the RRHs to the BBU pool. The communication functionalities of the VBSs are implemented on Virtual Machines (VMs) hosted over general-purpose computing platforms, which are housed in one or more racks of a small cloud datacenter. As a precautionary measure and to be on the safe side, the optical fiber transmission latency is limited to less than 1% of the PHY processing latency. Hence, the range of VBS pool is limited by latency constraints of wireless systems.

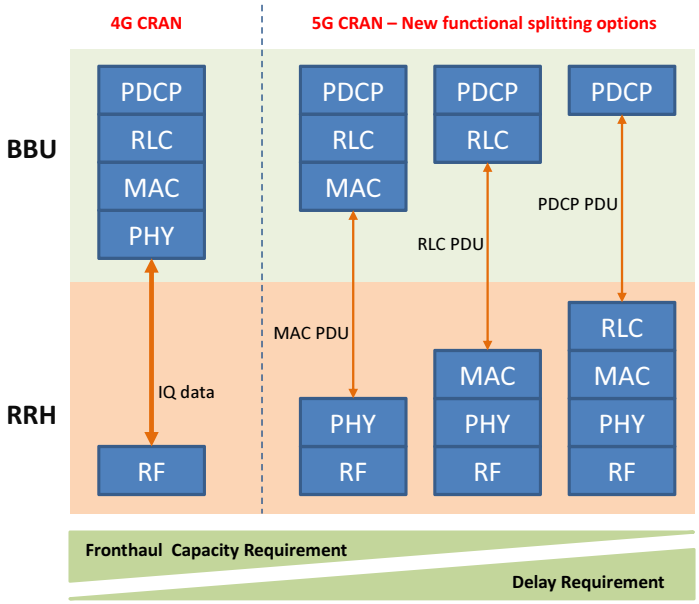



Figure 1.2: Functional split between BBU and RRH in 4G and 5G C-RANs. Shifting more functionalities to the RRH decreases capacity requirement and increases delay requirement on the fronthaul links.

The key concept of C-RAN is to offload computational functionalities from the RRHs to the BBU in the cloud in order to better utilize the shared computing resources that are co-located at the BBU pool, and to reduce the costs of densifying the RRHs. Depending on the fronthaul capacity and the processing capabilities of the BBU, different functional splitting between the RRHs are considered. Fig. 1.2 illustrates the functional split between BBU pool and RRH and compares the 4G C-RAN with the new options in 5G C-RAN.

The "full centralized" C-RAN architecture is realized when the RRH only carries the radio function. This option has has the advantages of easy upgrading and network capacity expansion; it also has better capability for supporting multi-standard operation, maximum resource sharing, and it is more convenient towards support of multi-cell collaborative signal processing [5]. Its major disadvantage is the high bandwidth requirement between the BBU and to carry the baseband I/Q signal. In the extreme case, a TD-LTE 8 antenna with 20MHz bandwidth will need a 10Gpbs transmission rate.

The "partial centralized" C-RAN architecture, where the RRH integrates not only the radio function but also the baseband function and possibly the higher layer functions, has the advantages of requiring much lower transmission bandwidth between BBU and RRH,

by separating the baseband processing from BBU and integrating it into RRH. Compared with the "full centralized" one, the BBU-RRH connection only need to carry demodulated data, which is only $1/20 - 1/50$ of the original baseband I/Q sample data. However, it also has its own shortcomings. Because the baseband processing is integrated into RRH, it has less flexibility in upgrading, and less convenience for multi-cell collaborative signal processing.

With either one of these C-RAN architectures, mobile operators can quickly deploy and make upgrades to their network. The operator only needs to install new RRHs and connect them to the BBU pool to expand the network coverage or split the cell to improve capacity. If the network load grows, the operator only needs to upgrade the BBU pool's HW to accommodate the increased processing capacity. Moreover, the "fully centralized solution", in combination with open platform and general purpose processors, will provide an easy way to develop and deploy software defined radio (SDR) which enables upgrading of air interface standards by software only, and makes it easier to upgrade RAN and support multi-standard operation. Different from traditional distributed BS architecture, C-RAN breaks up the static relationship between RRHs and BBUs. Each RRH does not belong to any specific physical BBU. The radio signals from /to a particular RRH can be processed by a virtual BS, which is part of the processing capacity allocated from the physical BBU pool by the real-time virtualization technology. The adoption of virtualization technology will maximize the flexibility in the C-RAN system [5].

**Advantages of C-RAN.** In a centralized VBS pool, since all the information from the BSs resides in a common place, the VBSs can exchange control data at Gbps rate. This centralized characteristic—along with virtualization technology and low-cost relay-like RRHs—provides a higher degree of freedom to make optimized decisions and has made C-RAN a promising technology candidate to be incorporated into the 5G wireless network, especially for urban/high-density areas. The VBSs in C-RAN are virtualized instances running on collocated computing servers, making it easy to share flexibly the common computing resources of the physical-server pool. Recent efforts [7–10] have implemented C-RAN prototypes and studied the computing resource consumption of the VBS pool with respect to the processing load. Their profiling results have demonstrated that the CPU frequency

needed to process the LTE subframes at the VBS in the allotted time (considered as 3 ms in [11]) increases with the effective data rate. By fitting a model of the processing time corresponding to different CPU frequencies, the computing resource consumption can be approximated as a linear increasing function of the user data rate [9, 10, 12]. By exploiting the global view of the network condition and traffic demand available at the BBU, dynamic provisioning and allocation of spectrum, computing, and radio resources can improve network performance [13–17]. Interestingly, C-RAN paves the way for bridging the gap between two so-far disconnected worlds: cellular communications and cloud computing. The list of other benefits brought by C-RAN can be summarized as follows [5, 18].

- *Energy Efficient:* Since in C-RAN a group of BSs are centralized in a common place, the number of cell sites can be reduced several folds. Hence, the air conditioning and power consumption of other site support equipments can be dramatically reduced. In addition, since the cooperative interference reduction techniques can be applied among the RRHs, a higher density of RRHs is allowed. Hence, smaller cells with lower transmission power can be deployed, thus aiming for higher frequency reuse and capacity, while the network coverage is not affected. Deploying small cells reduces the energy used for signal transmission, which is especially helpful to reduce the RAN power consumption and increase battery stand-by time. Lastly, because the BBU pool is an aggregated collective resource shared among a large number of virtual BSs, a much higher utilization rate of processing resources and lower power consumption can be achieved via statistical computing multiplexing.

- *Lower Operation and Maintenance Cost:* Because the BBUs and site support equipment are aggregated in a few big rooms, it is much easier for centralized management and operation, saving a lot of the operation and maintenance cost associated with the large number of BS sites in a traditional RAN network. Secondly, although the number of RRHs may not be reduced in a C-RAN architecture its functionality is simpler, size and power consumption are both reduced and they can sit on poles with minimum site support and management. The RRH only requires the installation of

the auxiliary antenna feeder systems, enabling operators to speed up the network construction to gain a first-mover advantage. Thus, operators can get large cost saving on site rental and operation and maintenance.

- *Capacity Improvement:* In C-RAN, VBSs are able to exchange the signaling, traffic data, and CSI of active MSs in the system with low latency. This way, it becomes much easier to implement joint processing and scheduling algorithms so to mitigate inter-cell interference and improve spectral efficiency [19–21]. For example, CoMP can efficiently be implemented under the C-RAN architecture.

- *Adaptability to Non-uniform Traffic:* C-RAN is also suitable to handle non-uniformly distributed traffic due to its intrinsic load-balancing capability in the centralized BBU pool [22]. Although the serving RRH changes dynamically according to the movement of the UEs, the serving BBU is still in the same BBU pool. As the coverage of a BBU pool is larger than in traditional BS, non-uniformly distributed traffic generated from UEs can be distributed in a VBS as this sits in the same BBU pool.

### 1.2.2 Mobile-Edge Computing (MEC)

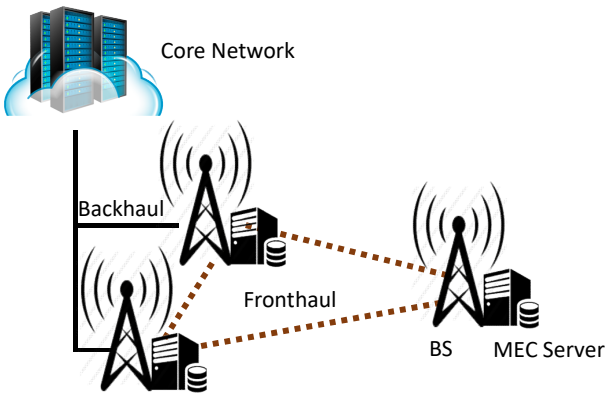



Figure 1.3: Illustration of a MEC network.

In the past decade, we have witnessed cloud computing play as significant role for massive data storage, control, and computation offloading. However, the rapid proliferation of mobile applications and the IoT over the last few years has posed severe demands on

cloud infrastructure and wireless access networks. Stringent requirements such as ultra-low latency, user experience continuity, and high reliability are driving the need for highly localized intelligence in close proximity to the end users. In light of this, MEC has been envisioned as the key technology to assist wireless networks with cloud computing-like capabilities in order to provide low-latency and context-aware services directly from the network edge.

Differently from traditional cloud computing systems where remote public clouds are utilized, the MEC paradigm is realized via the deployment of commodity servers, referred to as the MEC servers, at the edge of the wireless access network as shown in Fig. 1.3. Depending on different functional splitting and density of the BS, a MEC server can be deployed per BS or at an aggregation point serving several BSs. With the strategic deployment of these computing servers, MEC allows for data transfer and application execution in close proximity to the end users, substantially reducing end-to-end (e2e) delay and releasing the burden on backhaul network [23]. Additionally, MEC has the potential to empower the network with various benefits, including: (i) optimization of mobile resources by hosting compute-intensive applications at the network edge, (ii) pre-processing of large data before sending it (or some extracted features) to the cloud, and (iii) context-aware services with the help of the RAN information such as cell load, user locations, and radio resource allocation.

Fueled by the promising capabilities and business opportunities, the MEC paradigm has been attracting considerable attention from both academia and industry. A number of deployment scenarios, service use cases, and related algorithms design has been proposed to exploit the potential benefits of MEC and to justify its implementation and deployment from both a technical and business point of view. In the following, we briefly review the recent efforts from both standardization and research perspectives towards enabling MEC technologies in wireless networks.

**Proofs of Concepts and Standardization Efforts.** In 2013, Nokia Networks introduced the very first real-world MEC platform [24], in which the computing platform – Radio Applications Cloud Servers (RACS) – is fully integrated with the Flexi Multiradio BS. Saguna also introduced their fully virtualized MEC platform, so called Open-RAN [25], that can provide an open environment for running third-party MEC applications. Besides

these solutions, MEC standardization is being specified by the European Telecommunications Standards Institute (ETSI), which recently formed a MEC Industry Specifications Group (ISG) to standardize and moderate the adoption of MEC within the RAN. In the introductory white paper [26], four typical service scenarios and their relationship to MEC have been discussed, ranging from Augmented Reality (AR) and intelligent video acceleration to connected cars and IoT gateway. In the MEC World Congress 2016, ETSI has announced six Proofs of Concept (PoCs) that were accepted by the MEC ISG, including:

- Radio Aware Video Optimization in a Fully Virtualized Network (RAVEN).
- Flexible IP-based Services (FLIPS)
- Enterprise Services
- HealthcareDynamic Hospital User, IoT, and Alert Status Management
- Multi-Service MEC Platform for Advanced Service Delivery
- Video Analytics

These PoCs strengthen the strategic planning and decision making of organizations, helping them identify which MEC solutions may be viable in the network.

**MEC Architecture and Virtualization.** In recent years, the concept of integrating cloud computing-capabilities into the wireless network edge has been considered in the literature under different terminologies, including Small Cell Cloud (SCC), Mobile Micro Cloud (MMC), Follow Me Cloud (FMC), and CONCERT [27]. The basic idea of SCC is to enhance the small cells, such as microcells, picocells or femtocells, with additional computation and storage capabilities so as to support edge computing [28]. By exploiting the Network Function Virtualization (NFV) paradigm, the cloud-enabled small cells can pool their computation power to provide users with services/applications having stringent latency requirements. Similarly, the concept of MMC introduced in [29] allows users to have instantaneous access to the cloud services with low latency. Differently from the SCC where the computation/storage resources are provided by interworking clusters of enhanced small cells, the User Equipment (UE) exploits the computation resources of a single MMC, which

is typically connected directly to a BS. The FMC concept [30] proposes to move computing resources a bit further from the UEs, compared to SCC and MMC, to the core network. It aims at having the cloud services running at distributed data centers to be able to follow the UEs as they roam throughout the network. In all these described MEC concepts, the computing/storage resources have been fully distributed; conversely, the CONCERT concept proposes hierarchically placement of the resources within the network to flexibly and elastically manage the network and cloud services.

**Computation Offloading.** The benefits of computation offloading have been investigated widely in conventional Mobile Cloud Computing (MCC) systems. However, a large body of existing works on MCC assumed an infinite amount of computing resources available in a cloudlet, where offloaded tasks can be executed in negligible delay [31, 32]. Recently, several works have focused on exploiting the benefits of computation offloading in MEC network [33]. The problem of offloading scheduling was then reduced to radio resource allocation in [34], where the competition for radio resources is modeled as a congestion game of selfish mobile users. The problem of joint task offloading and resource allocation was studied in a single-user system with energy harvesting devices [35], and in a multi-cell, multi-user systems [36]; however, the congestion of computing resources at the MEC server was not taken into account. A similar problem is studied in [37] for single-server MEC systems, where the limited resources at the MEC server were factored in, and later on extended to multi-server MEC systems in [38].

**Edge Caching.** The increasing demand for massive multimedia services over mobile cellular network poses great challenges on network capacity and backhaul links. Distributed edge caching, which can well leverage MEC paradigm, has therefore been recognized as a promising solution to bring popular contents closer to the users, to reduce data traffic going through the backhaul links as well as the time required for content delivery, and to help smoothen/regulate the traffic during peak hours. Taking into account the heterogeneity of video transmissions in wireless networks in terms of video quality and device capabilities, our work in [39] proposes to utilize both caching and processing capabilities at the MEC servers to satisfy users' requests for videos with different bitrates. In this scheme, the collaborative caching paradigm has been extended to a new dimension where the MEC

servers can assist each other to not only provide the requested video via backhaul links but also to transcode it to the desired bitrate version.

### 1.2.3 C-RAN and MEC: Complementary Technologies for 5G

While the two technologies propose to move computing capabilities in a different direction (to the cloud versus to the edge), C-RAN and MEC are highly complementary technologies and their co-location will help make the economics of each of them significantly more attractive [40]. On the one hand, the centralized nature of C-RAN can be leveraged to address the capacity fluctuation problem and to increase system energy efficiency in mobile networks. The full centralization principle of C-RAN and the densification of cellular BSs, however, entails heavy exchanges of radio signals between the radio heads and cloud processing unit, which impose stringent requirement to the fronthaul connections in terms of throughput and latency. On the other hand, the MEC paradigm is useful in reducing latency and improving localized user experience. However, the amount of processing power and storage at each MEC server is in orders of magnitude below that of the centralized cloud in C-RAN, making the resource provisioning and allocation problem a critical challenge in MEC networks.

From a Mobile Network Operator's (MNO) point of view, collocating C-RAN and MEC helps support some of the key 5G applications that it would not be able to support otherwise. Firstly, a major challenge in enabling applications associated with the 5G use cases is the significant investment required to deploy a sufficiently extensive network of edge computing Points-of-Presence (PoPs), so that it becomes attractive to develop applications exploiting the edge processing infrastructure in mind. Moreover, this investment must be made in advance of applications being ready to take advantage of it, i.e., this is an investment in anticipation of future revenue, but without any guaranteed near-term returns. One way to mitigate the significant cost (and risk) of such strategic investment is to bootstrap a MEC deployment to the deployment of a C-RAN: the cost of providing additional processing power across an already planned BBU pool, should be significantly lower than a standalone MEC deployment. Conversely, deployment of a C-RAN across generic computing infrastructure (as opposed to dedicated, RAN-optimized hardware) is itself a significant investment for an

MNO. In addition to the costs of deploying C-RAN processing units themselves, there is the cost of moving towards virtualized RAN appliances, testing, integration and maintenance of these new solutions. While the operational flexibility and network re-configurability offered by virtualization may carry significant long-term benefits, the near-term effort and costs can make it a tough pill to swallow. The significant strategic benefits of MEC can make the decision a much clearer one [40].

## 1.3 Research Objectives and Contributions

The goal of this research is to design disruptive innovations for the wireless access network that always make best use of the resources available to satisfy service requests from the users. The new innovations should also make use of intelligence harvested from user and network context information such as the popular content being requested at a given time in a given location, the computing resources required to process baseband data and to execute computation tasks for each user. These additional information can enable the network to make optimized control decisions both proactively and reactively so as to improve the users' communications and computation experiences.

Fueled by the potential advantages of C-RAN and MEC, we aim at designing novel cooperative frameworks that optimize the control decisions for data transmissions, content provisioning, and computation in 5G systems. Specifically, our innovative solutions focus on improving downlink transmission throughput, reducing backhaul traffic load, and reducing end-to-end (e2e) latency for content delivery and mobile computation offloading. Our contributions in this dissertation are summarized in four specific topics as follows.

**1. Joint User-Centric Radio Clustering and Beamforming in C-RAN [14,41]:** In this work, we leverage the key capabilities of C-RAN in computing-resource sharing and real-time communications among the VBSs in order to design a joint user-centric radio clustering and beamforming scheme that maximizes the downlink Weighted Sum-Rate (WSR) performance. Due to the combinatorial nature of the radio clustering process and to the non-convexity of the joint beamforming design, the underlying optimization problem is NP-hard, and is extremely difficult to solve for a large network. The proposed approach

aims for a suboptimal solution by transforming the original problem into a Mixed-Integer Second-Order Cone Program (MI-SOCP) and applying Sequential Convex Approximation (SCA) to derive a novel iterative algorithm. Numerical simulation results show that our low-complexity algorithm provides near-optimal performance in terms of WSR while significantly outperforming conventional radio clustering and beamforming schemes. Additionally, the results also demonstrate the significant improvement in computing-resource utilization of C-RAN over a traditional RAN with distributed computing resources.

**2. Cooperative Hierarchical Caching and Request Scheduling in C-RAN [42, 43]:** We propose a novel cooperative hierarchical caching framework in C-RAN that aims at minimizing the network cost of content delivery and at improving users' Quality of Experience (QoE). A new cloud-cache layer in the Cloud Processing Unit (CPU) is proposed to bridge the storage-capacity/delay-performance gap between the traditional edge-based and core-based caching paradigms. Also, a delay-cost model is introduced to characterize and formulate the cache-placement optimization problem, which is shown to be NP-complete. Then, a low-complexity heuristic cache-management strategy comprising of a proactive cache-distribution algorithm and a reactive cache-replacement algorithm is proposed. A Cache-Aware Request Scheduling (CARS) algorithm is devised in order to optimize online the tradeoff between content download rate and content access delay. Via extensive numerical simulations—carried out using both real-world YouTube video requests and synthetic content requests—it is demonstrated that the proposed cache-management strategy outperforms traditional caching strategies in terms of cache hit ratio, average content access delay, and backhaul traffic load. Additionally, the proposed cache-aware content request scheduling algorithm achieves superior tradeoff performance over traditional approaches that optimize either users' rate or access delay alone.

**3. Joint Collaborative Caching and Processing in MEC Networks [39, 44]:** We propose a joint collaborative caching and processing framework that supports Adaptive Bitrate (ABR)-video streaming in a MEC networks. We formulate an Integer Linear Program (ILP) that determines the placement of video variants in the caches and the scheduling of video requests to the cache servers so as to minimize the expected backhaul cost of video retrieval. The considered problem is challenging due to its NP-completeness

and to the lack of a-priori knowledge about video request arrivals. Our approach decomposes the original problem into a cache placement problem and a video request scheduling problem while preserving the interplay between the two. We then propose practically efficient solutions, including: (i) a novel ABR-aware proactive cache placement algorithm with provable approximation performance when video popularity is available, and (ii) an online low-complexity video request scheduling algorithm that performs very closely to the optimal solution. Simulation results show that our proposed solutions achieve significant increase in terms of cache hit ratio and decrease in backhaul traffic and content access delay compared to the traditional approaches.

**4. Joint Computation Offloading and Resource Allocation in MEC Networks [38]:** In this work, a MEC enabled multi-cell wireless network is considered where each BS is equipped with a MEC server that assists mobile users in executing computation-intensive tasks via task offloading. The problem of Joint Task Offloading and Resource Allocation (JTORA) is studied in order to maximize the users' task offloading gains, which is measured by a weighted sum of reductions in task completion time and energy consumption. The considered problem is formulated as a Mixed Integer Non-linear Program (MINLP) that involves jointly optimizing the task offloading decision, uplink transmission power of mobile users, and computing resource allocation at the MEC servers. Due to the NP-hardness of this problem, solving for optimal solution is difficult and impractical for a large-scale network. To overcome this drawback, we propose to decompose the original problem into (i) a Resource Allocation (RA) problem with fixed task offloading decision and (ii) a Task Offloading (TO) problem that optimizes the optimal-value function corresponding to the RA problem. We address the RA problem using convex and quasi-convex optimization techniques, and propose a novel heuristic algorithm to the TO problem that achieves a sub-optimal solution in polynomial time. Simulation results show that our algorithm performs closely to the optimal solution and that it significantly improves the users' offloading utility over traditional approaches.

## 1.4 Dissertation Organization

The rest of this dissertation is organized as follows.

**Chapter 2** details our solution for dynamic radio cooperation for user-centric C-RAN. We provide details on the design of joint radio clustering and cooperative beamforming algorithm that maximize the downlink weighted sum-rate performance. We also discuss the limitation and practical considerations for the implementation of the proposed strategy in practice. Performance evaluation is presented via numerical simulations that demonstrate the superior performance of the proposed algorithm over existing approaches.

**Chapter 3** presents our proposed cooperative hierarchical caching and request scheduling strategies. A low-complexity cache management strategy that combines both proactive and reactive caching is proposed. Given the cache availability at each BS, an online cache-aware request scheduling algorithm is proposed to determine the user association decisions as new requests arrive. Simulation results using both real-world YouTube video requests and synthetic content requests demonstrate the advantages of using the proposed caching and request scheduling algorithms.

**Chapter 4** describes our novel framework for collaborative adaptive bitrate video caching and processing in MEC networks. The problem of joint collaborative caching and processing is formulated and efficient proactive and reactive cache-placement algorithms are proposed for the cases with known and unknown content popularity. An online request scheduling algorithm is proposed to make decision upon arrival of each new video request. Simulation results demonstrate the significant performance improvement of the proposed joint caching and processing scheme over traditional approaches.

**Chapter 5** shows our study of joint task offloading and resource allocation in a multi-server MEC network. The system model is explained and the task offloading problem is formulated. To derive a practical solution, the considered optimization problem is decomposed into a resource allocation problem, which can be solved using standard techniques, and a task offloading problem which can be solved using our proposed heuristic algorithm. Numerical simulations show that our proposed algorithm significantly improves the users' offloading gains over traditional approaches.

**Chapter 6** summarizes our main contributions and provides suggestions on future research directions that will push the state-of-the-art in cloud-assisted wireless networks.

# Chapter 2

# Joint User-centric Radio Clustering and Beamforming for C-RANs

In this chapter, we present the design of a novel dynamic radio-cooperation strategy for C-RANs. In particular, the key capabilities of C-RAN in computing-resource sharing and real-time communication among the VBSs are leveraged to design a joint dynamic radio clustering and cooperative beamforming scheme that maximizes the downlink Weighted Sum-Rate System Utility (WSRSU). Due to the combinatorial nature of the radio clustering process and to the non-convexity of the cooperative beamforming design, the underlying optimization problem is NP-hard, and is extremely difficult to solve for a large network. The proposed approach aims for a suboptimal solution by transforming the original problem into a Mixed-Integer Second-Order Cone Program (MI-SOCP) and applying Sequential Convex Approximation (SCA) to derive a novel iterative algorithm. Numerical simulation results show that our low-complexity algorithm provides near-optimal performance in terms of WSRSU while significantly outperforming conventional radio clustering and beamforming schemes. Additionally, the results also demonstrate the significant improvement in computing-resource utilization of C-RAN over a traditional RAN with distributed computing resources.

## 2.1 Introduction

Over the last few years, the proliferation of personal mobile-computing devices along with a plethora of data-intensive mobile applications has resulted in a tremendous increase in demand for ubiquitous and high-data-rate wireless communications. To cope with this challenge, the current trend in cellular networks is to densify the RAN by increasing the number of small cells and to leverage the cooperation among multiple antennae and BSs

via Coordinated Multi-Point (CoMP) transmission and reception techniques [45, 46]. In CoMP, it is important to select clusters of BSs and design beamforming vectors within each cluster so as to mitigate interference among users and thus increase the system throughput. Within this context, there have been a number of works proposing different BS clustering and cooperative beamforming techniques (see for example [47–51] and references therein). However, due the scarce interconnection among the BSs and the lack of global Channel State Information (CSI) at each BS, conventional clustering and cooperative beamforming techniques are rather simplistic, i.e., the clustering decision is made based on the relative signal strength and locations of the users, and the beamforming design does not account for inter-cluster interference.

In this work, we leverage C-RAN architecture to enable the dynamic adaptation of RRH clusters and cooperation within each cluster so to improve the overall network performance. Firstly, the co-location model of the VBSs allows for their real-time inter-communication, thus fully enabling a coordinated joint transmission of the RRHs that is currently practically constrained. In particular, control signals to realize CoMP between the BSs that traditionally travel via back-haul links can now be exchanged through the InfiniBand interconnection among the VBSs. A radio cooperation scheme deployed in C-RAN would be fully dynamic and user specific, in the sense that we can form a virtual cluster of RRHs to coordinate their downlink transmissions to each of the scheduled users. In this strategy, each scheduled user is always the central of a RRH cluster, making it different from the traditional CoMP techniques where the RRHs are grouped into fixed and non-overlapping clusters.

In traditional RANs, each BS is equipped with a fixed amount of computing resources for baseband processing and the BSs cannot share their resources with each other. As a result, computing resources will be the bottleneck at some BSs where traffic demand is high, and underutilized at other BSs where traffic demand is low. In contrast, the VBSs in C-RAN are virtualized instances running on collocated computing servers, making it easy to share flexibly the common computing resources of the physical-server pool. Recent efforts [7–10, 52] have implemented C-RAN prototypes and studied the computing resource (mainly CPU) consumption of the VBS pool with respect to the processing load. Their

profiling results have demonstrated that the CPU frequency needed to process the LTE subframes at the VBS in the allotted time (considered as 3 ms in [11]) increases with the effective data rate. By fitting a model of the processing time corresponding to different CPU frequencies, the computing resource consumption can be approximated as a linear increasing function of the user data rate [9, 10, 12]. In this article, we consider a general computing resource constraint at the VBS pool and design a feasible radio cooperation scheme.

In spite of its many promising advantages, C-RAN also poses various technical challenges in its design and deployment. Specifically, one must efficiently utilize the flexible processing resources in the VBS pool, design suitable communication schemes to leverage the cooperation among the RRHs, and minimize the overall energy consumption including the transmit power at the RRHs and power consumed for computing and cooling at the VBS pool. The overall system design and optimization in C-RAN is a complex problem involving research issues at multiple layers, for example at system-level, the Virtual Machine (VM, which holds the VBS) allocation strategy has to be energy-, thermal-, and user-QoS aware. Specifically, thermal awareness, which is the knowledge of heat generation and heat extraction at different regions inside the datacenter, is essential to maximize energy and cooling efficiency as well as to minimize server system failure rates. In [13, 53], the authors have proposed to employ thermal-aware VM consolidation to exploit the various benefits: (i) the energy spent on computation can be saved by turning off the unused physical servers after VM consolidation, (ii) the utilization of servers that are in the "better cooled" areas of the data centers (i.e., with high heat extraction) can be maximized, and (iii) according to thermodynamics, heat can be extracted more efficiently (i.e., by doing a lower amount of work) by the cooling system from the consolidated server racks, which are hotter than non-consolidated racks. While the main object of this work is on the physical-layer problem of designing coordinated RRHs transmission strategy, the holistic consideration of the overall system power consumption minimization is a subject for future investigation.

In wireless access networks, WSR system utility is a measure of accumulated users' throughput with consideration of their corresponding QoS priorities. Maximizing WSR has been an important subject in wireless resource management [54] and in recent studies in

C-RAN transmission design. In [55], the authors studied the joint optimization of antenna selection, regularization, and power allocation to maximize the average WSR. Along this line, works in [56,57] proposed sparse precoding/beamforming design for WSR maximization in C-RAN. In this article, we consider computing-resource constraint and investigate its influence on the scale of RRH cooperation and thus WSR performance of C-RAN system. In particular, we address a WSR maximization optimization problem under the computing-resource constraint at the VBS pool and the transmit power constraints at the RRHs.

**Main Contributions:** We proposed a *Dynamic Radio Cooperation (Dynamic-RC)* strategy that dynamically groups the RRHs into user-specific (potentially overlapping) clusters and designs the downlink beamformers at each RRH in order to maximize the WSRSU objective function. The underlying *Dynamic-RC* optimization problem that aims at maximizing the WSRSU under the transmit-power constraints at the RRHs and the total computing-resource constraint at the VBS pool is formulated. Due to the combinatorial nature of the radio-clustering process and to the non-convexity of the cooperative beamforming design, the *Dynamic-RC* problem is extremely difficult to solve in practical time. As such, our approach aims for a low-complexity solution that achieves near-optimal performance. In particular, we make the following contributions.

- We exploit *conic programming* techniques [58] and *sequential convex approximation* [59] in order to derive a novel iterative algorithm for the Dynamic-RC problem. In each iteration, we temporarily fix the clustering decision and solve the resulting *Cooperative Beamforming Design* (CBD) problem. The beamforming solution obtained from the CBD problem is used to adjust the clustering decision in the next iteration. As such, the joint clustering and beamforming design is quickly identified and is adaptive to the global network condition.

- While the CBD problem is NP-hard, we propose to first relax the computing-resource constraint and solve the *relaxed*-CBD problem by transforming it into an equivalent

SOCP problem. The obtained beamforming solution is verified against the computing-resource constraint and a user-rate-dropping process might be performed to finally obtain a suboptimal solution of the CBD problem. Furthermore, we present the Branch-and-Bound (BnB) method to optimally solve the *relaxed*-CBD problem. However, the BnB's complexity scales exponentially with problem size and thus it is mainly used for optimality benchmark.

- We propose to further reduce the complexity of Dynamic-RC via heuristically selecting the *candidate cluster* for each user before running the optimization algorithm, and performing clustering and beamforming updates in a two time-scale manner. Furthermore, we quantify the training overhead of CSI estimation required to perform Dynamic-RC and evaluate its impact on net-WSR performance.

- We carry out extensive numerical simulations in various user-distribution scenarios and show that our proposed *Dynamic-RC* strategy significantly improves the WSRSU performance over conventional radio clustering and beamforming schemes. Furthermore, the results also show the sizable gains of C-RAN using our *Dynamic-RC* strategy over distributed RAN in terms of computing resource and transmit-power utilization.

**Chapter Organization:** The remainder of this chapter is organized as follows: in Sect. 2.2, we discuss the related work; Sect. 2.3, we present the system model and formulate the problem under study; in Sect. 2.4 we present the proposed solution to the joint dynamic radio clustering and beamforming design; in Sect. 2.5, we discuss the practical considerations of the proposed approach; simulation results are illustrated in Sect. 2.6; finally, Sect. 2.7 concludes the chapter.

## 2.2 Related Work

Pioneering works on realizing the benefits of C-RAN have focused on the overall system architecture with emphasis on system issues, feasibility of virtual software base station stacks, performance requirements and analysis of optical links between RRHs and their

VBSs. For example, several LTE RAN prototypes have been implemented over General-Purpose Platforms (GPPs) such as the Intel solutions based on hybrid GPP-accelerator [60], Amarisoft solution [61], and OpenAirInterface platform [62]. Studies on these systems have demonstrated the preliminary potential benefits of C-RAN in improving statistical multiplexing gains, energy efficiency, and computing resource utilization. Field trial results in [5,63] show the feasibility of deploying C-RAN front-haul using CPRI compression, single fiber bidirection, and wavelength-division multiplexing.

On the other hand, considerable attention has been paid on cooperative communications techniques for C-RAN under various different objectives. For instance, the works in [64–67] consider the power minimization problem via jointly optimizing the set of active RRHs and precoding or beamforming design. The considered power models consist of the RRH transmission power [64], and additionally the user transmission power in [66], transport network power in [65], and power consumption at the VBS pool in [67]. In addition, the optimal tradeoff design between transmission power and backhaul capacity is studied in [68], while the tradeoff between transmission power and delay performance is investigated in [69, 70] via a cross-layer based approach. Furthermore, the works in [71, 72] address the front-haul uplink compression problem and [15] proposes a blind source separation method to mitigate uplink interference. Tackling the inter-operation between cloud computing and wireless networks, the authors in [73] jointly consider the spectrum efficiency in wireless networks and pricing information in the cloud. The problems of determining the price to charge for media services, resource allocation, and interference management are studied under the Stackelberg game model. Recently, Liao et. al [12, 74] have studied the impact of computing resource on the achievable sum-rate performance of a C-RAN system. In particular, the computing resource consumption by transmissions that involve machine-to-machine (M2M) communications is characterized in [12] while the work in [74] focuses on determining how much computing resource is needed given certain number of RRHs and user density. Different from these works, we take into account the computing resource constraint at the VBS pool while focusing on designing a novel joint RRH clustering and cooperative beamforming strategy.

With a similar focus as ours, the authors in [56, 75] study the problem of user-centric radio clustering for a C-RAN system, however, these solutions have a very slow convergence rate and in general do not hold a convergence guaranty. In contrast, our approach in this research aims at a low-complexity, fast-convergence solution with close-to-optimal performance in order to realize the practical implementation of such schemes.

## 2.3 System Model and Problem Formulation

In this section, we firstly introduce the system model of the considered downlink C-RAN system. The proposed dynamic radio cooperation strategy is then formulated as a joint clustering and beamforming design problem.

### 2.3.1 System Model

*System settings:* We consider a typical C-RAN consisting of multiple distributed RRHs that are connected to a VBS pool via low-latency, high-bandwidth CPRI links. The VBS pool is composed of high-speed programmable processors and real-time VMs to carry out PHY/MAC-layer functionalities.

Let $\mathcal{R} = \{1, 2, ..., R\}$ be the set of RRHs and $\mathcal{U} = \{1, 2, ..., U\}$ be the set of active users in the system. We assume that each RRH $r$ has $N_r$ antennae while, realistically, all the users are equipped with only a single antenna. Note that the solutions proposed here can be extended to the multi-antenna-user case, and to account for capacity-limited backhaul. The RRHs cooperate with each other to form virtual user-specific clusters, i.e., each RRH cluster is formed for a scheduled user, while each RRH can be part of multiple clusters. Hence, the number of virtual clusters is equal to the number of scheduled users in the system. Let $\mathcal{S} = \{s_u^r \,|u \in \mathcal{U}, r \in \mathcal{R}\}$ denote the clustering decision, in which $s_u^r$ is a binary variable equal to 1 if RRH $r$ is selected to serve user $u$, and 0 otherwise. Consequently, let $\mathcal{V}_u = \{r \in \mathcal{R} \,|s_u^r = 1\}$ denote the serving cluster of user $u$. We consider the system in a single time-frequency resource block, which is assumed to be spatially reused across all the users.

*Downlink transmissions:* We assume that each user has a single traffic flow that is statistically independent of all other users' flows. Baseband signals for user $u$ and the corresponding downlink beamforming information after being processed at the VBS pool will be transported to all the RRHs in the serving cluster $\mathcal{V}_u$. In each scheduling slot, all the RRHs in $\mathcal{V}_u$ will jointly transmit the normalized symbol $x_u \in \mathbb{C}$ of unit power to user $u$. It is assumed that the signals for different users are independent from each other and from the receiver noise. Now, let $\mathbf{w}_u^r \in \mathbb{C}^{N_r \times 1}$ be the linear downlink beamforming vector at RRH $r$ corresponding to user $u$ and $\mathbf{W} = \{\mathbf{w}_u^r \,|\forall u \in \mathcal{U}, r \in \mathcal{R}\}$ denote the network beamforming design. Note that $\mathbf{W}$ also implies the scheduling decision, i.e., user $u$, is not scheduled for the current time-frequency slot if $\mathbf{w}_u^r = \mathbf{0}, \forall r \in \mathcal{R}$. In the current scheduling slot, the received signal $y_u$ at user $u$ is,

$$y_u = \underbrace{\sum_{r \in \mathcal{V}_u} \mathbf{h}_u^r \mathbf{w}_u^r x_u}_{\text{desired signal}} + \underbrace{\sum_{u' \in \mathcal{U}, u' \neq u} \sum_{r' \in \mathcal{V}_{u'}} \mathbf{h}_u^{r'} \mathbf{w}_{u'}^{r'} x_{u'}}_{\text{interference}} + z_u, \tag{1}$$

where $\mathbf{h}_u^r \in \mathbb{C}^{1 \times N_r}$ is the channel coefficient vector from RRH $r$ to user $u$ and $z_u$ is the zero-mean circularly symmetric Gaussian noise denoted as $\mathcal{CN}(0, \sigma^2)$. For simplicity, let $\Psi_{u,u'} = \sum_{r' \in \mathcal{V}_{u'}} \mathbf{h}_u^{r'} \mathbf{w}_{u'}^{r'}$ and $\Psi_u = \Psi_{u,u} = \sum_{r \in \mathcal{V}_u} \mathbf{h}_u^r \mathbf{w}_u^r$. With this position, the received Signal-to-Interference-plus-Noise Ratio (SINR) at user $u$ simplifies to,

$$\gamma_u = \frac{|\Psi_u|^2}{\sum_{u' \in \mathcal{U}, u' \neq u} |\Psi_{u,u'}|^2 + \sigma^2}. \tag{2}$$

We consider that each user decodes its intended signal by treating all other interfering signal as noise. Without loss of generality, we assume that the spectral and the coding efficiencies of the downlink C-RAN system equal to 1. Thus, under the clustering decision $\mathcal{S}$ and the beamforming design $\mathbf{W}$, the normalized downlink data rate (bits/s/Hz) of user $u$ can be calculated as

$$R_u(\mathcal{S}, \mathbf{W}) = \log_2(1 + \gamma_u). \tag{3}$$

*Computing resource constraint:* In general, the computing-resource capacity of the VBS pool can be modeled as a multi-dimentional vector representing the capacities of the CPUs,

memory, and network interfaces. However, for the ease of analysis, in this work we only consider scalar computing capacity. From the profiling results [7,9,10] of the VBS processing time for LTE subframes, it has been suggested that the computing-resource consumption at the VBS grows with the traffic load. In general, the computing-resource required for each user $u$ can be expressed as $\Gamma(R_u)$, where $R_u$ is the data rate of user $u$ given in (3) and $\Gamma(.)$[1] is the characteristic (increasing) function specifying the relationship between the utilized computing resource and the processed user data rate. For example, in [12], the computing-resource consumption is approximated as a linear function of the user data rate, whereby $\Gamma(.)$ is given as $\Gamma(R_u) = \Gamma_b + \theta R_u$, in which $\Gamma_b$ is the basic computing part independent of the MCS and $\theta > 0$ is the slope. In this work, however, we consider a general realization of $\Gamma(.)$ that is applicable for any other models. In particular, let $C$ [cycles/s] denote the total computing capacity in the VBS pool, i.e., the sum capacity of all the CPUs allocated to the VBSs, which can be flexibly shared among all the VBSs. The computing-resource constraint on the accumulated data rate of all the users in the system can be expressed as,

$$\Gamma\left(\sum_{u\in\mathcal{U}} R_u\right) \leq C, \tag{4}$$

It should be noted that for a traditional system with distributed computing-resource at the RRHs, the accumulated data rate processed at each RRH $r$ will be subject to the per-RRH computing-resource constraint $C_r$, i.e.,

$$\Gamma\left(\sum_{u\in\mathcal{U}} s_u^r R_u\right) \leq C_r, \forall r \in \mathcal{R}. \tag{5}$$

### 2.3.2 Dynamic-RC Problem Formulation

Our objective is to maximize the WSRSU under the transmit-power constraint at each RRH and the total computing-resource constraint at the VBS pool. It is assumed that the capacity of the front-haul links connecting RRHs to the VBS pool is sufficiently provisioned

[1]The realization of $\Gamma(.)$ can be obtained by profiling the VBSs at different levels of offered load in a Cloud-RAN implementation.

to accommodate peak-capacity demand. Our proposed Dynamic-RC strategy involves finding the joint optimal clustering decision $\mathcal{S}^*$ and the beamforming design $\mathbf{W}^*$, and can be formulated as follows,

$$(\mathcal{S}^*, \mathbf{W}^*) = \underset{\substack{\{s_u^r, \mathbf{w}_u^r\} \\ r\in\mathcal{R}, u\in\mathcal{U}}}{\operatorname{argmax}} \sum_{u\in\mathcal{U}} q_u R_u(\mathcal{S}, \mathbf{W}) \tag{6a}$$

$$\text{s.t.} \quad \sum_{u\in\mathcal{U}} \|\mathbf{w}_u^r\|_2^2 \leq P_r, \forall r \in \mathcal{R}, \tag{6b}$$

$$\|\mathbf{w}_u^r\|_2^2 \leq s_u^r P_r, \tag{6c}$$

$$\Gamma\left(\sum_{u\in\mathcal{U}} R_u(\mathcal{S}, \mathbf{W})\right) \leq C, \tag{6d}$$

$$\sum_{u\in\mathcal{U}} s_u^r \leq M_r, s_u^r \in \{0,1\}, \tag{6e}$$

where $q_u$ is the utility marginal function corresponding to user $u$, which can represent the user-specific Quality of Service (QoS) or priority in the system, and $P_r$ [W] is the transmit-power constraint at RRH $r$. Constraint (6c) represents the coupling between the clustering variable $s_u^r$ and the beamforming vector $\mathbf{w}_u^r$, i.e., $\mathbf{w}_u^r = \mathbf{0}$ when $s_u^r = 0$ and $\mathbf{w}_u^r = \mathbf{1}$ when $s_u^r = 1$. Constraint (6e) ensures that each RRH $r$ only serves $M_r$ users at a time and this constraint is used to control the computational complexity as well as the CSI training overhead associated with the proposed solution.

We refer to problem in (6) as the *Dynamic-RC* problem, which is a Mixed-Integer Non-Linear Program (MINLP) and is intractable in practical time. Specifically, even when the binary variables $s_u^r$'s are fixed, solving for $\mathbf{w}_u^r$'s is still NP-hard. Given a large number of variables that scales linearly with the number of users and RRHs in the system, our main goal in this research is to design a low-complexity, suboptimal solution to the joint radio clustering and beamforming optimization problem as will be presented in the next Section.

## 2.4 Joint Dynamic Radio Clustering and Beamforming Design

In this section, in order to solve the *Dynamic-RC* problem efficiently, we firstly fix the clustering decision $\mathcal{S}$ and address the resulting problem, referred to as the CBD problem.

We present two approaches to solve the CBD problem using a proposed novel iterative SOCP algorithm in Sect. 2.4-A and using Branch-and-Bound algorithm in Sect. 2.4-B. The *Dynamic-RC* problem will then be solved in Sect. 2.4-C using a proposed iterative algorithm that make uses of the CBD solution to adjust the clustering decision at each iteration.

### 2.4.1 Cooperative Beamforming with Fixed Clustering Decision

In this subsection, we consider the problem of Cooperative Beamforming Design (CBD) for a given radio clustering decision $\mathcal{S}$. In particular, for given $s_u^r$'s satisfying constraints (6e), we need to find the optimal downlink beamformers $\mathbf{w}_u^r$'s by solving the CBD problem below.

$$\max_{\mathbf{w}_u^r, r\in\mathcal{R}, u\in\mathcal{U}} \sum_{u\in\mathcal{U}} q_u R_u\left(\mathcal{S}, \mathbf{W}\right) \tag{7a}$$

$$\text{s.t.} \quad \sum_{u\in\mathcal{U}} \|\mathbf{w}_u^r\|_2^2 \le P_r, \forall r \in \mathcal{R}, \tag{7b}$$

$$\Gamma\left(\sum_{u\in\mathcal{U}} R_u\left(\mathcal{S}, \mathbf{W}\right)\right) \le C. \tag{7c}$$

Observe that the rate functions $R_u$'s appear in *both* the constraint and objective of (7), making the problem difficult to deal with. To decouple this problem with respect to (w.r.t.) $R_u$'s, we temporarily remove the constraint (7c) and consider the CBD problem with constraint (7b) only. The solution $\{\tilde{\mathbf{w}}_u^r\}$ of the *relaxed*-CBD problem will be verified against constraint (7c) so to finally obtain the solution of the original CBD problem by solving an additional *feasibility* problem. In the following subsections, the *relaxed*-CBD first and then the *feasibility* problem will be addressed sequentially.

**Relaxed-CBD Problem.** The *relaxed*-CBD problem is rewritten from (7) without the computing-capacity constraint (7c), and is cast as follows,

$$\max_{\mathbf{w}_u^r, r\in\mathcal{R}, u\in\mathcal{U}} \sum_{u\in\mathcal{U}} q_u R_u\left(\mathcal{S}, \mathbf{W}\right) \tag{8a}$$

$$\text{s.t.} \quad \sum_{u\in\mathcal{U}} \|\mathbf{w}_u^r\|_2^2 \le P_r, \forall r \in \mathcal{R}. \tag{8b}$$

This is in fact a weighed sum-rate maximization problem, which is widely known to be NP-hard. Our approach aims for a local solution using a low-complexity algorithm designed by effectively exploiting the techniques of SOCP[2]. In order to use the efficient algorithms developed for SOCP, one must reformulate the problem into the standard form that the algorithms (e.g., those proposed in [76, 77]) are capable of dealing with.

Firstly, we rewrite the objective function (8a) using (3) as,

$$\sum_{u\in\mathcal{U}} q_u R_u\left(\mathcal{S},\mathbf{W}\right) = \sum_{u\in\mathcal{U}} \log_2(1+\gamma_u)^{q_u}. \tag{9}$$

Now, by introducing the variables $t_u$'s, we can recast the *relaxed*-CBD problem in (8) as,

$$\max_{\mathbf{w}_u^r, t_u, r\in\mathcal{R}, u\in\mathcal{U}} \prod_{u\in\mathcal{U}} t_u \tag{10a}$$

$$\text{s.t.} \quad \gamma_u \geq t_u^{1/q_u} - 1, \forall u \in \mathcal{U}, \tag{10b}$$

$$\sum_{u\in\mathcal{U}} \|\mathbf{w}_u^r\|_2^2 \leq P_r, \forall r \in \mathcal{R}, \tag{10c}$$

We can easily see that problem (10) is equivalent to problem (8) since the constraints in (10b) are active at the optimum [78] or, in other words, $\{\mathbf{w}_u^r, t_u\}$ are optimal for (10) if and only if $\{\mathbf{w}_u^r\}$ are optimal for (8) and $\gamma_u = t_u^{1/q_u} - 1$. We now introduce the following Lemma.

**Lemma 1.** *Let $\tilde{\mathbf{w}}_u^r = \mathbf{w}_u^r e^{j\nu_u^r}$, where $\nu_u^r$ is the phase rotation such that the imaginary part of $\mathbf{h}_u^r\tilde{\mathbf{w}}_u^r$ equals zero, $\forall u \in \mathcal{U}, r \in \mathcal{R}$. If $\mathbf{w}_u^r$ is the solution of Problem (10), then so is $\tilde{\mathbf{w}}_u^r$.*

*Proof.* We can express $\mathbf{h}_u^r\mathbf{w}_u^r$ as $\mathbf{h}_u^r\mathbf{w}_u^r = |\mathbf{h}_u^r\mathbf{w}_u^r|\, e^{j\theta_u^r}$. By choosing $\nu_u^r = -\theta_u^r$, we have $\mathbf{h}_u^r\tilde{\mathbf{w}}_u^r = \mathbf{h}_u^r\mathbf{w}_u^r e^{j\nu_u^r} = |\mathbf{h}_u^r\mathbf{w}_u^r|$. If we recall $\gamma_u$, given in (2), it is straightforward to verify that, by replacing $\mathbf{w}_u^r$ with $\tilde{\mathbf{w}}_u^r$, $\forall u \in \mathcal{U}, r \in \mathcal{R}$, the constraints in (10b) and (10c) still hold. Thus, if $\mathbf{w}_u^r$ is a solution of Problem (10), then $\tilde{\mathbf{w}}_u^r$ is also a solution. □

According to Lemma 1, we can restrict ourselves to the beamformers in which each product $\mathbf{h}_u^r\mathbf{w}_u^r \geq 0$, $\forall u \in \mathcal{U}, r \in \mathcal{V}_u$, has a non-negative real part and a zero imaginary

[2]Second-Order Cone Problems (SOCP) are convex-optimization problems in which a linear function is minimized over the intersection of an affine set and the product of second-order (quadratic) cones.

part. This does not affect the optimality of problem in (10). Notice that constraint (10b) is equivalent to

$$\frac{|\Psi_u|^2}{\sum\limits_{u' \in \mathcal{U}, u' \neq u} \left|\Psi_{u,u'}\right|^2 + \sigma^2} \geq t_u^{1/q_u} - 1, \forall u \in \mathcal{U}, \tag{11}$$

which can be recast as,

$$\Psi_u \geq \beta_u \sqrt{t_u^{1/q_u} - 1}, \forall u \in \mathcal{U}, \tag{12}$$

$$\text{and,} \qquad \sqrt{\sum_{u' \in \mathcal{U}, u' \neq u} \left|\Psi_{u,u'}\right|^2 + \sigma^2} \leq \beta_u, \forall u \in \mathcal{U}, \tag{13}$$

by introducing the slack variables $\beta_u$'s and considering that both constraints (12) and (13) are active at the optimum of problem (10). It can be verified that (10c) and (13) follow the Linear Programming (LP) constraint expression with generalized equalities/inequalities, which can be directly written as Second-Order Constraints (SOCs)[3] [58]. To deal with the non-convex constraint (12), we further exploit the sequential parametric convex-approximation approach in [59] to approximate (12) as a convex constraint. Firstly, (12) can be rewritten as,

$$\Psi_u \geq \beta_u \sqrt{\xi_u}, \forall u \in \mathcal{U}, \tag{14}$$

$$\xi_u + 1 \geq t_u^{1/q_u}, \forall u \in \mathcal{U}. \tag{15}$$

Observe that, for a given $\phi_u$, we have

$$\beta_u \sqrt{\xi_u} \leq \frac{\phi_u}{2} \beta_u^2 + \frac{\xi_u}{2\phi_u}, \tag{16}$$

which follows the inequality of arithmetic and geometric means of $\phi_u \beta_u^2$ and $\xi_u \phi_u^{-1}$. The equality in (16) is achieved when $\phi_u = \sqrt{\xi_u}/\beta_u$, leading to the equivalent form of constraint (14), i.e.,

$$\Psi_u - \frac{\xi_u}{2\phi_u} \geq \frac{\phi_u}{2} \beta_u^2, \forall u \in \mathcal{U}. \tag{17}$$

---

[3]In a SOC representation, the hyperbolic constraint $ab \geq c^2$, with $a, b \geq 0$, is equivalent to $||[(a-b) \quad 2c]^T||_2 \leq a + b$.

Furthermore, without loss of generality, we scale $q_u$'s in (7a) such that $q_u > 1, \forall u \in \mathcal{U}$ to make $t_u^{1/q_u}$ become concave. Thanks to the concavity of $t_u$'s, we can adopt the results in [59] to replace the right side of (15) by its iterative first-order approximation as,

$$t_u^{1/q_u} \leq {t_u^{(*)}}^{1/q_u} + \frac{1}{q_u} {t_u^{(*)}}^{(1/q_u)-1} \left( t_u - t_u^{(*)} \right), \tag{18}$$

where $t_u^{(*)}$ denotes the value of $t_u$ in the previous iteration. From (13), (17), and (18), the *relaxed*-CBD optimization problem in (8) can be finally recast as,

$$\max_{\mathbf{w}_u^r, t_u, r \in \mathcal{R}, u \in \mathcal{U}} \prod_{u \in \mathcal{U}} t_u \tag{19a}$$

$$\text{s.t.} \quad \sum_{u \in \mathcal{U}} \|\mathbf{w}_u^r\|_2^2 \leq P_r, \forall r \in \mathcal{R}, \tag{19b}$$

$$(13), (17), (18). \tag{19c}$$

Note that the objective function and all the constraints in (19) admit a SOC representation; in particular, the product of optimization variables in (19a) can be transformed into a set of hyperbolic constraints by collecting two variables at a time. To illustrate the transformation of the objective function, we consider a simple example with $U = 4$ users, i.e., $\mathcal{U} = \{1, 2, 3, 4\}$. In this case, we have $\prod_{u \in \mathcal{U}} t_u = t_1 t_2 t_3 t_4$, and the objective function in (19a) becomes equivalent to

$$\max \vartheta_1^0 \quad \text{s.t.} \quad \left(\vartheta_1^0\right)^2 \leq \vartheta_1^1 \vartheta_2^1, \ \left(\vartheta_1^1\right)^2 \leq t_1 t_2, \ \left(\vartheta_2^1\right)^2 \leq t_3 t_4, \tag{20a}$$

which can be expressed in SOC form as,

$$\max \vartheta_1^0 \tag{21a}$$

$$\text{s.t.} \quad \left\| \begin{bmatrix} 2\vartheta_1^0 & \left(\vartheta_1^1 - \vartheta_2^1\right) \end{bmatrix} \right\|_2 \leq \vartheta_1^1 + \vartheta_2^1, \tag{21b}$$

$$\left\| \begin{bmatrix} 2\vartheta_1^1 & (t_1 - t_2) \end{bmatrix} \right\|_2 \leq t_1 + t_2, \tag{21c}$$

$$\left\| \begin{bmatrix} 2\vartheta_2^1 & (t_3 - t_4) \end{bmatrix} \right\|_2 \leq t_3 + t_4. \tag{21d}$$

If $U = 2^M$, where $M$ is some positive integer, we can represent (19) in SOCP form as in (22); otherwise, if $U \neq 2^M$, by defining additional variables $t_k = 1$ for $k = U+1, ..., 2^{\lceil \log_2 U \rceil}$, where $\lceil x \rceil$ is the smallest integer not less than $x$, we can still apply the SOCP in (22).

$$\max_{\substack{\mathbf{w}_u^r, t_u, \xi_u, \beta_u, \vartheta_j^k \\ r\in\mathcal{R}, u\in\mathcal{U}}} \vartheta_1^0 \tag{22a}$$

$$\text{s.t.} \quad \left\| \begin{bmatrix} 2\vartheta_j^k & \left(\vartheta_{2j-1}^{k+1} - \vartheta_{2j}^{k+1}\right) \end{bmatrix} \right\|_2 \leq \vartheta_{2j-1}^{k+1} + \vartheta_{2j}^{k+1}, j = 1, ..2^k; k = 0, ... (M-2), \tag{22b}$$

$$\left\| \begin{bmatrix} 2\vartheta_j^{M-1} & (t_{2j-1} - t_{2j}) \end{bmatrix} \right\|_2 \leq t_{2j-1} + t_{2j}, j = 1, ...2^{M-1}, \tag{22c}$$

$$\left\| \begin{bmatrix} \frac{1}{2}\left(\Psi_u - \frac{\xi_u}{2\phi_u^{(n)}} - 1\right) & \beta_u \ \sqrt{\phi_u^{(n)}/2} \end{bmatrix} \right\|_2 \leq \frac{1}{2}\left(\Psi_u - \frac{\xi_u}{2\phi_u^{(n)}} + 1\right), \forall u \in \mathcal{U}, \tag{22d}$$

$$t_u^{(n)^{1/q_u}} + \frac{1}{q_u} t_u^{(n)^{(1/q_u)-1}} \left(t_u - t_u^{(n)}\right) \leq \xi_u + 1, \forall u \in \mathcal{U}, \tag{22e}$$

$$\|[\sigma, \Psi_{u,1} ... \Psi_{u,u-1}, \Psi_{u,u+1}, ... \Psi_{u,U}]\|_2 \leq \beta_u, \forall u \in \mathcal{U}, \tag{22f}$$

$$\sum\nolimits_{u\in\mathcal{U}} \|\mathbf{w}_u^r\|_2^2 \leq P_r, \forall r \in \mathcal{R}. \tag{22g}$$

In (22), $t_u^{(n)}$ and $\phi_u^{(n)}$ are the values of $t_u$ and $\phi_u$ in the $n$th iteration, respectively. The SOCP in (22) can be solved very efficiently and quickly using standard solvers. The iterative SOCP-based algorithm to solve the *relaxed*-CBD problem in (8) is described in Algorithm 1.

**Algorithm 1** Iterative SOCP for the Relaxed-CBD problem.

1: Initialize $n = 0$, and $t_u^{(n)}, \phi_u^{(n)}$ randomly.
2: **repeat**
3: Given $t_u^{(n)}, \phi_u^{(n)}$, solve (22) to obtain $\tilde{\mathbf{W}}$ and $\left\{t_u^{(n)*}, \beta_u^{(n)*}, \xi_u^{(n)*}\right\}$.
4: Set $\left\{t_u^{(n+1)}, \beta_u^{(n+1)}, \xi_u^{(n+1)}\right\} = \left\{t_u^{(n)*}, \beta_u^{(n)*}, \xi_u^{(n)*}\right\}$.
5: Set $\phi_u^{(n+1)} = \sqrt{\xi_u^{(n+1)}}/\beta_u^{(n+1)}$.
6: Set $\tilde{\mathbf{W}}^{(n)} = \tilde{\mathbf{W}}$, $n = n + 1$.
7: **until** convergence.

**Lemma 2.** *Algorithm 1 converges to a locally optimal solution of problem* (22).

*Proof.* Let $\left(\vartheta_1^0\right)_n$ be the out-come value of the objective function of problem (22) in the $n$th iteration. We will prove that $\left(\vartheta_1^0\right)_n$ increases over each iteration, and thus our proposed Algorithm 1 converges. Firstly, in the $n$th iteration, given the values of $\left\{t_u^{(n)}, \beta_u^{(n)}, \xi_u^{(n)}, \phi_u^{(n)}\right\}$,

denote $\mathcal{F}_n$ as the feasible set of $\{t_u, \beta_u, \xi_u\}$ satisfying constraints (22b)-(22g). The optimal solutions obtained from the $n$th iteration are $\left\{t_u^{(n)*}, \beta_u^{(n)*}, \xi_u^{(n)*}\right\}$ and, hence,

$$\left(\vartheta_1^0\right)_n = \vartheta_1^0 \Big|_{\{t_u,\beta_u,\xi_u\}=\left\{t_u^{(n)*},\beta_u^{(n)*},\xi_u^{(n)*}\right\}} . \tag{23}$$

Due to the update rules in steps 4, 5 of Algorithm 1, the solutions $\left\{t_u^{(n)*}, \beta_u^{(n)*}, \xi_u^{(n)*}\right\}$ constitute the feasible set in the next iteration, i.e.,

$$\left\{t_u^{(n)*}, \beta_u^{(n)*}, \xi_u^{(n)*}\right\} \subseteq \mathcal{F}_{n+1}. \tag{24}$$

Also, since $\left(\vartheta_1^0\right)_{n+1}$ is the optimal objective value in the $(n+1)$th iteration, we have,

$$\left(\vartheta_1^0\right)_{n+1} \geq \vartheta_1^0 \big|_{\{t_u,\beta_u,\xi_u\}\in\mathcal{F}_{n+1}} . \tag{25}$$

From (23), (24) and (25), we can conclude that $\left(\vartheta_1^0\right)_{n+1} \geq \left(\vartheta_1^0\right)_n$, or in other words, Algorithm 1 leads to the monotonic increasing of the objective function in (22). In addition, since problem (22) is upper bounded due to transmit power constraint, Algorithm 1 will converge to a locally optimal solution. □

**Remark 1**: (Feasible Initialization) It is crucial to generate the feasible values of $\{t_u, \phi_u\}$ that satisfy all the constraints to guarantee the feasibility and convergence of Algorithm 1. Therefore, we provide Routine 1 to generate the initial values that ensure feasibility in the first iteration. □

**Remark 2**: (Complexity Analysis) The computational complexity of Algorithm 1 mainly lies in step 3 where a SOCP problem (22) is solved. Assuming the same number of antennae $N_r$ on the RRHs, the total number of variables in this SOCP problem is $URN_r$. The computational complexity of the interior-point method to solve such a SOCP problem is approximately $\mathcal{O}\left(\left(URN_r\right)^{3.5}\right)$ [79], which is advantageous for a large network compared to the optimal design using existing solvers, which are characterized by an exponential-time complexity. □

**Routine 1** Generate Initial Values of $\{t_u, \phi_u\}$

1: Generate channel-matching beamforming vectors satisfying constraint (22g), i.e.,

$$\mathbf{w}_u^r = \frac{(\mathbf{h}_u^r)^H}{\|\mathbf{h}_u^r\|_2^2} \sqrt{\frac{\Pr}{\sum\limits_{u\in\mathcal{U}} s_u^r}}, \forall u \in \mathcal{U}, r \in \mathcal{R}. \tag{26}$$

2: Calculate $\beta_u$ and $\xi_u$ satisfying the equality in (22f) and (14), respectively, as,

$$\beta_u = \|[\sigma, \Psi_{u,1}...\Psi_{u,u-1}, \Psi_{u,u+1}, ...\Psi_{u,U}]\|_2, \forall u \in \mathcal{U}, \tag{27}$$

$$\xi_u = |\Psi_u|^2/\beta_u^2, \forall u \in \mathcal{U}. \tag{28}$$

3: Calculate $t_u$ and $\phi_u$ satisfying the equality in (15) and (17), respectively, as,

$$t_u = (\xi_u + 1)^{q_u}, \forall u \in \mathcal{U}, \tag{29}$$

$$\phi_u = \sqrt{\xi_u}/\beta_u, \forall u \in \mathcal{U}. \tag{30}$$

**CBD Feasibility Problem.** Here, the solution of the *relaxed*-CBD problem (8), which was obtained using Algorithm 1, will be verified against the computing-capacity constraint in (7c) to obtain finally the beamforming solution of the original CBD problem cast in (7).

Suppose that $\tilde{\mathbf{W}}$ is the beamforming solution of problems (8). If $\tilde{\mathbf{W}}$ satisfies the computing-resource constraint (7c), i.e., $\Gamma\left(\sum\limits_{u\in\mathcal{U}} R_u\left(\mathcal{S}, \tilde{\mathbf{W}}\right)\right) \leq C$, then $\tilde{\mathbf{W}}$ is also the optimal solution of (7). In this case, the WSRSU is limited by the per-RRH power budget only, and not by the computing-resource capacity of the VBS pool. Otherwise, when the computing-resource constraint is violated, we need to selectively drop the rates of some users. This can be done via a greedy user-rate-dropping algorithm that keeps dropping rates of the users having the smallest marginal utility function $q_u$ from the current scheduling interval until the total data rate of all the scheduled users satisfies the computing-resource constraint. Let $\tilde{\boldsymbol{\gamma}}$ be the set of SINR values and $\mathcal{A} = \{u \,|u \in \mathcal{U}, \gamma_u > 0\}$ be the set of scheduled users corresponding to the beamforming solution $\tilde{\mathbf{W}}$. For simplicity, let $R(\mathcal{A}) = \sum\limits_{u\in\mathcal{U}} R_u\left(S, \tilde{\mathbf{W}}\right) = \sum\limits_{u\in\mathcal{A}} \log_2(1+\gamma_u)$. The user-rate-dropping process is summarized in Routine 2.

Let $\{R_u^* \geq 0, u \in \mathcal{U}\}$ be the suboptimal user rates obtained after the greedy user-rate-dropping process is applied and $\gamma_u^* = 2^{R_u^*} - 1$; the beamformer design $\mathbf{W}$ that achieves these rates can be obtained via solving the feasibility problem given below,

$$\text{find} \quad \{\mathbf{w}_u^r\}, u \in \mathcal{U}, r \in \mathcal{V}_u \tag{31a}$$

$$\text{s.t.} \quad \sum_{u \in \mathcal{U}} \|\mathbf{w}_u^r\|_2^2 \leq P_r, \forall r \in \mathcal{R}, \tag{31b}$$

$$\frac{|\Psi_u|^2}{\sum\limits_{u' \in \mathcal{U}, u' \neq u} \left|\Psi_{u,u'}\right|^2 + \sigma^2} \geq \gamma_u^*, \forall u \in \mathcal{U}. \tag{31c}$$

**Routine 2** Greedy User-Rate-Dropping.

**repeat**: Set $u^* = \arg\min_{u \in \mathcal{A}} \{q_u\}$, $\mathcal{A}^* = \mathcal{A} \backslash \{u^*\}$.
  **if** $R(\mathcal{A}^*) > \Omega$ **then** Set $\gamma_{u^*} = 0$, $\mathcal{A} = \mathcal{A}^*$.
  **else** Set $\gamma' = 0$ and $\gamma'' = \gamma_{u^*}$.
    **repeat**: Set $\gamma_{u^*} = (\gamma' + \gamma'')/2$.
      **if** $\Gamma(R(\mathcal{A})) > C$ **then** Set $\gamma'' = \gamma_{u^*}$.
      **else** Set $\gamma' = \gamma_{u^*}$.
    **until** $\gamma'' - \gamma' < \epsilon_b$. With small tolerance $\epsilon_b > 0$.
**until** $R(\mathcal{A}) \leq \Omega$.

The feasibility problem in (31) is *not convex*; however, by exploiting its special structure, we can transform this problem into a SOCP form, which can be solved efficiently. The transformation is presented as follows. Firstly, notice that (31c) is equivalent to

$$(1 + 1/\gamma_u^*) |\Psi_u|^2 \geq \sum\nolimits_{u' \in \mathcal{U}} \left|\Psi_{u,u'}\right|^2 + \sigma^2, \forall r \in \mathcal{R}. \tag{32}$$

Since we consider that $\mathbf{h}_u^r \mathbf{w}_u^r \geq 0$, taking the square root of both sides in (32) yields,

$$\Psi_u \sqrt{1 + 1/\gamma_u^*} \geq \sqrt{\sum\nolimits_{u' \in \mathcal{U}} \left|\Psi_{u,u'}\right|^2 + \sigma^2} = \|[\Psi_{u,1}, ...\Psi_{u,U}, \sigma]\|_2. \tag{33}$$

It can be seen that (33) follows the SOC form. Furthermore, let $\mathbf{w}^r$ be the long column vector such that $[\mathbf{w}^r]^T = \left[(\mathbf{w}_1^r)^T, (\mathbf{w}_2^r)^T, ...(\mathbf{w}_U^r)^T\right]$, $\forall r \in \mathcal{R}$. We are now ready to recast the feasibility problem in (31) in the standard SOCP form as follows,

$$\text{find} \quad \{\mathbf{w}_u^r\}, u \in \mathcal{U}, r \in \mathcal{V}_u \tag{34a}$$

$$\text{s.t.} \quad \|\mathbf{w}^r\|_2 \leq \sqrt{P_r}, \forall r \in \mathcal{R}, \tag{34b}$$

$$\|[\Psi_{u,1}, ...\Psi_{u,U}, \sigma]\|_2 \leq \Psi_u \sqrt{1 + 1/\gamma_u^*}, \tag{34c}$$

The solution $\mathbf{W}^*$ for (34) can be obtained using standard SOCP techniques such as the interior-point methods [79] or the SOCP solvers (e.g., CPLEX, MOSEK).

### 2.4.2 Cooperative Beamforming Design via Branch-and-bound

In the previous subsection, the sub-optimal solution for the CBD problem can be obtained by solving the *relaxed*-CBD problem using the proposed iterative SOCP method in Algorithm 1 and solving the feasibility problem. In this section, we present the Branch-and-Bound (BnB) method to solve the *relaxed*-CBD problem in (8) to a globally optimal solution. While the BnB method generally has very high computational complexity, which grows exponentially with the problem size, we mainly use the resulting solution to benchmark the suboptimality of the proposed iterative SOCP solution.

The BnB method presented in the following is an extension of the method in [80] for a Multiple Input Single Output (MISO) network with non-cooperative BSs. Firstly, let us express problem in (8) in an equivalent form as,

$$\min_{\substack{\gamma_u, \mathbf{w}_u^r \\ u\in\mathcal{U}, r\in\mathcal{R}}} \sum_{u\in\mathcal{U}} -q_u \log_2(1+\gamma_u) \tag{35a}$$

$$\text{s.t.} \quad \gamma_u \leq \frac{|\Psi_u|^2}{\sum\limits_{u'\in\mathcal{U}, u'\neq u} |\Psi_{u,u'}|^2 + \sigma^2}, \forall u \in \mathcal{U} \tag{35b}$$

$$\sum_{u\in\mathcal{U}} \|\mathbf{w}_u^r\|_2^2 \leq P_r, \forall r \in \mathcal{R}. \tag{35c}$$

Let $\boldsymbol{\gamma} = [\gamma_1, ...\gamma_U]$. We denote the objective function and the feasible region of $\boldsymbol{\gamma}$ in (35), respectively, as

$$f(\boldsymbol{\gamma}) = \sum_{u\in\mathcal{U}} -q_u \log_2(1+\gamma_u), \tag{36}$$

$$\mathcal{G} = \{\boldsymbol{\gamma} | (35\text{b}), (35\text{c})\}. \tag{37}$$

The idea of the BnB algorithm is to generate a sequence of asymptotically tight *upper* and *lower bounds* for the objective function such that they both converge to a global optimal value. The algorithm starts with a known $U$-dimensional rectangle $\mathcal{Q}_{\text{init}}$ that contains the

feasible region $\mathcal{G}$, which can be specified as follows,

$$\mathcal{Q}_{\text{init}} = \left\{ \boldsymbol{\gamma} \middle| 0 \leq \gamma_u \leq P_r/\sigma^2 \sum_{r\in\mathcal{R}} \|\mathbf{h}_u^r\|_2^2 \,, \forall u \in \mathcal{U} \right\}. \tag{38}$$

It is easy to verify that $\mathcal{G} \subseteq \mathcal{Q}_{\text{init}}$. In each iteration, the lower and upper bounds are updated by partitioning $\mathcal{Q}_{\text{init}}$ into smaller rectangles. In order for the algorithm to converge, the bounds should be chosen such that they become tight as the number of partitions of $\mathcal{Q}_{\text{init}}$ increases. The iterative BnB algorithm terminates when the difference between the upper and lower bounds is within a predefined accuracy level $\epsilon$. For any rectangle $\mathcal{Q} = \{\boldsymbol{\gamma} \,|\gamma_{u,\min} \leq \gamma_u \leq \gamma_{u,\max} \,, \forall u \in \mathcal{U}\}$ such that $\mathcal{Q} \subseteq \mathcal{Q}_{\text{init}}$, we define the functions to calculate the lower and upper bounds as $f_{\text{lb}}(\mathcal{Q})$ and $f_{\text{ub}}(\mathcal{Q})$, respectively. For clarity, the BnB algorithm is summarized below (Algorithm 2).

**Algorithm 2** BnB algorithm for the Relaxed-CBD problem.

1: Initialize $\mathcal{Q}_{\text{init}}$ using (38) and optimality tolerance $\epsilon > 0$.
2: Set $\bar{\mathcal{Q}} = \mathcal{Q}_{\text{init}}$, $\mathcal{B} = \{\bar{\mathcal{Q}}\}$, $F_{ub} = f_{\text{ub}}(\bar{\mathcal{Q}})$, and $F_{lb} = f_{\text{lb}}(\bar{\mathcal{Q}})$.
3: **repeat**
4: Split $\mathcal{Q}$ along its longest edge into $\mathcal{Q}_I$ and $\mathcal{Q}_{II}$ using bisection subdivision.
5: Update $\mathcal{B} = (\mathcal{B} \backslash \{\mathcal{Q}\}) \cup \{\mathcal{Q}_I, \mathcal{Q}_{II}\}$.
6: Set $F_{ub} = \min_{\mathcal{Q}\in\mathcal{B}} \{f_{ub}(\mathcal{Q})\}$, $F_{lb} = \min_{\mathcal{Q}\in\mathcal{B}} \{f_{lb}(\mathcal{Q})\}$, $\bar{\mathcal{Q}} = \text{argmin}_{\mathcal{Q}\in\mathcal{B}} \{f_{lb}(\mathcal{Q})\}$.
7: **until** $F_{ub} - F_{lb} \leq \epsilon$. Return $\bar{\mathcal{Q}}$.

We use the bounding functions derived in [80], which can be expressed as,

$$f_{\text{ub}}(\mathcal{Q}) = \begin{cases} f(\boldsymbol{\gamma}_{\min}), & \boldsymbol{\gamma}_{\min} \in \mathcal{G}, \\ 0, & \text{otherwise;} \end{cases} \tag{39}$$

and

$$f_{\text{lb}}(\mathcal{Q}) = \begin{cases} f(\bar{\boldsymbol{\gamma}}), & \boldsymbol{\gamma}_{\min} \in \mathcal{G}, \\ 0, & \text{otherwise.} \end{cases} \tag{40}$$

In (40), $\bar{\boldsymbol{\gamma}} = [\bar{\gamma}_1, ...\bar{\gamma}_U]$ can be obtained using bisection search on each edge of the rectangle $\mathcal{Q}$ as described in Routine 3. Define the optimal value of $\boldsymbol{\gamma}$ for problem (35) as $\tilde{\gamma} = \inf_{\boldsymbol{\gamma}\in\mathcal{G}} f(\gamma)$. By using the bounding functions in (39) and (40), it is shown in [80–82] that Algorithm 2 will converge in a finite number of iterations to a solution arbitrarily close to $\tilde{\gamma}$.

**Routine 3** Bisection search for finding $\bar{\boldsymbol{\gamma}}$ for a given $\mathcal{Q}$.

1: **for** $u = 1 : U$ **do** Set $\mathbf{a} = \boldsymbol{\gamma}_{\min}$, $\mathbf{a}[u] = \mathbf{a}[u] + \boldsymbol{\gamma}_{\max}[u] - \boldsymbol{\gamma}_{\min}[u]$.
2: **if** $\mathbf{a} \in \mathcal{G}$ **then** Set $\bar{\boldsymbol{\gamma}}[u] = \boldsymbol{\gamma}_{\max}[u]$.
3: **else** Set $\boldsymbol{\gamma}' = \boldsymbol{\gamma}_{\min}$, $\boldsymbol{\gamma}'' = \mathbf{a}$, and tolerance $\epsilon_b > 0$.
4: **repeat**: Set $\mathbf{m} = (\boldsymbol{\gamma}' + \boldsymbol{\gamma}'')/2$.
5: **if** $\mathbf{m} \in \mathcal{G}$ **then** Set $\boldsymbol{\gamma}' = \mathbf{m}$.
6: **else** Set $\boldsymbol{\gamma}'' = \mathbf{m}$.
7: **until** $\|\boldsymbol{\gamma}' - \boldsymbol{\gamma}''\|_2 \leq \epsilon_b$. **return** $\bar{\boldsymbol{\gamma}}[u] = \mathbf{m}[u]$.
8: Return $\bar{\boldsymbol{\gamma}}$.

It should be noted that verifying whether $\boldsymbol{\gamma} \in \mathcal{G}$, which is required in (39) and (40) as well as in lines 4 and 10 of Routine 3, is equivalent to solving a feasibility problem to determine if the set of SINR values specified by $\boldsymbol{\gamma}$ are achievable and, if so, return a set of feasible beamforming vectors $\mathbf{w}_u^r$'s. Such a feasibility problem can be addressed using our method presented in Sec. 2.4.1.

### 2.4.3 Joint Dynamic Radio Clustering and Beamforming Design

In the previous subsections, we have addressed the CBD problem for a given clustering decision. Using the solution of the CBD problem, we now go back to address the original Dynamic-RC problem in (6), which involves making a *joint* radio clustering decision and beamforming design. Generally, in a network with $U$ users and $R$ RRHs, there are $2^{UR}$ possible clustering patterns. The optimal solution to the clustering decision could be found via exhaustive search or using standard global optimization solvers; such approaches, however, would have a prohibitive complexity growing exponentially with the problem size.

In this work, we leverage the efficient iterative SOCP algorithm proposed for the CBD problem in Sect. 2.4 to design a low complexity joint radio clustering and CBD algorithm. One method is to combine SOCP with mixed-integer programming. In particular, the framework developed in Algorithm 1, which iteratively solves problem (22), can be used to solve the Mixed-Integer SOCP (MI-SOCP) form of the original Dynamic-RC problem in (6). *This way of solving* (6) *is referred to as the MI-SOCP method in our simulations* where the MI-SOCP in (41) below with binary variables $s_u^r$'s, is solved in line 3 in each iteration of Algorithm 1.

$$\max_{\substack{\mathbf{w}_u^r, s_u^r, t_u, \xi_u, \beta_u, \vartheta_j^k \\ r\in\mathcal{R}, u\in\mathcal{U}}} \vartheta_1^0 \tag{41a}$$

$$\text{s.t.} \quad \|\mathbf{w}_u^r\|_2^2 \leq s_u^r P_r, \forall u \in \mathcal{U}, r \in \mathcal{R}, \tag{41b}$$

$$\sum_{u\in\mathcal{U}} s_u^r \leq M_r, s_u^r \in \{0,1\}, \tag{41c}$$

$$(22b) - (22g). \tag{41d}$$

The derivation of problem (22) becomes particular useful as it reduces each iteration to a MI-SOCP in (41), which can be solved efficiently using available solvers such as MOSEK. However, the complexity of this method is still too complex for practical systems. Another method of dealing with the joint radio clustering and beamforming problem is to employ $l1$-reweighing technique to find a sparse solution of the beamforming vectors, as explored in [41, 56]. However, this method does not hold a convergence guaranty.

Motivated by these shortcomings, we propose a low complexity method that can find a good feasible point via applying Sequential Convex Approximation (SCA) on the continuous relaxation of problem (41), which we referred to as the iterative SCA-SOCP method. The continuous relaxation of (41) allows $s_u^r$ to take any value in the interval $[0, 1]$; this makes the problem become a SOCP and can be solved directly. However, the resulting solution will contains many $s_u^r$'s that are infeasible to the original MI-SOCP (41) and thus it is very difficult to decide the clustering pattern. In the following, we will consider a tighter continuous relaxation that allows us to apply sequential convex approximation on the constraint (41b). Firstly, we can rewrite (41b) as $\|\mathbf{w}_u^r\|_2^2 / \mathrm{P}_r \leq s_u^r$. Following the approach in [83], we replace $s_u^r$ on the right side of the constraint with the tighter bound $(s_u^r)^2$ and consider an equivalent formulation of the continuous relaxation of (41) as,

$$\max_{\substack{\mathbf{w}_u^r, s_u^r, t_u, \xi_u, \beta_u, \vartheta_j^k \\ r\in\mathcal{R}, u\in\mathcal{U}}} \vartheta_1^0 \tag{42a}$$

$$\text{s.t.} \quad \|\mathbf{w}_u^r\|_2^2 / \mathrm{P}_r \leq (s_u^r)^2, \forall u \in \mathcal{U}, r \in \mathcal{R}, \tag{42b}$$

$$\sum_{u\in\mathcal{U}} s_u^r \leq M_r, \tag{42c}$$

$$(22b) - (22g). \tag{42d}$$

which follows from that fact that $(s_u^r)^2 \leq s_u^r, \forall s_u^r \in [0,1]$. We note that both sides of (42b) are convex and thus a SCA method can be applied. In particular, problem (42) can be solved iteratively, in which in the $(k+1)$th iteration, the constraint in (42b) is replaced by

$$\|\mathbf{w}_u^r\|_2^2 / \mathrm{P}_r \leq \left(s_u^{r(k)}\right)^2 + 2s_u^{r(k)}\left(s_u^r - s_u^{r(k)}\right), \forall u \in \mathcal{U}, r \in \mathcal{R}, \tag{43}$$

where $s_u^{r(k)}$ denotes the value of $s_u^r$ in the $k$th iteration. It should be noted that the right hand side of (43) is the linear approximation of $(s_u^r)^2$ in the $(k+1)$th iteration. We summarize the steps to solve problem (42) in Algorithm 3.

**Algorithm 3** SCA-based Algorithm for Problem (42).

1: Given $\{t'_u, \phi'_u\}, \forall u$.
2: Set $k = 0$, and randomly generate $s_u^{r(k)} \in [0,1], \forall u, r$.
3: **repeat**
4: Solve the problem below using standard SOCP solver to obtain $\left\{s_u^{r(k)*}, t_u^{(k)*}, \beta_u^{(k)*}, \xi_u^{(k)*}\right\}$

$$\max_{\substack{\mathbf{w}_u^r, s_u^r, t_u, \xi_u, \beta_u, \vartheta_j^k \\ r\in\mathcal{R}, u\in\mathcal{U}}} \vartheta_1^0 \tag{44a}$$

$$\text{s.t.} \quad \left\| \left[ \begin{array}{cc} \frac{1}{2}\left(\Psi_u - \frac{\xi_u}{2\phi'_u} - 1\right) & \beta_u \\ \sqrt{\phi'_u}/2 & \end{array} \right] \right\|_2 \leq \frac{1}{2}\left(\Psi_u - \frac{\xi_u}{2\phi'_u} + 1\right), \forall u \in \mathcal{U}, \tag{44b}$$

$$t_u'^{1/q_u} + \frac{1}{q_u} t_u'^{(1/q_u)-1}\left(t_u - t'_u\right) \leq \xi_u + 1, \forall u \in \mathcal{U}, \tag{44c}$$

$$(22b), (22c), (22f), (22g), (42c), (43). \tag{44d}$$

5: Set $s_u^{r(k+1)} = s_u^{r(k)*}$, $k = k+1$.
6: **until** convergence.
7: Return $\left\{t_u^{(k)*}, \beta_u^{(k)*}, \xi_u^{(k)*}\right\}$.

**Lemma 3.** *Algorithm 3 converges to a locally optimal solution of problem (42).*

*Proof.* The proof is similar to the proof of Lemma 2. Let $\mathcal{S}_k$ and $\left(\vartheta_1^0\right)_k$ be the feasible set of $\{s_u^r\}$ and the returned objection function, respectively, of problem (44) in the $k$th iteration. We have,

$$\left(\vartheta_1^0\right)_k = \vartheta_1^0 \Big|_{\{s_u^r, t_u, \beta_u, \xi_u\} = \left\{s_u^{r(k)*}, t_u^{(k)*}, \beta_u^{(k)*}, \xi_u^{(k)*}\right\}}. \tag{45}$$

Due to the linear approximation in (43) and the updating rule in Step 5 of Algorithm 3, it can be verified that the returned values $\left\{s_u^{r(k)*}\right\}$ in $k$th iteration are feasible for the problem in the next iteration, i.e.,

$$\left\{s_u^{r(k)*}\right\} \subseteq \mathcal{S}_{k+1}. \tag{46}$$

Moreover, since $\left(\vartheta_1^0\right)_{k+1}$ is the optimal objective value in the $(k+1)$th iteration, it holds that

$$\left(\vartheta_1^0\right)_{k+1} \geq \vartheta_1^0 \left|_{\{s_u^r, t_u, \beta_u, \xi_u\} \in \mathcal{S}_{k+1}}\right. . \tag{47}$$

From (45), (46), and (47), we have $\left(\vartheta_1^0\right)_{k+1} \geq \left(\vartheta_1^0\right)_k$. In other words, Algorithm 3 yields a non-deceasing sequence of the objective function. Due to the power constraint, the objective function is bounded above; hence, Algorithm 3 will converge to a locally optimal solution.

□

In summary, by combining SOCP and SCA, the original Dynamic-RC problem can be solved iteratively using Algorithm 1 as the outer-loop, within which, Step 3 solves problem (42) using Algorithm 3. After obtaining the continuous solutions of $s_u^r$'s, a simple mapping scheme can be used to set $M_r$ highest values of $s_u^r$'s to '1' and the rest to '0', $\forall r \in \mathcal{R}$. While characterizing the degree of suboptimality of the proposed iterative approach is a non-trivial task and thus beyond the scope of this research, in our simulations, we have compared the performance of Algorithm 1 and Algorithm 3 with the globally optimal counterparts using BnB and MI-SOCP, respectively.

## 2.5 Discussion on Practical Considerations

In the previous section, we have designed a low-complexity joint dynamic radio clustering and beamforming strategy to maximize the WSRSU performance in C-RAN system. Nevertheless, as with any CoMP-based techniques, the proposed scheme comes with the cost of increased optimization overhead, more channel estimation effort, and a tight synchronization requirement. In the following, we provide some discussion on these technical challenges and provide possible approaches to mitigate such issues. As mentioned earlier, the low

computational complexity of our proposed strategy is already significantly advantageous compared to existing approaches. However, it is expected that a dense C-RAN would manage a large number of users and RRHs; consequently, the proposed solution may become costly when making decision in every scheduling time-slot and for every possible user-RRH pair. In addition, it is shown that the training overhead for CSI estimation required to perform user-centric clustering and beamforming scales with the RRH cluster size [84]. Hence, we hereby discuss on how to further reduce the computational complexity and CSI training overhead of our proposed strategy.

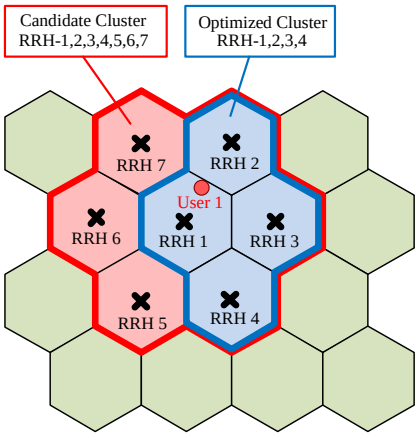



Figure 2.1: Example of *candidate* and *optimized* serving clusters for user 1.

**Candidate Cluster Selection.** Observe that, in practice, a RRH $r$ should not be included in the serving cluster of user $u$ if the channel gain between $r$ and $u$ is very weak. Considering a network of hexagonal cells, we can pre-select a small group $\mathcal{C}_u$ consisting of RRHs that have high potential to be included in the optimized serving cluster $\mathcal{V}_u$ of user $u$. Here, we refer to $\mathcal{C}_u$ as the *candidate cluster* of user $u$, which is formed by selecting $V_{\max}$ RRHs having the strongest channel gains to user $u$. This $V_{\max}$ parameter is used to control the maximum candidate cluster size. The candidate cluster selection process can be run instantly as we only need to sort the channel gains for each user in a descending order. After that, Dynamic-RC algorithm will identify the optimal serving cluster $\mathcal{V}_u$ as a subset of $\mathcal{C}_u$. For example, Fig. 2.1 illustrates the *candidate cluster* and *optimized cluster* of user 1 when we set $V_{\max} = 7$. In particular, the candidate cluster of user 1 is pre-selected to include RRHs $\{1, 2, 3, 4, 5, 6, 7\}$; under the current channel condition, the solution of Dynamic-RC yields

the optimized cluster for user 1 consisting of RRHs $\{1, 2, 3, 4\}$. Note that the candidate clusters implicitly impose bias constraints on Dynamic-RC problem, i.e., $s_u^r = 0, \forall r \notin \mathcal{C}_u$, resulting in *small* performance loss as will be shown in our simulations. However, this significantly reduces the complexity of our proposed algorithm from $\mathcal{O}\left((URM_r)^{3.5}\right)$ to $\mathcal{O}\left((UV_{\max}M_r)^{3.5}\right)$, where $V_{\max}$ is much smaller than $R$.

**Training Resource Allocation.** The clustering and beamforming design in Dynamic-RC are based on instantaneous CSI and thus it requires estimation of the downlink channels in every time slot. In a TDD system, this can be done through uplink training by exploiting the reciprocity between uplink and downlink channels. The uplink training overhead is characterized by (i) the percentage of resources taken by the uplink training of each user in TDD systems, denoted as $\eta$, and (ii) the occupied uplink resources, denoted as $T$, to ensure the orthogonality among the training signals of multiple users connecting to the same RRH [84]. Taking into account the uplink training overhead, the *net* downlink data rate of user $u$ can be expressed as,

$$\widehat{R_u}\left(\mathcal{S}, \mathbf{W}\right) = \left(1 - \eta T\right) R_u\left(S, \mathbf{W}\right). \tag{48}$$

According to the LTE specification [85], $\eta$ is set to 1% for 10 ms training period. While increasing $\eta$ leads to more accurate channel estimation, a large value of $\eta$ will cost more system resources and eventually decreases the net downlink data rate. On the other hand, the value of $T$ depends on the cluster formation and it should be chosen so that the training signals of all users connected to the same RRH are orthogonal in order for the RRH to distinguish the users. Intuitively, if we select large candidate clusters for the users, each RRH needs to perform uplink training with more users, resulting in a large value of $T$. For a user-centric RRH clustering scheme, $T$ can be computed by solving a graph coloring problem on a graph representation of the network where each vertex represents a user and each edge represents two users sharing at least one RRH in their clusters [84].

**Two-time-scale Dynamic-RC.** Here, we propose to perform Dynamic-RC in a two-time-scale manner. In particular, at the beginning of each *large* time-slot, the clustering

decision is updated based on channel statistics to adapt to large-scale fading due to shadowing. Within each large time-slot, the beamforming solution is computed in every *small* scheduling slot based on instantaneous CSI, which is changing much faster. Once the candidate clusters are selected, each user only needs to feed back the instantaneous CSI to the RRHs in its candidate cluster. Therefore, this strategy significantly reduces the overhead of CSI acquisition process during each large time-slot.

**Impact of Synchronization Errors.** In C-RAN, each RRH is equipped with a local clock with individual synchronization parameters. In order to facilitate coherence cooperation, the clock boards of the RRHs must be synchronized to a common external reference clock (for example via GPS). In particular, these RRHs' clocks have to be synchronized in frequency such that Inter Carrier Interference (ICI) be avoided, and in time in order to avoid both ICI and Inter Symbol interference (ISI) [86]. In this work, we consider the downlink C-RAN system operating in a single frequency band, and assume that the RRHs are precisely synchronized. However, in practice, if not carefully designed, the system performance may be compromised by the time-synchronization errors. These errors, referred to as *clock jitters*, are the differences in time between the individual RRH clocks and the reference clock. The effect of clock jitters at the RRHs is that the composite pulse shape (sum of the pulses from each RRH antenna shifted by the corresponding clock jitters) seen at the user's receiver will no longer be Nyquist [87]. Hence, the neighboring bits will cause ISI, reducing the mean of the received signal and increasing the variance of the noise. Consequently, the average SINR at the users will be reduced and so the system WSR performance. Therefore, considering the design of a good RRH synchronization strategy is important in a future study in order to preserve the cooperation gains.

## 2.6 Performance Evaluation

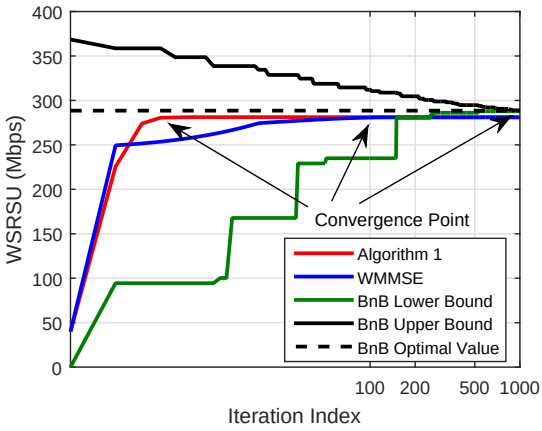
In this section, simulation results are presented to evaluate the performance of our proposed *Dynamic-RC* strategy. We consider a C-RAN system consisting of multiple hexagonal cells with a RRH in the center of each cell. The neighboring RRHs are 1 Km apart from each other. We assume that all the wireless channels in the system experience *block fading* such that the channel coefficients stay constant during each scheduling interval but can vary

from interval to interval, i.e., the *channel coherence time* is not shorter than the scheduling interval. We assume that all the RRHs have the same number of transmit antennae $N_r$ and transmit power budget $P_r = P, \forall r$. To ensure fair comparison with the baselines, we also set $M_r = N_r, \forall r$. The channel coefficients are calculated following the path-loss model, given as $L$ [dB] $= 148.1 + 37.6 \log_{10} d_{[\text{km}]}$, and the log-normal shadowing variance set to 8 dB. In addition, it is assumed that the channel bandwidth $B$, reused across all the users, is 10 MHz, and that the noise power is $-100$ dBm. Unless otherwise stated, the maximum candidate cluster size is set to $V_{\max} = 7$ and the utility marginal functions $q_u$'s are generated randomly such that $0 < q_u \leq 1, \forall u \in \mathcal{U}$.

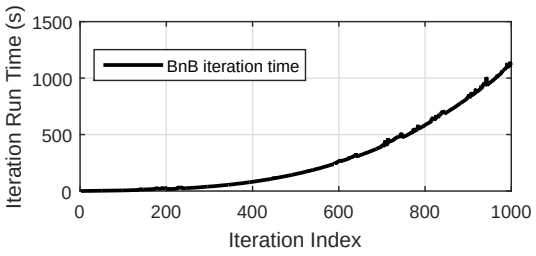
### 2.6.1 Convergence Rate of Proposed Beamforming Algorithm

Firstly, we evaluate the performance of Algorithm 1 in yielding the CBD solution, compared to that of the optimal BnB method in Algorithm 2 and of a popular method so-called iterative Weighted Minimum Mean Square Error (WMMSE) [56, 88, 89]. Since the computational complexity of the BnB method is high, it is difficult to solve the CBD problem with a large number of variables. Hence, we carry out the comparison in a small network as reported in Fig. 2.2(a, b, c) in which we set $U = 4$, $R = 4$, $N_r = 4$, $P = 10$ dBm (for (a) and (b)), $s_u^r = 1, \forall u, r$ and the optimality tolerance set to $\epsilon = 10^{-3}$;

We generate one random channel realization and set the same initial point for both Algorithm 1 and WMMSE. We observe from Fig. 2.2(a) that the objective value (WSRSU) obtained from Algorithm 1 is comparable to that of WMMSE and very close to the optimal value obtained via BnB method. Iteration run time until convergence of the BnB algorithm for one random channel realization is also depicted in Fig. 2.2(b). Additionally, in Fig. 2.2(b), we record the average iteration number and iteration run time of Algorithm 1 and WMMSE over 100 random channel realizations. It can be seen that the BnB method takes extremely long time to converge and is clearly not a practical solution. The solution of Algorithm 1 converges the fastest, at an order of magnitude faster than that of the WMMSE algorithm both in average number of iterations and average iteration run time. This fast-convergence performance of Algorithm 1 is very important for the practical feasibility of Dynamic-RC since we want to optimize the beamforming design in each scheduling

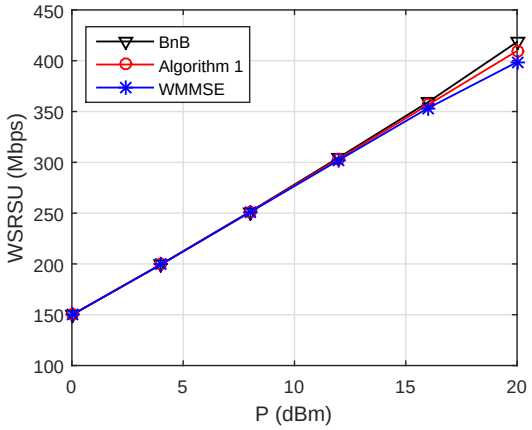



(a) Convergence behavior




| | Avg. Iter. Run Time (s) | Avg. Iter. Num. |
|---|---|---|
| Algorithm 1 | 0.1119 | 7.96 |
| WMMSE | 0.5058 | 58.7 |

(b) Iteration run time comparison




(c) WSRSU performance comparison

Figure 2.2: Simulations on a small C-RAN network: (a) Convergence behavior; (b) Iteration run time comparison; (c) WSRSU performance comparison.

slot. In Fig. 2.2(c), we compare the CBD solution of the three algorithms over different values of transmit power $P$ at the RRHs, each with 100 random channel realizations. It can be seen that when $P$ is small all three algorithms perform the same; conversely, at higher $P$ Algorithm 1 slightly outperforms WMMSE while both are very close to the optimal BnB performance.

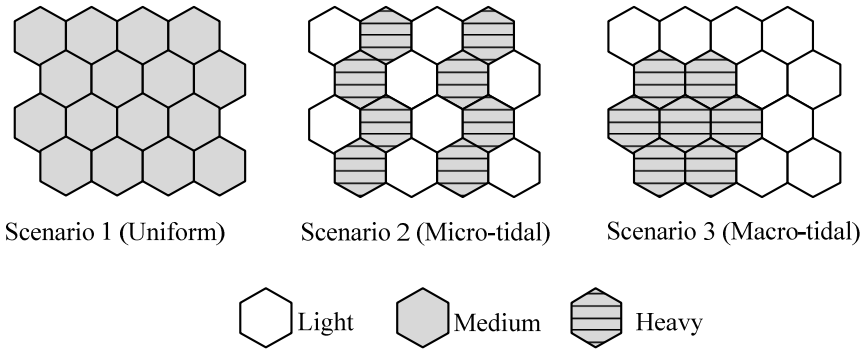


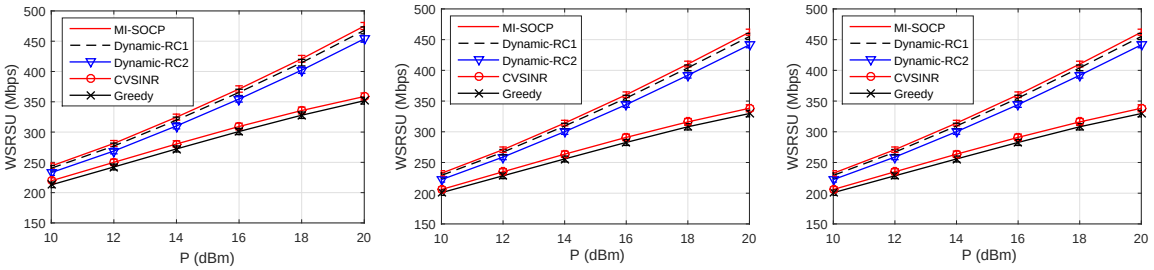

Figure 2.3: Different user distribution scenarios.

### 2.6.2 Weighted Sum-rate Performance

We now consider a system without the computing-resource constraint and evaluate the performance of the following joint clustering and beamforming schemes.

- *MI-SOCP*: The WSRSU of this scheme is obtained by using the solver MOSEK to solve problem (41) in each iteration of Algorithm 1.
- *Dynamic-RC1*: Our proposed dynamic radio cooperation scheme where the clustering and beamforming decisions are updated *simultaneously* in each scheduling slot.
- *Dynamic-RC2*: Our proposed dynamic radio cooperation scheme where the clustering decision is updated in each shadowing realization while the beamforming decision is updated in every small-scale fading realization.
- *CVSINR*: A downlink user-centric joint scheduling and clustering scheme in [47] where the clustered virtual SINR (CVSINR) algorithm is used to design the beamforming vectors.

- *Greedy*: A greedy clustering algorithm in [49] that solves an equivalent set-covering problem to select the set of non-overlapping base station clusters. This scheme uses zero-forcing beamforming and greedy user scheduling.




(a) Uniform (b) Micro-tidal (c) Macro-tidal

Figure 2.4: WSRSU of a C-RAN downlink system using different radio cooperation schemes, evaluated on three different user distribution scenarios.

We performed simulations in a network of 16 cells with three different user-distribution scenarios, as shown in Fig. 2.3, including: Scenario 1 with all medium cells; Scenario 2, light and heavy cells are intermixed together to simulate micro-tidal effect; Scenario 3, heavy cells are grouped together, and the heavy cell group is surrounded by light cells to simulate macro-tidal effect. Note that light, medium, and heavy cells are to describe user density in each cell. We randomly generate 100 large-scale fading (shadowing) realizations, each consisting of 20 small-scale fading (Rayleigh) instances. For each large-scale fading realization, we place 32 users in random locations in the network such that there are 1 user in each light cell, 2 users in each medium cell, and 3 users in each heavy cell.

Figure 2.4(a-c) plots the WSRSU performance of the considered radio cooperation schemes in Scenario 1, 2, and 3, respectively. It can be seen that *Dynamic-RC1* scheme performs very closely to *MI-SOCP* scheme. Note that in *MI-SOCP* the clustering solution obtained by the global optimization solver is optimal. *Dynamic-RC2* exposes a small loss compared to *Dynamic-RC1* due to less frequent updates of clustering decision, while still significantly outperforming *CVSINR* and *Greedy* schemes. This is because the heuristic clustering of the RRHs in the last two schemes is suboptimal, plus their beamforming algorithms only aim at minimizing the intra-cluster interference *but not* the inter-cluster interference. On the other hand, our proposed Dynamic-RC schemes take into account the

global network condition that is available at the VBS pool, which provides better clustering decision and beamforming design. Compared to the *optimal* scheme, our proposed Dynamic-RC strategy has a significant advantage in reducing the execution time. In fact, in our simulation for the considered system configuration ($U$=32, $R$=16), MOSEK solver takes more than 100 s to obtain the optimal solution of the MI-SOCP problem, while each iteration in *Dynamic-RC*-1, 2 takes less than a second.

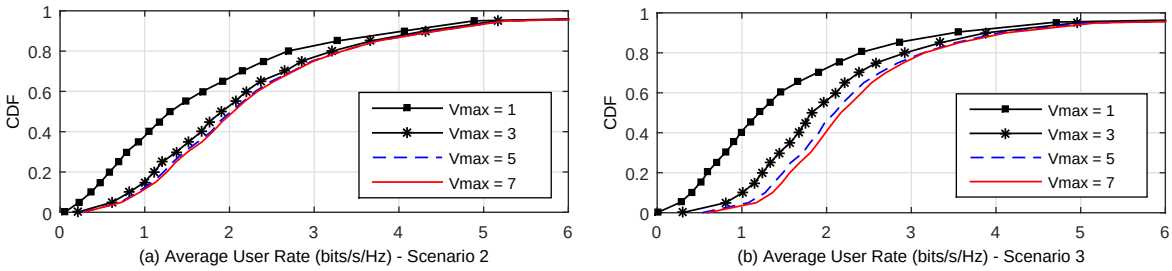
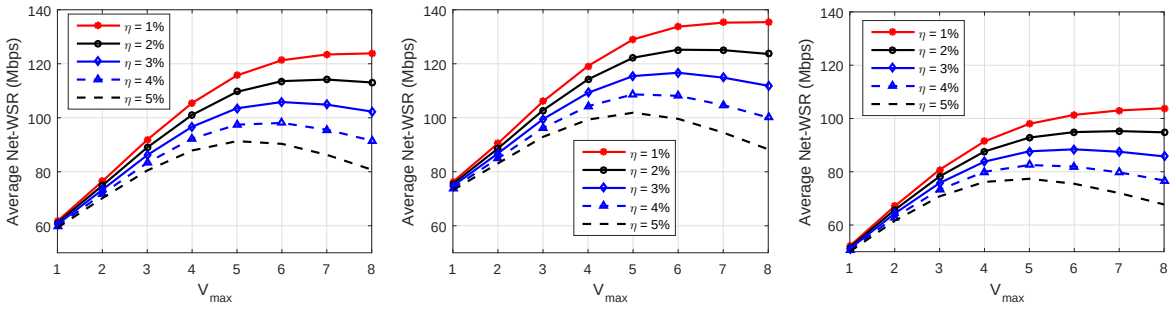



Figure 2.5: CDF of average user rate obtained by Dynamic-RC1.

### 2.6.3 Impact of Maximum Cluster Size

Figure 2.5(a-b) plots the CDF of average user rate (w.r.t. 32 users) achieved by Dynamic-RC1 scheme with different values of the maximum candidate cluster size, $V_{\max}$. When $V_{\max} = 1$, there is no cooperation among the RRHs. The results in Fig. 2.5(a-b) are obtained by performing 100 drops on *Scenario 2* and *Scenario 3*, respectively, with $P = 10$ dBm. The utility marginal functions are updated in each drop according to the proportional fairness criterion, i.e., $q_u = 1/\bar{R}_u$, where $\bar{R}_u$ is the long-term average data rate for user $u$. We observe that the improvement in average user rate due to larger cluster size in Scenario 3 (macro-tidal effect) is greater than that of Scenario 2 (micro-tidal effect). For instance, the Dynamic-RC1 scheme with $V_{\max} = 3, 5, 7$ provides 30%, 37%, and 38.6% gain, respectively, for the 60th-percentile average user rate over the non-cooperation scheme ($V_{\max} = 1$) in Scenario 2; while the corresponding gains in Scenario 3 are 45%, 59%, and 64%, respectively.

Figure 2.6(a-b) plot the average net-WSR of a downlink C-RAN system using Dynamic-RC1 scheme with different values of $V_{\max}$ and the percentage of allocated uplink resources training $\eta$ on the three described user-distribution scenarios. We observe the same trend in all three scenarios than when the cluster size is small, i.e., the average net-WSR increases




(a) Uniform (b) Micro-tidal (c) Macro-tidal

Figure 2.6: Average net-WSR of a downlink C-RAN system using Dynamic-RC1 strategy, evaluated on three different user distribution scenarios.

with cluster size. When the cluster size becomes higher, the average net-WSR eventually decreases since the training overhead used too many resources.

### 2.6.4 Benefits of Computing Resource Sharing

Figure 2.7 compares the WSRSU performance of our considered system with the *centralized* computing-resource constraint versus a conventional system with a *distributed* computing-resource constraint. In particular, we consider a network of 4 medium cells. For a fair comparison, we set $\arg\Gamma(C)$ to 400 Mbps and $\arg\Gamma(C_r)$ to 100 Mbps, and ran the *Dynamic-RC1* scheme on both systems. Note that, in this setting each of the 4 RRHs in the *distributed* system is provisioned to process maximum 100 Mbps of user baseband traffic at a time, while in the *centralized* system the VBS pool is provisioned to process maximum 400 Mbps baseband traffic at a time. We say that the computing resource is saturated in each system when the achieved Sum Rate (SR) of all the users reaches the maximum provisioned processing traffic rate. As the transmit power increases, observe in Fig. 2.7 that the computing capacity of the VBS pool in the *centralized* system saturates earlier than the distributed system does (when the computing capacity is saturated at all the RRHs). In fact, the WSRSU and SR of the distributed system saturate almost at the same time while the WSRSU of the centralized system continues to increase after the saturation point (of the SR), and is significantly higher (up to 150% gain) than that of the distributed system. This demonstrates the significant potential gains of C-RAN over the conventional distributed RAN in terms of WSRSU, computing resource, and transmit-power utilization.

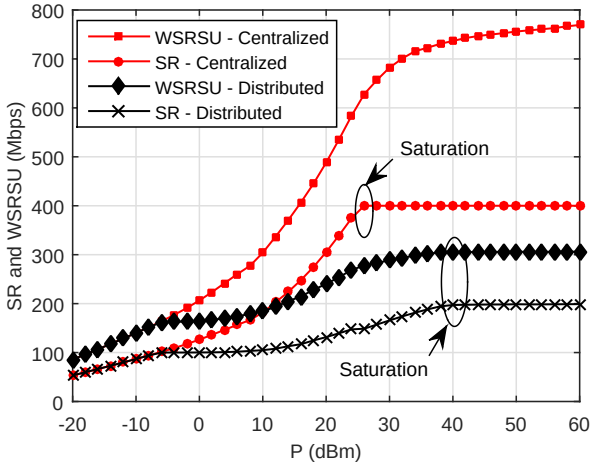



Figure 2.7: WSRSU of a downlink C-RAN system with centralized computing resource versus that of a traditional system with distributed computing resources.

## 2.7 Summary

In this chapter, we proposed a novel dynamic radio cooperation strategy for a Cloud Radio Access Network (C-RAN) that takes advantage of real-time communication and computing-resource sharing among Virtual Base Stations (VBSs). The underlying optimization problem was formulated as a mixed-integer non-linear program, which is NP-hard. Our approach decomposes the original problem into a cooperative beamforming design (CBD) problem with fixed clustering decision and a master problem that iterative adjusts the clustering solution. We effectively exploit conic programming and sequential convex approximation techniques to derive a novel iterative algorithm to solve the Dynamic-RC problem. Simulation results showed that our proposed low-complexity algorithm provides close-to-optimal performance in terms of weighted sum-rate system utility while significantly outperforming conventional radio clustering and beamforming schemes.

# Chapter 3

# Cooperative Hierarchical Caching and Request Scheduling in C-RANs

In this chapter, we present a novel caching framework aimed at fully exploiting the potential of C-RAN systems through cooperative hierarchical caching which minimizes the network costs of content delivery and improves users' Quality of Experience (QoE). In particular, we consider the cloud-cache in the cloud processing unit (CPU) as a new layer in the RAN cache hierarchy, bridging the capacity-performance gap between the traditional edge-based and core-based caching schemes. A delay-cost model is introduced to characterize and formulate the cache placement optimization problem, which is shown to be NP-complete. As such, a low complexity, heuristic cache management strategy is proposed, constituting of a proactive cache distribution algorithm and a reactive cache replacement algorithm. A Cache-Aware Request Scheduling (CARS) algorithm is devised in order to optimize online the tradeoff between content download rate and content access delay. Via extensive numerical simulations—carried out using both real-world YouTube video requests and synthetic content requests—it is demonstrated that the proposed cache-management strategy outperforms traditional caching strategies in terms of cache hit ratio, average content access delay, and backhaul traffic load. Additionally, the proposed cache-aware content request scheduling algorithm achieves superior tradeoff performance over traditional approaches that optimize either users' rate or access delay alone.

## 3.1 Introduction

Over the last few years, the demand on mobile networks has fundamentally shifted from being a steady increase in traffic for connection-centric communications such as phone calls and text messages to an explosion of content-centric communications such as video

streaming and content sharing. According to a recent forecast by Cisco [90], global mobile data traffic will reach 30.6 exabytes per month by 2020 with 75% of that being video traffic. The resulting demand poses immense pressure on the mobile network capacity and challenges the efficiency of the backhaul link that connects the access network to the core network. In order to prevent the backhaul capacity from becoming the system bottleneck (especially during peak-traffic hours), edge caching has been recognized as a promising solution to mitigate backhaul usage while reducing content-access latency [91,92]. In cellular edge caching, popular contents are cached at the BSs so that demands from users to the same content can be accommodated easily without duplicate transmissions from remote servers in the Content Delivery Network (CDN). This directly translates into sizable reduction in Capital Expenditure (CapEx) and Operational Expenditure (OpEx) [93].

One key limitation of caching in wireless networks is that the relatively small cache storage at individual BSs often lead to moderate cache hit performance. Recently, a number of solutions have been proposed to improve cache hit performance via collaborative caching, in which different cache entities share their contents, thus forming a larger cache-storage pool [94–97]. Existing collaborative-caching paradigms can be categorized as *horizontal collaboration* among the BSs' caches or *vertical collaboration* (hierarchical caching) between the BSs' caches and cache at the Core Network (CN). While offering great potential to improve cache hit ratio over non-collaborative schemes, there are several challenges that fundamentally limit the effectiveness of these collaborative-caching paradigms. First, in hierarchical caching, fetching contents from CN's cache to the BSs often undergoes considerable delay that is many-fold higher than the delay of transferring content among the BSs [94, 95, 97–99]. Second, current collaborative-caching techniques at the BSs rely on the direct interconnections between the BSs, which have very limited capacity and cannot handle large amount of content sharing. Since the BSs in the current 4G cellular network are connected with each other via the X2 interface, which is designed for exchanging control information or users' data buffer during handover [100,101], it is impractical to exploit such interface for inter-cache data transfer and to realize the benefits of collaborative caching.

**Our Vision:** In C-RAN, the computational functionalities of the BSs can be fully or partially implemented in a common Cloud Processing Unit (CPU) that can be hosted in

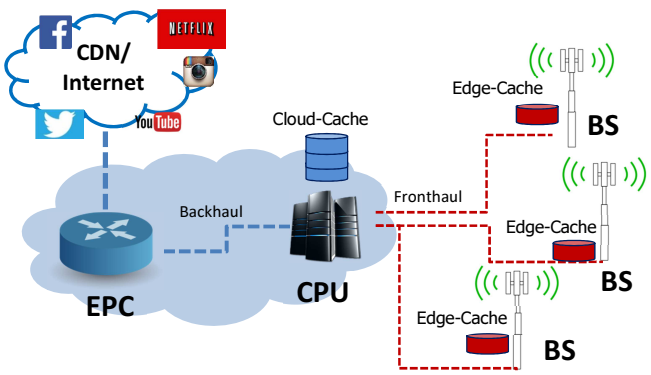



Figure 3.1: Illustration of a C-RAN caching system where the cloud-cache is deployed at the CPU and the edge-caches are deployed at the BSs.

a small data center. Fueled by the strong computing capabilities and storage resources at the CPU, C-RAN can provide a central port for traffic offload and content management in order to handle effectively the fast-growing multimedia traffic and meet the demands of mobile users. By leveraging the C-RAN architecture, we propose a *cooperative hierarchical caching* strategy making use of distributed *edge-caches* at the BSs and the *cloud-cache* at the CPU (see Fig. 3.1). The deployment of edge-caches and cloud-cache are complementary and inter-operable. Unlike existing solutions, the cloud-cache presents a new layer in the RAN cache hierarchy, compromising between the edge-based caching (small cache size, low access latency) and core-based caching schemes (large cache size, high access latency) to improve cache hit performance. In particular, we leverage the high-bandwidth, low-latency fronthaul links (e.g., optical fiber) [11, 102] connecting the BSs for cache content sharing; this way, each BS retrieves cache contents from the neighboring BSs via a "U-turn" (BS-CPU-BS), which is much more latency- and cost-effective than fetching content from the original remote server in the CDN via backhaul network [94, 95, 98]. The cloud-cache and edge-caches collaboratively form an "octopus-like" caching network and thus we name our proposed caching scheme Octopus. This scheme has the potential to exploit in full the extra degrees of cooperation brought by C-RAN to pool the storage resources and increase cache hit ratio as well as to reduce outbound requests to the higher-level network elements.

Traditionally, the request scheduling problem has been considered in the literature in the form of the user-BS association problem, in which the main objective is to achieve load balancing. Specifically, the key inputs to traditional user-association policies are the

wireless channel and interference characteristics as well as the number of users already associated with each BS. Most existing works consider caching and user association separately [103, 104], both in the solutions and objective functions. In contrast, our work brings the additional consideration of caching into user association, thus providing an additional degree of freedom in allowing the users the flexibility of associating with the BS depending on both channel condition and content availability.

**Challenges and Contributions:** To exploit the benefits of the envisioned cooperative hierarchical caching paradigm, there are several important challenges that need to be addressed. *First*, the presence of multiple cache layers coupled with the geographical variation of content popularity at different cell sites makes the cache-placement problem difficult. As the number of available content files is huge, the overhead and complexity of the cache-management algorithm need to be carefully considered. *Second*, when a user's request for content arrives, one needs to decide to which BS the user should be connected, depending on different performance objectives. From the users' perspective, some might want to download the content from the BS that can provide the highest rate so as to maximize the user rate utility [103, 104]; whereas, with the presence of caches, the network might want to direct the user to the BS that already stores the requesting content so as to mitigate backhaul usage. The latter case also benefits those users who prefer lower content access delay. Therefore, it is imperative to design a content request scheduling algorithm that optimizes the tradeoff between users' rate utility and content access delay. In summary, the novelty and contributions of this work are as follows.

- To the best of our knowledge, we are the first to introduce the cooperative hierarchical caching paradigm in C-RAN, which comprises the edge-caches distributively deployed at the BSs and the cloud-cache at the CPU. All the cache entities are managed by the Central Cache Manager (CCM) at the CPU.

- We formulate a cache-management optimization problem that addresses the questions of *what* content and *where* to place it among the cache nodes so as to minimize the average content access delay cost, subject to the cache-size constraint at each node. We show that this is an NP-hard problem and propose a low-complexity, heuristic

strategy that combines both proactive and reactive caching, and that yields at least $\frac{1}{2}$ of the optimal value.

- We propose an online Cache-Aware Request Scheduling (CARS) that—in conjunction with the cache management policy—optimizes the tradeoff between content download rate and content access delay. The algorithm achieves a formal competitive performance bound, and allows flexible control of content download rate and content access delay.

- We carry out extensive numerical simulations using both a real-world YouTube video request trace and synthetic content requests following Zipf-based popularity model. We demonstrate that our caching strategy outperforms traditional caching deployment architectures and cache-management algorithms in terms of cache hit ratio, average content access delay, and backhaul traffic load. Furthermore, we show that the proposed cache-aware content request scheduling achieves superior tradeoff performance over the existing approaches.

Note that the overall design for such caching system, which could involve the prediction of content popularity, request arrival rate, and user mobility pattern, is a complex problem and goes well beyond the scope of this work. Here, we assume that such information is available at the CCM to make cache-management and request scheduling decisions.

**Chapter Organization:** The remainder of this chapter is organized as follows. In Sect. 3.2, we review the related work. In Sect. 3.3, we present the system architecture and introduce the content access delay cost model; in Sect. 3.4, we describe our cache-management strategy to minimize the content access delay cost; in Sect. 3.5, we present our cache-aware request scheduling algorithm; in Sect. 3.6, we discuss performance evaluation via numerical simulations; finally, in Sect. 3.7, we conclude the chapter.

## 3.2 Related Work

There have been a number of works exploiting the benefits of content caching in cellular networks (cf. [92, 105–108]), which have different settings from traditional caching on the

Internet. In these works, the authors propose to alleviate backhaul congestion via proactive caching at the small cell BSs, whereby files are proactively cached during off-peak hours based on file popularity and on correlations between users and file patterns. In [91], the notion of femtocaching is introduced, in which the femtocell-like BSs are used to form a distributed caching network that assists the macro BS to handle requests of popular files. To overcome the cache-size limitation at individual BSs, collaborative caching has been exploited in small-cell networks [94–96, 109] and among the operators [110]. Along this line, the authors in [95, 111] propose online collaborative caching algorithms that aim at minimizing the total cost paid by the content providers without requiring prior knowledge about the content popularity. Recently, we propose in [112] a collaborative joint caching and processing strategy for multi-bitrate video streaming in Mobile-Edge Computing (MEC) networks. In this paradigm, each MEC server deployed next to a BS acts as both the cache server and the video transcoding server. In [113], Poularakis et al. propose to combine caching and multicast to improve energy efficiency in 5G cellular network with a massive demand for delay-tolerant content.

While offering great potential to bring popular content closer to the users, the aforementioned caching schemes only rely on the deployment of edge-caches. Hence, due to limited cache-size at the BSs (compared to the very large amount of popular content), these *edge-only* caching schemes suffer from high cache miss ratio. To overcome this issue, the authors in [94] consider caches both in the RAN edge and in the Evolved Packet Core (EPC). Along this line, the techniques in [105] are further extended to a hierarchical caching scheme in [96] where the gateways in the EPC also have video caches. While it is possible to implement relatively large cache size at the EPC so as to improve the cache hit ratio, fetching content from EPC to the BSs still undertakes considerable delay due to the involvement of multiple intermediate network components. More recently, caching in C-RAN has been studied in [114], which focuses on resource allocation and BS association with the presence of cluster content caching, and in [115], which focuses on dynamic BS clustering and multicast beamforming. Differently form these works which only consider static caching, our strategy dynamicaly updates and optimizes the cache placement at each node given cached contents at other nodes.

Traditionally, the problem of user-BS association was mostly addressed using offline approaches, see for example [103,116]. While providing optimal solutions, these approaches are not practical as they require *a-priori* knowledge of the users' arrivals and rates, and often result in re-scheduling large numbers of on-going connections. On the other hand, practical online algorithms for user-BS association are exploited recently in [104, 117] to maximize the sum-rate utility of users. Differently from these works, we consider the presence of cache contents at the BSs and designs a content request scheduling that optimizes the tradeoff between content download rate and content access delay. There are also several existing works considering the joint design of content caching and request scheduling policy in wireless networks. For example, [118] designs a video caching and user association scheme to minimize the user experienced delay, while [119] focuses on designing the joint user association and data caching strategy to minimize the requests served by the macro BSs. In addition, [120] studies the complexity of the joint user association and caching scheme. Most of these works, however, do not consider heterogeneity aspects such as the difference of wireless channel quality among different users, and their solutions do not allow control of the tradeoff between content download rate and content access delay.

## 3.3 Caching System Model

In this section, we describe the considered cooperative hierarchical caching system and introduce the content access delay cost model. The key parameters used in this chapter are listed in Table 5.1.

### 3.3.1 System Architecture

We consider a C-RAN that consists of multiple distributed BSs, all connected to a common CPU via low-latency, high-bandwidth fronthaul links, as illustrated in Fig. **??**. The CPU is connected to the EPC via a backhaul network that further connects the EPC to the Internet and the CDN. Most of the content providers make use of CDN to distribute their content geographically closer to the end users. While serving a large portion of the Internet content, the CDN servers are deployed outside of the mobile network domain.

Table 3.1: Summary of Key Parameters for C-RAN Caching System

| Symbol | Description |
|---|---|
| $\mathcal{R}$ | Set of $R$ BSs |
| $\mathcal{U}$ | Set of $U$ users |
| $\mathcal{U}_r$ | Set of users served by BS $r$ |
| $\mathcal{F}$ | Set of $F$ files |
| $p_{ir}$ | Probability of file $i$ being requested from cell $r$ |
| $M_r$ | Storage capacity [files] of edge-cache at BS $r$ |
| $M_0$ | Storage capacity [files] of cloud-cache |
| $\mathcal{C}_r$ | Set of files cached at BS $r$ |
| $\mathcal{C}_0$ | Set of files cached at cloud-cache |
| $\mathcal{V}$ | Cache placement ground set |
| $\mathcal{V}_r$ | Set of files that might be placed in cache $\mathcal{C}_r$ |
| $f_{ir}$ | Copy of file $i$ in cache $\mathcal{C}_r$ |
| $\mathcal{C}$ | Cache placement decision |
| $c_{ik}(\mathcal{C})$ | Indicator of whether (I.o.w) $f_{ik} \in \mathcal{C}_k$ |
| $x_{ir}^{k}$ | I.o.w a request for file $i$ from BS $r$ is retrieved from $\mathcal{C}_k$ |
| $x_{ir}^{R+1}$ | I.o.w a request for file $i$ from BS $r$ is retrieved from the CDN |
| $d_r$ | Delay cost of transferring a file between cloud-cache and BS $r$ |
| $d_0$ | Delay cost of transferring a file from the CDN to the CPU |
| $d_{rk}$ | Delay cost of transferring a file between BS $r$ and BS $k$ |
| $P_r$ | Transmit power of BS $r$ |
| $h_{ur}$ | Channel gain between BS $r$ and user $u$ |
| $y_{ur}$ | I.o.w user $u$ is scheduled to download content from BS $r$ |
| $\beta$ | Trade-off parameter |

The realization of fronthaul links in C-RAN allows the BSs to retrieve cache contents from each other via the "U-turn" (BS-CPU-BS). This way of getting content is more latency- and cost-effective than fetching content from the original remote server in the CDN via backhaul network [94, 95, 98]. In this work, we introduce a *cooperative hierarchical caching* paradigm that consists of the *cloud-cache* deployed at the CPU and the distributed *edge-caches* at the BSs. In the proposed system, we consider that there is a CCM implemented at the CPU to monitor all the requests generated from users within the local RAN, and is responsible for making cache (re)placement decisions as well as content request scheduling. In addition, fueled by the powerful processing capability at the CPU, one can implement sophisticated machine-learning and data-mining algorithms to estimate the content popularity in each cell. Such methods would involve analyzing data from popular websites,

newspapers, and social networks to determine, around a specific BS, what kinds of contents people like, search for, and what the consumer profiles of these people are [121, 122]. While the actual content files are physically stored in separated caches, a global indexing table can be maintained by the CCM to facilitate content lookup and cache management.

Let $\mathcal{R} = \{1, 2, ...R\}$ denote the set of $R$ BSs in the considered C-RAN, $\mathcal{U} = \{1, 2, ...U\}$ denote the set of active users in the system, and $\mathcal{F} = \{1, 2, ..., F\}$ denote the set of indexes of all content files available for download. For notational convenience, we assume an equal size [MB] for all files (as also considered in [91, 92]). This assumption could be easily lifted by considering a finer packetization, and by breaking longer files into blocks of the same length. In this work, we extend the analysis in [42], which only considers global content popularity. Rather, we assume to have content popularity at cell level, and define the popularity distribution of the files at each cell $r$ as $\mathcal{P} = \{p_{1r}, p_{2r}, ..., p_{Fr}\}$, where $p_{ir} \in [0, 1]$ is the probability of file $i$ being requested from a user in cell $r$ and $\sum_{i\in\mathcal{F}} p_{ir} = 1, \forall r \in \mathcal{R}$.

We consider that each BS integrates not only the front Radio Frequency (RF), but also certain capabilities to enable caching such as content storage and look up. We assume that each BS $r \in \mathcal{R}$ is equipped with an *edge-cache*, with normalized storage capacity of $M_r$ [files], and that the CPU is equipped with a *cloud-cache*, with normalized storage capacity of $M_0$ [files] (usually $M_0 \gg M_r, r = 1, ..., R$). The total cache capacity in the system is thus given by $M = \sum_{r=0}^{R} M_r$. We refer to $\mathcal{C}_r$, $r \in \mathcal{R}$, as the cache at BS $r$, which implies the set of files stored at BS $r$. Similarly, we refer to $\mathcal{C}_0$ as the cache or equivalently the set of files stored at the CPU. To describe the cache-placement decision, i.e., which files should be stored in which caches, we define the cache-placement ground set as,

$$\mathcal{V} = \{f_{10}, f_{20}, ..., f_{F0}, ..., f_{1R}, f_{2R}, ..., f_{FR}\}, \tag{1}$$

where $f_{ir}$ denotes the copy of file $i$ in cache $\mathcal{C}_r$. Note that the indexing of caches $\mathcal{C}_r$'s, $r = 0, 1, ..., R$, includes all the edge-caches and cloud-cache. In the subsequent analysis, unless otherwise stated, we will refer to file $i$ and to its copy $f_{ir}$ interchangeably. The ground set $\mathcal{V}$ can be partitioned into $R+1$ disjoint sets, $\mathcal{V}_0, \mathcal{V}_1, ..., \mathcal{V}_R$, where $\mathcal{V}_r = \{f_{1r}, f_{2r}, ..., f_{Fr}\}$ is the set of all files that might be placed in the cache $\mathcal{C}_r$. Hence, we can write $\mathcal{C}_r \subseteq \mathcal{V}_r$. A

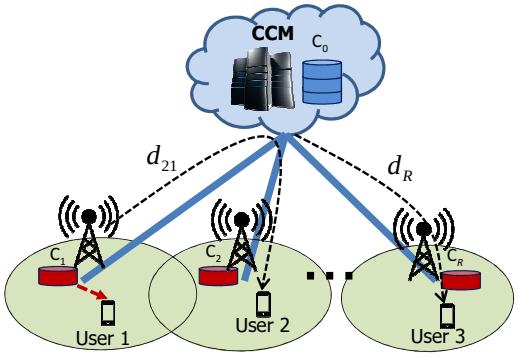



Figure 3.2: Illustration of Octopus caching system constituted of cloud-cache $\mathcal{C}_0$ and edge-caches $\mathcal{C}_1, ..., \mathcal{C}_R$, which can share cached contents via fronthaul links.

feasible cache-placement decision, denoted as $\mathcal{C} = \{\mathcal{C}_0, \mathcal{C}_1, ..., \mathcal{C}_R\}$, must satisfy the storage capacity constraints as follows,

$$|\mathcal{C}_r| \leq M_r, \forall r = 0, 1, ..., R. \tag{2}$$

When a user makes a request to BS $r$ for file $i$ that is already stored in the local edge-cache $\mathcal{C}_r$, it can directly download file $i$ from $\mathcal{C}_r$ without incurring traffic on the fronthaul and backhaul links. If the requested file $i$ is not stored in the local edge-cache, the request is forwarded to the CCM at the CPU. Upon receiving the request for file $i$ from BS $r$, the CCM will firstly search for file $i$ in the cloud-cache $\mathcal{C}_0$, and then in the neighboring caches of BS $r$, i.e., $\mathcal{C}_k$'s, $k \neq r$. If file $i$ is found in one of the caches, the CCM will direct the user to download the file directly from that cache via fronthaul links; otherwise the user will download the file from the origin server in the CDN, incurring traffic in the backhaul links. In Fig. 3.2 we illustrate the overview of our proposed Octopus caching system with an example where requests from user 1 (in cell 1) and user 2 (in cell 2) are retrieved from edge-cache $\mathcal{C}_1$, whereas request from user 3 in cell $R$ is retrieved from cloud-cache $\mathcal{C}_0$.

To facilitate subsequent analyses, let us define the following binary variables for a given cache-placement decision $\mathcal{C}$ as follows ($\forall i \in \mathcal{F}, r \in \mathcal{R}, k \in \{0\} \cup \mathcal{R}$),

$$c_{ik}(\mathcal{C}) = \begin{cases} 1 & f_{ik} \in \mathcal{C}_k, \\ 0 & \text{otherwise}, \end{cases} \tag{3}$$

$$x_{ir}^{k} = \begin{cases} 1 & \text{if request for file } i \text{ from BS } r \text{ is retrievedfrom } \mathcal{C}_k, \\ 0 & \text{otherwise;} \end{cases} \tag{4}$$

$$x_{ir}^{R+1} = \begin{cases} 1 & \text{if request for file } i \text{ from BS } r \text{ is retrievedfrom the CDN,} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Since each request should be downloaded from only one place, we impose the following constraint,

$$\sum_{k=0}^{R+1} x_{ir}^{k} = 1, \forall i \in \mathcal{F}, r \in \mathcal{R}. \tag{6}$$

### 3.3.2 Content Access Delay Cost

Although storage resources (e.g., hard disk) are cheap nowadays, it is neither cost-efficient nor feasible to store all available files in the caches. When a user requests for a file that is not available in the cache of the serving BS, it has to retrieve the file from other places, thus incurring additional access delay. The content access delay reflects the time it takes for a user to send the request for content until it first receives the data, which includes the time for the data to traverse the user-BS wireless link and backhaul network to the CDN. In this work, we will focus on the cost model for content access delay as considered in [91, 94].

Let $d_r$ denote the delay cost incurred when transferring a file from the cloud-cache to BS $r$ via fronthaul link, which we assume to be the same as the delay cost of retrieving that file from edge-cache $\mathcal{C}_r$ to the CPU. Let $d_0$ denote the delay cost incurred when transferring a file from the CDN to the CPU. Furthermore, we assume the cost of transferring a file from cache of BS $k$ to BS $r$ is $d_{rk} = d_r + d_k$. In practice, $d_0$ is usually much greater than $d_r$ and $d_{rk}$ as the backhaul link connecting the CPU to the original content server is many-fold further than the fronthaul links between the BSs and the CPU. This makes it cost-effective to retrieve content from the in-network caches whenever possible rather than downloading them from the remote server [94, 97, 98]. We consider that the incurred delay cost of a user downloading a file directly from its serving BS's cache is zero [98]. This is because such delay is negligible and will incur no matter whether caching is used or not.

Let us denote $\mathcal{U}_r \subseteq \mathcal{U}$ as the set of users served by BS $r$. Thus, for a given cache-placement decision $\mathcal{C}$ and popularity distribution $\mathcal{P}$, we can calculate the average delay cost of user $u \in \mathcal{U}_r$ as in (7) below.

$$\bar{D}_{u,r} = \sum_{i \in \mathcal{F}} p_{ir} \left( x_{ir}^0 d_r + \sum_{k \in \mathcal{R} \setminus \{r\}} x_{ir}^k d_{rk} + x_{ir}^{R+1}(d_r + d_0) \right). \tag{7}$$

This delay cost reflects the expected content access delay that the users have to wait before having access to the requested content. Reducing the average access delay cost also directly translates to a decrease in backhaul network usage, i.e., the amount of data traffic going through the backhaul links, and thus to a reduced network resource consumption. Another key aspect that affects the users' QoE is the content download rate. For example, both the initial access delay and content download rate directly affect the initial buffer time and the number of stalls during the video streaming session.

### 3.3.3 Decomposition Approach

The overall goal of our design is to improve Quality of Experience (QoE) for content-downloading users, which is mainly characterized by the content access latency and content download rate. To this end, our solutions for content caching and request scheduling strive to minimize the average content access delay and maximize the average content download rate, which are not always achieved simultaneously, and hence, an optimized tradeoff between the two objectives is targeted.

Interestingly, content placement and request scheduling generally occur at different time scales: content popularity often varies slowly, at the scale of hours or days based on the measurement and prediction from various sources; on the other hand, request-scheduling decisions have to adapt to the dynamics in user location and channel conditions, which vary in the order of seconds. This makes a joint optimization to react promptly to the dynamic network changes difficult; hence, it motivates the decomposition of the overall problem into (i) the cache management subproblem, which takes the long-term content popularity as input, and (ii) the content request scheduling subproblem, which takes into account the incoming specific requests, the corresponding channel condition of the users, and the

cache availability at the BSs and the CPU. Although the two sub-problems are addressed separately due to their different time-scales, the coupling between the two is reflected by the fact that the solution of the cache-placement policy is used as input to the request scheduling policy. Likewise, the request-scheduling solution will affect the cache-placement decision the next time it is recalculated. This is because the content popularity is calculated based on the number of requests to each content at different BSs as the result of the request scheduling policy; hence, after a long-time-scale period (hours or days), the cache-placement decision will be made based on the *updated* content popularity.

In the following, the cache management subproblem first and then the cache-aware request scheduling subproblem will be presented in Sects. 3.4 and 3.5, respectively.

## 3.4 Cache Management Strategy

We present here the formulation of the cache management problem, followed by the background material and intuition of our approach. We then describe our proposed greedy solution, with guaranteed performance, consisting of a cache-distribution algorithm and of a replacement algorithm.

### 3.4.1 Problem Formulation

As described in the previous section, the reduction of content access delay cost directly translates into the improvement in users' QoE and the reduction in network operational cost. It is therefore imperative to design an efficient cache-management strategy so as to minimize the expected content access delay. In particular, we consider a *dynamic* cache-management strategy that proactively distributes content files in the caches and reactively updates the cached files. Notice that multiple copies of the same file can be stored at different caches.

The underlying optimization problem to realize the proposed strategy can be formulated as follows,

$$\min_{\mathcal{C}, x_{ir}^k} \sum_{r\in\mathcal{R}} \sum_{u\in\mathcal{U}_r} \bar{D}_{u,r}, \tag{8a}$$

$$\text{s.t.} \sum\nolimits_{i\in\mathcal{F}} c_{ir}(\mathcal{C}) \leq M_r, \ \forall r = 0, 1, ..., R, \tag{8b}$$

$$\sum\nolimits_{k=0}^{R+1} x_{ir}^k = 1, \ \forall r = 1, ..., R, \tag{8c}$$

$$x_{ir}^k \leq c_{ik}(\mathcal{C}), \ \forall r \in \mathcal{R}, k \in \{0\} \cup \mathcal{R}, i \in \mathcal{F}, \tag{8d}$$

$$x_{ir}^k \in \{0,1\}, x_{ir}^{R+1} \in \{0,1\}, c_{ik}(\mathcal{C}) \in \{0,1\}, \forall r \in \mathcal{R}, k \in \{0\} \cup \mathcal{R}, i \in \mathcal{F}, \tag{8e}$$

with $\bar{D}_{u,r}$ given by (7). The objective function (8a) represents the total average delay cost incurred by satisfying content requests from all users in the network. The constraint (8b) imposes the cache storage capacities and the constraint (8d) represents the cache availability constraint, i.e., ensuring that a content file can be retrieved from a cache only if it has been stored in that cache. Note that, mathematically, the average delay cost of user $u$ served by BS $r$ in a *non-hierarchical* caching system (cf. [95, 111]) may be viewed as a special case of the average delay cost $\bar{D}_{u,r}$ presented in (7) when the cloud cache $\mathcal{C}_0$ and the CDN are considered as edge-cache nodes. However, the whole optimization problem considered in (8) is fundamentally different from that of the non-hierarchical case, given the different constraints and settings. For instance, there is no constraint on the cache capacity at the CDN and the content popularity at the cloud-cache is unknown; hence, the CDN and the cloud-cache are fundamentally different from the edge-caches.

From constraint (8c), by substituting $x_{ir}^{R+1}$ by $1 - \sum_{k=0}^{R} x_{ir}^k$ into (7), we get,

$$\bar{D}_{u,r} = \sum\nolimits_{i\in\mathcal{F}} p_{ir} \left(d_o + d_r - S_{ir}\right), \tag{9}$$

where

$$S_{ir} = x_{ir}^0 d_0 + x_{ir}^r \left(d_0 + d_r\right) + \sum_{k\in\mathcal{R}\setminus\{r\}} x_{ir}^k \left(d_0 - d_k\right). \tag{10}$$

Observe that $S_{ir}$ can be seen as the delay cost saving when file $i$ is requested by a user $u$ at BS $r$, and that $S_{ir}$ is the only term in (9) that depends on the optimization variables. Hence,

problem (8) can be recast as a problem of maximizing the average delay cost reduction, expressed as,

$$\max_{\mathcal{C},x_{ir}^k} \sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{U}_r}\sum_{i\in\mathcal{F}} p_{ir}S_{ir}, \tag{11a}$$

$$\text{s.t. } \sum\nolimits_{i\in\mathcal{F}} c_{ir}(\mathcal{C}) \leq M_r,\ \forall r = 0,1,...,R, \tag{11b}$$

$$\sum\nolimits_{k=0}^{R} x_{ir}^k \leq 1,\ \forall r = 1,...,R,\ i\in\mathcal{F}, \tag{11c}$$

$$x_{ir}^k \leq c_{ik}(\mathcal{C}),\ \forall r\in\mathcal{R}, k\in\{0\}\cup\mathcal{R}, i\in\mathcal{F}, \tag{11d}$$

$$x_{ir}^k \in \{0,1\}, x_{ir}^{R+1}\in\{0,1\}, c_{ik}(\mathcal{C})\in\{0,1\},\ \forall r\in\mathcal{R}, k\in\{0\}\cup\mathcal{R}, i\in\mathcal{F}. \tag{11e}$$

The objective function (11) can be seen as the sum of the *utility value* seen by each BS and our goal here is to maximize the sum utility value seen by all BSs. The intractability of this problem can be shown in Theorem 1 below.

**Theorem 1.** *The cache-placement optimization problem in* (11) *is NP-complete.*

*Proof.* See Appendix in [42] or Theorem 1 in [110]. □

Due to the NP-completeness of the problem, a global optimal solution usually comes with exponential computational complexity, which is impractical to implement. Therefore, our approach aims for a low-complexity, suboptimal solution that can be implemented in a practical system. In particular, we will show that problem (11) belongs to the classical class of problems of maximizing a *monotone submodular function* over a *matroid* constraint [123, 124]. We then propose a greedy cache-management solution for problem (11) consisting of a cache-distribution algorithm and a backtracking cache-replacement algorithm.

Before going into details of our proposed solution, we introduce some essential background material and intuition of our approach.

### 3.4.2 Preliminaries

In the following, we provide the basic definitions of matroids and submodular functions [125], which will be used in the analysis of our proposed cache-management strategy.

**Matroids.** A *matroid* is a pair $(\mathcal{V}, \mathcal{I})$ such that $\mathcal{V}$ is a finite set and $\mathcal{I} \subseteq 2^{\mathcal{V}}$ is a collection of subsets of $\mathcal{V}$ satisfying the following two properties:

- $\mathcal{I}$ is downward closed, i.e., if $A \subseteq B \subseteq \mathcal{V}$ and $B \in \mathcal{I}$, then $A \in \mathcal{I}$;
- If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists $e \in A \backslash B$ such that $B \cup \{e\} \in \mathcal{I}$.

Matroids generalize the concept of linear independence found in linear algebra to general sets, and sets in $\mathcal{I}$ described above are called *independent*. One of the important applications of matroids is the concept of matroid constraint defined via the *partition matroid*. Consider a finite ground set $\mathcal{V}$ that is partitioned into $n$ disjoint sets $\mathcal{V}_1, \mathcal{V}_2, ...\mathcal{V}_n$ with associated integers $m_1, m_2, ..., m_n$, a partition matroid $\mathcal{I}$ is given as,

$$\mathcal{I} = \{A \subseteq \mathcal{V} : |A \cap \mathcal{V}_i| \leq m_i, \forall i = 1, ..., n\}. \tag{12}$$

**Submodular functions.** Consider a finite ground set $\mathcal{V}$, a set function $g : 2^{\mathcal{V}} \to \mathbb{R}$ is submodular if, for all sets $A, B \subseteq \mathcal{V}$,

$$g(A) + g(B) \geq g(A \cup B) + g(A \cap B). \tag{13}$$

Given a submodular function $g : 2^{\mathcal{V}} \to \mathbb{R}$ and $A, S \subset \mathcal{V}$, the function $g_A$ defined by $g_A(S) = g(A \cup S) - g(A)$ is also submodular, and if $g$ is monotone then $g_A$ is also monotone. For $i \in \mathcal{V}$, we abbreviate $A \cup \{i\}$ by $A + i$. Let $g_A(i) = g(A + i) - g(A)$ denote the marginal value of an element $i \in \mathcal{V}$ with respect to the subset $A \subseteq \mathcal{V}$. Then, $g$ is submodular if, for all $A \subseteq B \subseteq \mathcal{V}$ and for all $i \in \mathcal{V} \backslash B$, we have,

$$g_A(i) \geq g_B(i). \tag{14}$$

Intuitively, submodular functions capture the concept of *diminishing returns*: as the set becomes larger the benefit of adding a new element to the set will decrease. The function $g$ is monotone if, for $A \subseteq B \subseteq \mathcal{S}$, we have $g(A) \leq g(B)$.

### 3.4.3 Online Cache Management Algorithms

We exploit the special structure of problem (11) to formulate it as the problem of maximizing a submodular function subject to matroid constraints. In particular, motivated by the approach in [91], we will show that the constraints in (11) can be expressed as the independent sets of a matroid and the objective function can be expressed as a monotone submodular function.

**Cache Placement via Monotone Submodular Maximization:** Firstly, recall that every cache-placement decision $\mathcal{C}$ is a subset of the ground set $\mathcal{V}$ defined in (1), and that $\mathcal{C}_r = \mathcal{C} \cap \mathcal{V}_r$. With this position, the cache-capacity constraints in (11b) are equivalent to the condition $\mathcal{C} \subseteq \mathcal{I}$, where,

$$\mathcal{I} = \{\mathcal{C} \subseteq \mathcal{V} : |\mathcal{C} \cap \mathcal{V}_r| \leq M_r, \forall r = 0, 1, ..., R\}. \tag{15}$$

From (12) and (15), we see that our constraints form a partition matroid $\mathcal{M} = (\mathcal{V}, \mathcal{I})$. Also, notice from (3) that the set $\{c_{ir}(\mathcal{C}) : i \in \mathcal{F}\}$ can be considered as the Boolean representation of $\mathcal{C}_r$. We now have the following Lemma.

**Lemma 4.** *The objective function in* (11a) *is a monotone submodular function.*

*Proof.* For each file $i \in \mathcal{F}$ and BS $r \in \mathcal{R}$, we introduce the new variables $t_{ir}^k$'s as: $t_{ir}^0 = d_0$, $t_{ir}^r = d_0 + d_r$, $t_{ir}^k = d_0 - d_k$, $\forall k \in \mathcal{R} \backslash r$. Denote the objection function in (11a) as $g(\mathcal{C})$, it can now be expressed as,

$$g(\mathcal{C}) = \sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{U}_r} \sum_{i \in \mathcal{F}} p_{ir} S_{ir} = \sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{U}_r} \sum_{i \in \mathcal{F}} p_{ir} \sum_{k=0}^{R} x_{ir}^k t_{ir}^k. \tag{16}$$

Since sum of monotone submodular functions is monotone submodular, it is enough to prove that for each BS $r \in \mathcal{R}$, the set function $g_{ir}(\mathcal{C}) = S_{ir}$ is monotone submodular. Firstly, notice that from (16), we have,

$$g_{ir}(\mathcal{C}) = \max_{x_{ir}^k} \sum\nolimits_{k=0}^{R} x_{ir}^k t_{ir}^k \quad \text{s.t. (11c), (11d), (11e),} \tag{17}$$

which can be simplified to,

$$g_{ir}\left(\mathcal{C}\right)=\max\left\{0,t_{ir}^{0}c_{i0}\left(\mathcal{C}\right),t_{ir}^{1}c_{i1}\left(\mathcal{C}\right),...,t_{ir}^{R}c_{iR}\left(\mathcal{C}\right)\right\}. \tag{18}$$

For a new file $f_{in}\in\mathcal{V}\backslash\mathcal{C}$, let $\mathcal{C}_{in}=\mathcal{C}+f_{in}$. It is straightforward to verify that $g_{ir}\left(\mathcal{C}_{in}\right)\geq g_{ir}\left(\mathcal{C}\right)$ and, therefore, $g_{ir}\left(\mathcal{C}\right)$ is a *monotonic* function $\forall\mathcal{C}\subseteq\mathcal{V}$. Intuitively, adding a new file to a cache-placement set cannot decrease the value of the set function. Let us now consider another cache-placement set (decision) $\mathcal{K}$ such that $\mathcal{K}\subseteq\mathcal{C}$. Denote $\mathcal{K}_{in}=\mathcal{K}+f_{in}$, we have $g_{ir}\left(\mathcal{K}\right)=t_{ir}^{(\mathcal{K})}$ and $g_{ir}\left(\mathcal{K}_{in}\right)=t_{ir}^{(\mathcal{K}_{in})}$. Since $g_{ir}\left(.\right)$ is monotone, we have,

$$g_{ir}\left(\mathcal{C}\right)\geq g_{ir}\left(\mathcal{K}\right). \tag{19}$$

The marginal value of adding the file $f_{in}$ to the sets $\mathcal{C}$ and $\mathcal{K}$ can be expressed, respectively, as,

$$g_{ir,\mathcal{C}}\left(f_{in}\right)=g_{ir}\left(\mathcal{C}_{in}\right)-g_{ir}\left(\mathcal{C}\right), \tag{20}$$

$$g_{ir,\mathcal{K}}\left(f_{in}\right)=g_{ir}\left(\mathcal{K}_{in}\right)-g_{ir}\left(\mathcal{K}\right). \tag{21}$$

To prove that $g_{ir}\left(.\right)$ is submodular, we need to show that $g_{ir,\mathcal{K}}\left(f_{in}\right)\geq g_{ir,\mathcal{C}}\left(f_{in}\right)$ or, equivalently, that $\Delta_{r}^{in}=g_{ir,\mathcal{K}}\left(f_{in}\right)-g_{ir,\mathcal{C}}\left(f_{in}\right)\geq 0$. Using (18), we distinguish three cases,

(i) $t_{ir}^{n}>g_{ir}\left(\mathcal{C}\right)$: we have $g_{ir}\left(\mathcal{C}_{in}\right)=g_{ir}\left(\mathcal{K}_{in}\right)=t_{ir}^{n}$; hence, $\Delta_{r}^{in}=g_{ir}\left(\mathcal{C}\right)-g_{ir}\left(\mathcal{K}\right)\geq 0$, which stems from the inequality in (19).

(ii) $g_{ir}\left(\mathcal{K}\right)\leq t_{ir}^{n}\leq g_{ir}\left(\mathcal{C}\right)$: we have $g_{ir}\left(\mathcal{C}_{in}\right)=g_{ir}\left(\mathcal{C}\right)$ and $g_{ir}\left(\mathcal{K}_{in}\right)=t_{ir}^{n}$; hence, $\Delta_{r}^{in}=t_{ir}^{n}-g_{ir}\left(\mathcal{K}\right)\geq 0$.

(iii) $t_{ir}^{n}<g_{ir}\left(\mathcal{K}\right)$: in this case, adding $f_{in}$ does not provide any added value; we have $g_{ir}\left(\mathcal{C}_{in}\right)=g_{ir}\left(\mathcal{C}\right)$ and $g_{ir}\left(\mathcal{K}_{in}\right)=g_{ir}\left(\mathcal{K}\right)$, hence $\Delta_{r}^{in}=0$.

In summary, we always have $\Delta_{r}^{in}\geq 0$, which implies that $g_{r}\left(.\right)$ is submodular function in $\mathcal{V}$. The proof is complete. □

A popular approach to the problem of maximizing a monotone submodular function subject to a matroid constraint is to use a greedy algorithm [123, 124]. Based on the result from Lemma 4, we extent such algorithm to solve our problem in (11). Our proposed cache-management solution consists of two phases: first, the content files are proactively distributed to the caches using proactive caching; then, every time there is a cache miss and a new file is downloaded from the content server, the CCM will decide whether to replace this file with existing ones in the caches via reactive caching. The proposed Utility-based Hierarchical Proactive Cache Distribution (UHPCD) and Utility-based Hierarchical Reactive Cache Replacement (UHRCR) algorithms, which constitute the Octopus cache management strategy, are detailed in the following.

**Proactive Cache Distribution:** The UHPCD algorithm incrementally builds a placement solution starting with an empty cache placement set. In each iteration, it adds a new file with the highest marginal value to the cache-placement set, until all the caches are full. Since the objective function is submodular, the marginal value of a new file decreases as the cache-placement set grows. We outline the procedure of the greedy UHPCD algorithm as in Algorithm 4. Thanks to the monotone submodular property of problem 11, the solution obtained by Algorithm 4 is guaranteed to achieve a ratio of at least $\frac{1}{2}$ of the optimal value [124].

**Algorithm 4** Utility-based Hierarchical Proactive Cache Distribution (UHPCD)

1: Initialize: $\mathcal{V}_r = \{f_{1r}, f_{2r}, ..., f_{Fr}\}$, $\mathcal{C}_r = \emptyset$, $r = 0, 1, ..., R$,
$\mathcal{V} = (\mathcal{V}_0, \mathcal{V}_1, ..., \mathcal{V}_R)$, $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, ..., \mathcal{C}_R)$.
2: **repeat**
3: $f_{j'r'} = \underset{f_{jr} \in \mathcal{V} \backslash \mathcal{C}}{\arg\max} \left[ g\left(\mathcal{C} + f_{jr}\right) - g\left(\mathcal{C}\right) \right]$
4: $\mathcal{C} \leftarrow \mathcal{C} + f_{j'r'}$
5: **if** $|\mathcal{C}_{r'}| = M_{r'}$ **then** $\mathcal{V} \leftarrow \mathcal{V} \backslash \mathcal{V}_{r'}$
6: **until** $\mathcal{V} = \emptyset$
7: Output: $\mathcal{C}$

Algorithm 4 initializes all the caches as empty sets in step 1 and begins the iteration process in step 2. In each iteration, step 3 identifies the placement of file $j'$ in cache $\mathcal{C}_{r'}$, denoted by $f_{j'r'}$, that provides the highest marginal value when adding to the current cache placement set $\mathcal{C}$. Hence, $f_{j'r'}$ can be seen as the next best cache placement among the unplaced files $\{f_{jr} \in \mathcal{V} \backslash \mathcal{C}\}$. Step 4 adds $f_{j'r'}$ to the current cache placement set. In step 5,

if a cache is full, it will be excluded from the evaluation in the next iteration. The iteration process terminates in step 7 when the candidate set is empty.

**Remark 3**: (Complexity of Algorithm 4) In Algorithm 4, there are $\sum_{r=0}^{R} M_r$ iterations until all the caches are full, each of which involves calculating the marginal value of at most $(R+1)F$ elements that have not been included in the cache set. Evaluating each marginal value takes $\mathcal{O}(U)$ time; hence, the total running time is $\mathcal{O}\left((R+1)\,FU\sum_{r=0}^{R} M_r\right)$. When $M_r$ is a constant fraction of $F$, the time complexity is given by $\mathcal{O}\left((R+1)^2F^2U\right)$. □

**Reactive Cache Replacement:** Algorithm 4 described above initializes the cache distribution, which can be done during off-peak traffic hours (e.g., night-time) to utilize the unused backhaul bandwidth. Over the course of the day, following each cache miss, a new file will be downloaded from the remote content server to the BSs and delivered to the requesting user. As all the caches are full already, the CCM will decide to replace this new file with existing files in the caches only if such replacement could improve the value of the objective function. This approach ensures that the up-to-date cache-placement set always yields the highest marginal value. The UHRCR algorithm is shown in Algorithm 5.

**Algorithm 5** Utility-based Hierarchical Reactive Cache Replacement (UHRCR)

1: For a new file request $i$: $f_{ir} \notin \mathcal{C}, \forall r = 0, 1, ..., R$
2: **for** $t = 0 : R$ **do**
3: $\quad f_{j'r'} = \underset{f_{jr} \in \mathcal{C}}{\arg\min} \left[g\left(\mathcal{C}\right) - g\left(\mathcal{C} - f_{jr}\right)\right]$
4: $\quad$ **if** $g\left(\mathcal{C} - f_{j'r'} + f_{ir'}\right) > g\left(\mathcal{C}\right)$ **then**
5: $\quad\quad \mathcal{C} \leftarrow \mathcal{C} - f_{j'r'} + f_{ir'}$
6: $\quad$ **else** Break
7: Output: $\mathcal{C}$

Step 1 in Algorithm 5 indicates that the process will be initiated any time there is a cache miss, e.g., a request for file $f_{ir}$ that is not in the caches. Step 2 initiates the search for $R+1$ iterations. This is because there are potentially at most $R+1$ files, one in each cache, that can be replaced by the new file while ensuring the cache capacity constraint. In each iteration, step 3 finds a file with the least utility value. If swapping the least-valued file with the new file can increase the overall objective value, as verified in step 4, the swapping operation will be performed in step 5. *This ensures that the cache replacement process will not decrease the objective value initially achieved by the UHPCD process.*

**Remark 4**: (Complexity of Algorithm 5) The running time of step 3 in each iteration is $\mathcal{O}\left(U\sum_{r=0}^{R}M_r\right)$. Since there are at most $R+1$ iterations, the overall time complexity of Algorithm 5 is $\mathcal{O}\left((R+1)\,U\sum_{r=0}^{R}M_r\right)$, which simplifies to $\mathcal{O}\left((R+1)^2FU\right)$ when $M_r$ is a constant fraction of $F$. □

## 3.5 Online Cache-aware Request Scheduling

In this section, we design the content request scheduling policy that, in conjunction with the cache-management policy designed in the previous section, optimizes the tradeoff between content download rate and content access delay.

Without loss of generality, we consider that each user only requests for one content file at a time. Hence, a user becomes *active* when it is downloading a content file and remains *inactive* otherwise. Each new content request is coupled with a new user arrival to the network. Upon each new content request, one needs to decide to which BS should the user be associated, depending on different performance desire. On one hand, to maximize the sum-rate utility, each user should connect to a BS that could provide high download rate without degrading too much the rates of other users; on the other hand, to minimize the content access delay, the requesting user should download data from the BS whose cache contains the requested content.

In the following, we formulate the request scheduling problem in Subsection 3.5.1 and present the online solution with guaranteed performance bound in Subsection 3.5.2.

### 3.5.1 Problem Formulation

We assume that there are $N$ content requests arriving online to the network, one after the other. Each request is associated with the arrival of a new user and the user departs after it finishes downloading the content. The users are indexed by their order of arrival, e.g., user 1 arrives first and user $N$ arrives last. We denote $t_n$ as the time of the $n$-th departure for which a user (possibly different from user $n$) departs the network. Hence, there is no departure except at $t_n$ during each time window $[t_n, t_{n+1})$. Suppose there is a new request arrival at time $t \in [t_n, t_{n+1})$ and let $\mathcal{U}(t)$ be the current set of active users. Without loss of

generality, our analysis in the following will focus on the scheduling decisions in one time window, in which the system at each arrival time $t$ is investigated. For simplicity, the time index $t$ is dropped in subsequent analysis.

We consider that both the BSs and the users are equipped with a single antenna each. Each BS $r$ transmits at a fixed power $P_r, r \in \mathcal{R}$, over a single-carrier system with bandwidth $W$. Note that the analysis can be easily extended to a multi-carrier system where the spectrum is divided into resource blocks. Denote $h_{ru} \in \mathbb{C}$ as the channel gain between BS $r$ and user $u \in \mathcal{U}$, which captures the effect of path-loss, shadowing, and antenna gain. Note that the user-BS association usually takes place in a large time-scale (duration of the content download session) that is much larger than the time-scale of small-scale fading. Hence, similar to [103,104], we consider that the effect of fast-fading is averaged out during the association. In this case, the Signal-to-Interference-plus-Noise Ratio (SINR) of user $u$ when associated with BS $r$ is given by,

$$\mathrm{SINR}_{ur} = \frac{|h_{ru}|^2 P_r}{\sum_{s \neq r} |h_{su}|^2 P_s + \sigma^2}, \forall u \in \mathcal{U}, r \in \mathcal{R}, \tag{22}$$

where $\sigma^2$ is the background noise power.

Let the scheduling variables be $y_{ur} \in \{0,1\}, \forall u \in \mathcal{U}, r \in \mathcal{R}$, where $y_{ur} = 1$ if user $u$ is scheduled to download content from BS $r$ and $y_{ur} = 0$ otherwise. We denote the request scheduling policy as $\Psi = \{y_{ur} \,|u \in \mathcal{U}, r \in \mathcal{R}\}$. Here, we employ equal time-sharing allocation when a BS is associated with multiple users, which has been shown to be optimal in terms of maximizing the log-utility of user data rates [103]. Consequently, the data rate [bits/s] of user $u$ when downloading content from BS $r$ is calculated as,

$$\Gamma_{ur} = \frac{\phi_{ur}}{\sum_{v \in \mathcal{U}} y_{vr}}, \tag{23}$$

where $\phi_{ur} = W \log_2 (1 + \mathrm{SINR}_{ur}), \forall u \in \mathcal{U}, r \in \mathcal{R}$.

To provide proportional fairness among users in a wireless network, the logarithmic user-rate utility is popularly used [103] and is defined as $\sum_{r \in \mathcal{R}} y_{ur} \log (\Gamma_{ur})$ for each user $u \in \mathcal{U}$. Thus, the network rate utility can be seen as the sum log-rate utility of all users,

expressed as,

$$\Omega\left(\Psi\right)=\sum_{u\in\mathcal{U}}\sum_{r\in\mathcal{R}}y_{ur}\log\left(\frac{\phi_{ur}}{\sum_{v\in\mathcal{U}}y_{vr}}\right). \tag{24}$$

The cache-aware request scheduling optimization problem, which determines $\{y_{ur}\}$ so as to maximize a weighted objective of users' sum log-rate utility and content access delay cost saving, is formulated as follows,

$$\max_{\Psi}\ \sum_{u\in\mathcal{U}}\sum_{r\in\mathcal{R}}y_{ur}\log\left(\frac{\phi_{ur}}{\sum_{v\in\mathcal{U}}y_{vr}}\right)+\beta\sum_{u\in\mathcal{U}}\sum_{r\in\mathcal{R}}y_{ur}\sum_{i\in\mathcal{F}}p_{ir}S_{ir}, \tag{25a}$$

$$\text{s.t.}\ \sum\nolimits_{r\in\mathcal{R}}y_{ur}=1,\forall u\in\mathcal{U}, \tag{25b}$$

$$y_{ur}\in\left\{0,1\right\},\forall u\in\mathcal{U},r\in\mathcal{R}. \tag{25c}$$

In (25a), $\beta$ is a parameter that can be varied to control the tradeoff between users' sum log-rate utility and content access delay. In this case, each point on the tradeoff curve corresponds to a specific value of $\beta$. In practice, for a video streaming client, the initial access delay and download rate directly affect the initial buffer time as well as the number of stalls during the video session, which are the two key factors determining users' QoE [126]. Therefore, it is imperative to find an efficient solution to problem (25) that optimizes the tradeoff between users' download rate and content access delay, while ensuring proportional fairness among the users.

### 3.5.2 Proposed CARS Algorithm

Let $\mathcal{U}_r$ denote the set of users that BS $r\in\mathcal{R}$ is associated with, i.e., $\mathcal{U}_r=\{u\in\mathcal{U}\,|y_{ur}=1\}$, and $|\mathcal{U}_r|$ be the cardinality of $\mathcal{U}_r$. The objective function (25a) can be seen as the sum of rate-delay tradeoff utility of all BSs in the network. In particular, for a given request scheduling policy $\Psi$, the tradeoff utility of BS $r\in\mathcal{R}$ can be defined by,

$$\Omega_r\left(\mathcal{U}_r\right)=\sum_{u\in\mathcal{U}_r}\log\left(\frac{\phi_{ur}}{|\mathcal{U}_r|}\right)+\beta\,|\mathcal{U}_r|\sum_{i\in\mathcal{F}}p_{ir}S_{ir}. \tag{26}$$

It can be seen that the objective function (25a) is now equivalent to $\sum_{r\in\mathcal{R}}\Omega_r\left(\mathcal{U}_r\right)$. To facilitate the analysis, we will now show that $\Omega_r\left(\mathcal{U}_r\right)$ is monotone submodular.

**Lemma 5.** *If $\phi_{ur} \geq Ue$ [bits/s], $\forall u \in \mathcal{U}$, where $U = |\mathcal{U}|$ is the total number of users and $e$ is the Euler's constant, then the tradeoff utility function $\Omega_r(\cdot)$ defined by (26) is monotone.*

*Proof.* Let $\mathcal{W} \subset \mathcal{U}$, $v \in \mathcal{U}$ and $v \notin \mathcal{W}$. We will show that adding user $v$ to the set $\mathcal{W}$ will not decrease the utility value $\Omega_r$. In particular, the marginal value of adding $v$ to the set $\mathcal{W}$, denoted as $\Omega_{r,\mathcal{W}}(v)$, can be calculated as,

$$\Omega_{r,\mathcal{W}}(v) = \Omega_r(\mathcal{W}+v) - \Omega_r(\mathcal{W}) \tag{27a}$$

$$= \beta \sum_{i\in\mathcal{F}} p_{ir} S_{ir} + \sum_{u\in\mathcal{W}\cup\{v\}} \log\left(\frac{\phi_{ur}}{|\mathcal{W}|+1}\right) - \sum_{u\in\mathcal{W}} \log\left(\frac{\phi_{ur}}{|\mathcal{W}|}\right) \tag{27b}$$

$$= \beta \sum_{i\in\mathcal{F}} p_{ir} S_{ir} + \log(\phi_{vr}) - \Lambda(|\mathcal{W}|), \tag{27c}$$

in which the function $\Lambda(\cdot)$ over $z > 0$ is defined as,

$$\Lambda(z) \triangleq \log(z+1) + \log\left(z + \frac{1}{z}\right)^z. \tag{28}$$

Since $\mathcal{W} \subset \mathcal{U}$, it holds that $|\mathcal{W}|+1 \leq |\mathcal{U}| = U$. Furthermore, recall that $\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$, where $e$ is the Euler's constant. Therefore it follows from (28) that,

$$\Lambda(|\mathcal{W}|) \leq \log U + \log\left(1 + \frac{1}{|\mathcal{W}|}\right)^{|\mathcal{W}|} = \log(Ue). \tag{29}$$

From (27c) and (29), we have,

$$\Omega_{r,\mathcal{W}}(v) \geq \beta \sum_{i\in\mathcal{F}} p_{ir} S_{ir} + \log(\phi_{vr}) - \log(Ue). \tag{30}$$

Therefore, if $\phi_{vr} \geq Ue$ [bits/s], $\forall v \in \mathcal{U}$, we have $\Omega_{r,\mathcal{W}}(v) \geq 0$, which confirms that $\Omega_r(\cdot)$ is monotone. □

**Remark 5**: Note that for Lemma 5 to hold, it is required that the data rate (measured in bits/s) of user $u$ scheduled to BS $r$, $\phi_{ur}$, is greater than $e \approx 2.72$ times the number of active users $U$ between any two consecutive departures, which is trivially satisfied in any real-world system. Such threshold is anyway necessary in an LTE system since a user

whose channel power is weaker than a minimum threshold will never be selected as it cannot support the smallest Modulation and Coding Scheme (MCS). □

**Lemma 6.** *The tradeoff utility function* $\Omega_r(\cdot)$ *defined by* (26) *is submodular.*

*Proof.* Let $\mathcal{W} \subset \mathcal{T} \subset \mathcal{U}$ and $v \in \mathcal{U}, v \notin \mathcal{T}$. The marginal value of adding $v$ to the sets $\mathcal{W}$ and $\mathcal{T}$ are, respectively,

$$\Omega_{r,\mathcal{W}}(v) = \beta \sum_{i \in \mathcal{F}} p_{ir} S_{ir} + \log(\phi_{vr}) - \Lambda(|\mathcal{W}|), \tag{31}$$

$$\Omega_{r,\mathcal{T}}(v) = \beta \sum_{i \in \mathcal{F}} p_{ir} S_{ir} + \log(\phi_{vr}) - \Lambda(|\mathcal{T}|). \tag{32}$$

By taking the derivative of $\Lambda(z)$ in (28) with respect to (w.r.t.) $z$, it can be verified that $\Lambda(\cdot)$ is an increasing function with $z > 0$. Hence, since $|\mathcal{W}| < |\mathcal{T}|$, it follows that $\Lambda(|\mathcal{W}|) < \Lambda(|\mathcal{T}|)$. As a result, from (31) and (32), we have $\Omega_{r,\mathcal{W}}(v) > \Omega_{r,\mathcal{T}}(v)$. This confirms that $\Omega_r(\cdot)$ is a submodular function. □

From Lemmas 5 and 6, it is confirmed that the objective function in problem (25) is monotone submodular. Note that a similar result can be obtained by considering the average content-access-delay saving $\beta \sum_{i \in \mathcal{F}} p_{ir} S_{ir}$ as a bias term added to the log-rate of each user in (26). It is shown in [127] that adding the non-negative bias term does not affect the submodularity and monotonicity of the log-rate function. With this position, one can employ a greedy algorithm that assigns each new arriving user to the BS whose added marginal utility value gets maximized. Based on the classical result in online combinatorial auction [124,128], such greedy algorithm achieves at least $\frac{1}{2}$ the optimal objective value. As we consider the scheduling during each time window $[t_n, t_{n+1})$ between two consecutive departures, the performance bound can be generalized for the whole considered period of time with any $N$ departures. The proposed online Cache-Aware Request Scheduling (CARS) is summarized in Algorithm 6.

Step 1 in Algorithm 6 initiates the scheduling algorithm any time there is a new request, which is coupled with the arrival of a new user. Step 2 starts the loop to evaluate the utility marginal value associated with assigning the incoming user to each of the BSs, as calculated

**Algorithm 6** Cache-Aware Request Scheduling (CARS)

1: **for** each new arrival, user $u$ requests for content $i$ **do**
2: **for** $r \in \mathcal{R}$ **do**
3: Calculate the utility marginal value at each BS $r$ w.r.t. the current associated user set $\mathcal{U}_r$,

$$\Omega_{r,\mathcal{U}_r}(u) = [\Omega_r(\mathcal{U}_r + u) - \Omega_r(\mathcal{U}_r)] \tag{33}$$

4: Select $k = \arg\max_{r \in \mathcal{R}} \Omega_{r,\mathcal{U}_r}(u)$
5: Direct user $u$ to download $i$ from BS $k$,

$$\mathcal{U}_k \leftarrow \mathcal{U}_k + u \tag{34}$$

in step 3. Step 5 selects the BS that gets the highest marginal value when associating with the new user and step 6 directs this user to download content from the selected BS.

**Remark 6**: The complexity of Algorithm 6 mainly comes from the iteration process in step 2; thus, it increases linearly with the number of BSs, e.g., $\mathcal{O}(R)$. □

### 3.6 Performance Evaluation

We present now numerical results to evaluate the performance of the proposed caching and request scheduling strategies. In particular, we evaluate the effects of the proposed cooperative hierarchical caching architecture, cache-management, and cache-aware request scheduling algorithms compared to the existing approaches. Thorough simulations are carried out using both real-world trace-based and synthetic content requests. In the following, we describe the simulation settings followed by different performance evaluation scenarios in the corresponding subsections.

We consider a C-RAN system with $R = 7$ hexagonal cells, with 6 cells forming a ring around one central cell. Each cell has one BS located at the cell's center and the distance between two BSs of the two neighboring cells is 1 km. It is assumed that the backhaul and fronthaul links' capacities are sufficiently provisioned to handle all the generated traffic requests. We consider the content requests being video requests that arrive one-by-one to the network following a Poisson process with arrival rate equals to $\lambda R$ [reqs/min]. Each content request is coupled with a user arrival, and the location of the user is randomly and uniformly placed in the network's coverage area.

The end-to-end (e2e) latency of retrieving a video from the CDN to the CPU, $d_0$, is considered to be 80 ms [1], while the e2e latency of transferring a video between the BSs and the CPU, $d_r$'s, are randomly assigned, following a uniform distribution in the range $[10 - 30\text{ms}]$ [94]. In addition, we consider that the size of each video is 20 MB and the duration of each video is 2 min. For brevity of presentation, we consider a representative case where the cloud-cache and each of the 7 edge-caches are allocated 30% and 10% of the total cache capacity, respectively. The extension to different cache capacity allocation is straightforward.

To compare the performance of different cache management schemes, we consider three key metrics: *(i) cache hit ratio*: the fraction of requests that can be retrieved from one of the caches; *(ii) average access delay* [ms]: average latency of the contents traveling from the caches or the CDN server to the requesting user; and *(iii) backhaul traffic load* [TB]: the volume of traffic going through the backhaul network due to users downloading contents from the CDN servers.

Unless otherwise stated, the simulation results are based on the YouTube request trace data collected on the University of Massachusetts' Amherst campus during the day of March 12th, 2008 [129]. We consider the content files being the requested videos, whereas the video popularity is extracted directly from the trace, which consists of $122,280$ requests for $77,414$ different videos.

### 3.6.1 Impact of Cooperative Cloud-cache

Here, we evaluate the benefits of the proposed caching scheme from the architectural perspective by comparing against traditional caching architectures as described below.

- *Edge Non-Cooperative (EdgeNC)*: This scheme only caches most popular files at the edge-caches and these cache entities do not cooperate with each other. If the requested file from a mobile user is found in the cache of the serving BS, the file will be downloaded immediately from the cache; otherwise, it will be fetched from the CDN server.

[1] Refer to: 3GPP TS 23.203, V13.5.1, Sept. 2015

- *Edge Cooperative (EdgeCP)*: This scheme only caches most popular files at the edge-caches while allowing the BSs to retrieve contents from other edge-caches using fronthaul links.

- *Hierarchical Non-Cooperative (HrchNC)*: This is a hierarchical caching scheme employing both the edge-caches and the cloud-cache; these cache entities, however, do not cooperate with each other. Each edge-cache stores the most popular files seen by the corresponding BS and the cloud-cache stores the most popular files based on global popularity.

- *Octopus*: This is our proposed cooperative hierarchical caching scheme employing both the edge-caches and the cloud-cache; each BS can retrieve contents from the cloud-cache or from other edge-caches using fronthaul links.

To ensure a fair comparison, we set the total capacity of all caches in each architecture to be the same. Figure 3.3(a-c) compares the performance of the considered four caching architectures using three key performance metrics: (a) cache hit ratio, (b) average access delay [ms], and (c) backhaul traffic load [TB]. The relative total cache capacity $M$ is measured as the fraction of the total content library size. Note that, when $M = 0.2$ and $M = 0.3$, the backhaul traffic load of *HrchCP* architecture is zero. Observe that the proposed caching architecture, Octopus, significantly improves cache hit ratio and reduces average access delay as well as backhaul traffic load compared to the other baselines for any total cache capacity value. On the other hand, deploying hierarchical caching alone without cooperation as in *HrchNC* will result in performance degradation compared to *EdgeCP*.

### 3.6.2 Impact of Proposed Cache Management Algorithms

In this subsection, considering a hierarchical caching system, we evaluate the performance of different cache-management algorithms that identify the files to be stored in each cache. In particular, we compare the proposed Octopus scheme (which employs the UHPCD and UHRCR algorithms) with five baselines below.

- *Layered-aware Cooperative Caching (LCC)*: In this scheme, we extend the LLC algorithm proposed in [110] to exploit the cooperative hierarchical caching architecture

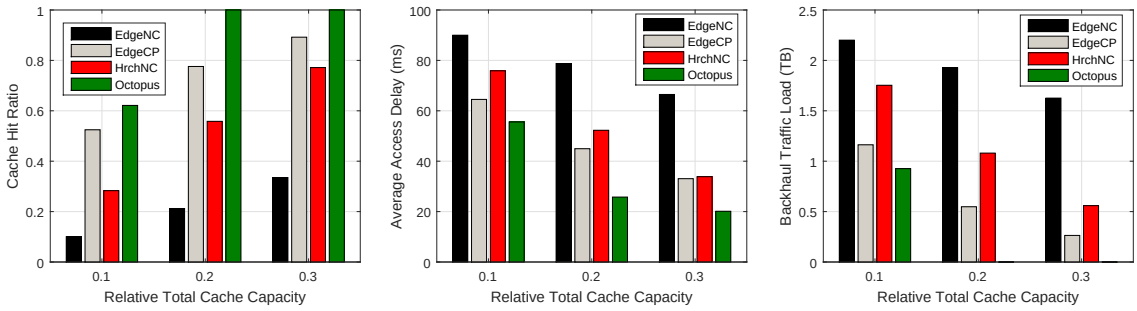



Figure 3.3: Comparison of different caching architectures using three key performance metrics: (a) cache hit ratio, (b) average access delay, and (c) backhaul traffic load.

in C-RAN. In particular, in addition to the local edge-cache at each BS, we add the cloud-cache that stores the most popular files according to the global content popularity. Since we do not consider layered-videos, the algorithm will treat each file as one video with a single layer.

- *Exclusive Most Popular Caching (ExMPC)*: In this scheme, each edge-cache stores the most popular files based on the content popularity observed at the corresponding BS. The cloud-cache stores the most popular files based on the global content popularity observed at the CPU, excluding the files that have been cached at the edge-caches. This scheme is the realization of the greedy cache-placement algorithm in [130] for inter-level cache cooperation. The exclusive mechanism in ExMPC avoids the redundancy in the pure MPC scheme [114, 115, 131] as the same files might be cached at both the edge and cloud layers.

- *FemtoCaching Extension (FemtoX)*: This scheme is an extension of the FemtoCaching [91] to a hierarchical caching system in C-RAN. In FemtoCaching, the femtocell-like BSs act like *helpers* with weak backhaul links but large storage capacity. These helpers form a distributed caching network that assists the macro BS by handling requests and caching content according to a greedy algorithm. In FemtoX in this simulation, we map each helper's cache in FemtoCaching to an edge-cache and introduce the additional cloud-cache.

- *Least Frequently Used (LFU)*: We apply the LFU scheme [132] to the hierarchical caching system; when the cache is full and if there is a cache miss, LFU fetches the file from the CDN server and replaces it with the file in the cache that has been least frequently used.

- *Least Recently Used (LRU)*: This scheme is analogous to the LFU scheme; however, when the cache is full, it chooses to evict the file that has been least recently used. The cache hit ratio of LRU scheme depends on the overlap of content requests of the active users in the local RAN.

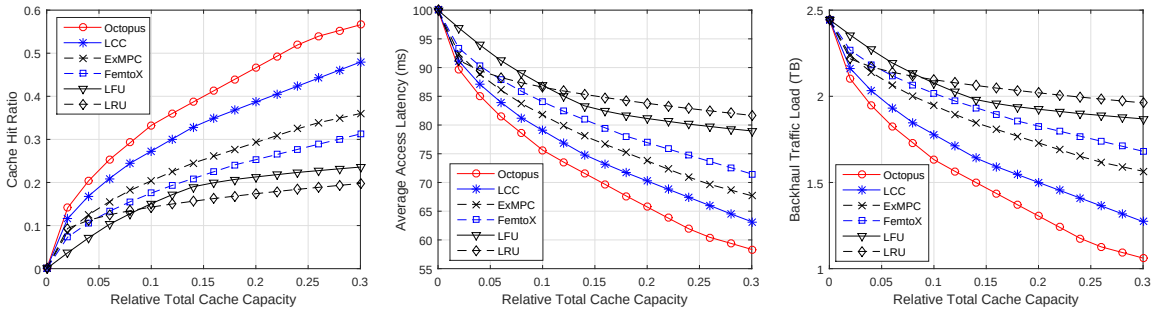



Figure 3.4: Comparison of different cache management policies using three key performance metrics: (a) cache hit ratio, (b) average access delay, and (c) backhaul traffic load.

Again, we use the YouTube trace for this simulation. In Fig. 3.4(a-c), we compare the performance of Octopus caching policy with the five baselines. It can be seen that Octopus always achieves superior performance in terms of cache hit ratio, average access delay, and backhaul traffic load. This is because the UHPCD algorithm reduces the redundancy among the caches compared to ExMPC and LLC schemes and the UHRCR algorithm updates the caches upon cache misses.

### 3.6.3 Impact of Cache-aware Request Scheduling

Here, we evaluate the performance of our proposed cache-ware request scheduling, CARS, presented in Sect. 3.5. We assume that the locations of the users making request to each BS $r$ follow a uniform distribution inside $r$'s cell area. In order to obtain a scenario where content popularities are different at different BSs, we randomly shuffle the popularity distribution

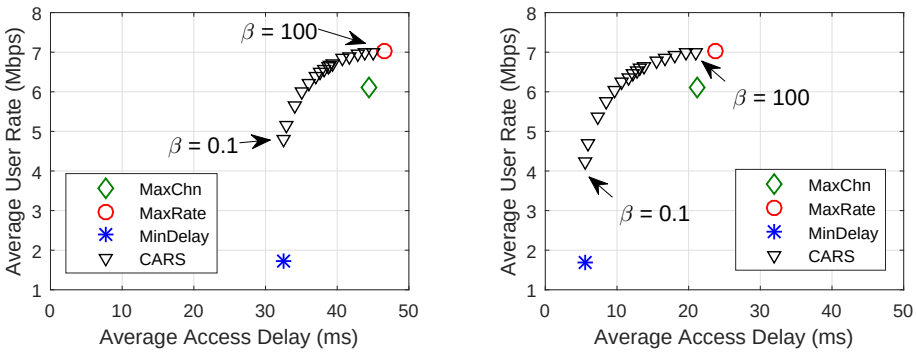
extracted from the YouTube trace. For the radio network setting, we assume the transmit power of each BS to be $P_r = 20$ dBm, $\forall r \in \mathcal{R}$, and the background noise power to be $\sigma^2 = -100$ dBm. The channel gains are generated using distance-dependent path-loss model, given as $L\,[\text{dB}] = 140.7 + 36.7\log_{10} d_{[\text{km}]}$, and the log-normal shadowing variance is set to 8 dB. In addition, the channel bandwidth is set to $W = 10$ MHz.

We first compare the tradeoff between average user rate and content access delay of the proposed CARS algorithm with the three baselines described below that decide how to associate a new arriving user to a BS. In our simulation, the average user rate and content access delay are calculated over the one-day period for which the YouTube request trace [129] was recorded.

- *MaxChn*: the user is associated to the BS having the strongest channel gain to the user.
- *MaxRate*: the user is associated to the BS that maximizes the user rate utility, similar to the algorithm in [104].
- *MinDelay*: the user is associated to the BS that minimizes the content access delay.

In Figs. 3.5 and 3.6, we compare the tradeoff performance of the four algorithms while varying the total cache capacity and request arrival rate, respectively. Notice that for each given cache capacity and request arrival rate, the tradeoff points of the three baseline algorithms are given by fixed points. In all cases, the *MaxRate* scheme provides the highest average user rate with the highest average access delay while the *MinDelay* scheme provides the lowest average user rate with lowest average access delay. For each subfigure, different values of the tradeoff factor $\beta$ are given by the series $[0.1, 1, 2, ..., 9, 10, 15, 20, 30, 50, 100]$, which corresponds to different tradeoff points of the CARS scheme, from bottom-left to top-right on the plot, respectively. Note that when $\beta = 0$, CARS becomes *MinDelay*; whereas when $\beta = \infty$, CARS becomes *MaxRate*. The tradeoff performance of CARS scheme can be controlled by varying the tradeoff factor $\beta$.

It can be seen in both figures that when $\beta$ is small, CARS can achieve similar average access delay as that of *MinDelay* scheme with significantly higher average user rate; on the




(a) $M = 10\%$ (b) $M = 20\%$

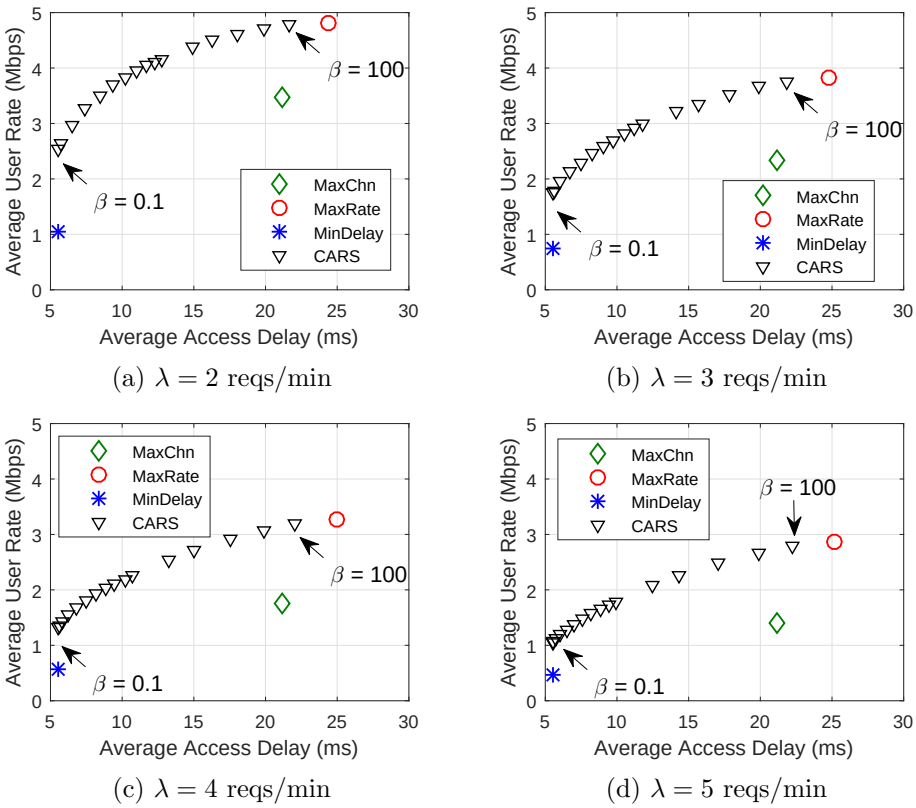
(c) $M = 30\%$ (d) $M = 40\%$

Figure 3.5: Tradeoff between average user rate utility and content access delay of different request scheduling algorithms: the total cache capacity $M$ in (a-d) is given as % of the total content library size, $\lambda = 1$ reqs/min, $\beta = [0.1, 1, 2, ..., 9, 10, 15, 20, 30, 50, 100]$ (from bottom left to top right).

other hand, when $\beta$ is large, CARS achieves an average user rate that approaches that of the *MaxRate* scheme, albeit with significantly lower average access delay when the request arrival rate $\lambda$ is 2 or higher, as seen in Fig. 3.6. In summary, the proposed CARS scheme allows for flexible control of the average user rate and content access delay tradeoff while achieving superior performance over the existing baselines in wide ranges of cache capacity and request arrival rate.

### 3.6.4 Impact of Content Popularity Skewness

In the previous subsections, using the YouTube request trace, we have demonstrated the superior performance of our proposed caching (Octopus) scheme and request scheduling scheme (CARS) over traditional policies. Here, in order to generalize the results, we employ an analytical content-request model where the popularity distribution of the files is assumed




(a) $\lambda = 2$ reqs/min (b) $\lambda = 3$ reqs/min

(c) $\lambda = 4$ reqs/min (d) $\lambda = 5$ reqs/min

Figure 3.6: Tradeoff between average user rate utility and content access delay of different request scheduling algorithms: $M = 20\%$ total content library size and $\beta = [0.1, 1, 2, ..., 9, 10, 15, 20, 30, 50, 100]$ (from bottom left to top right).

to follow a Zipf distribution [133, 134]. In particular, the request probability of the $i$-th most popular content (among the set of $F$ contents) at BS $r$ can be calculated as $p_{ir} = \frac{1/i^{\alpha}}{\sum_{n=1}^{F} 1/n^{\alpha}}, \forall r \in \mathcal{R}$, where $\alpha$ is the Zipf skewness parameter, which determines the rate of popularity decline as $i$ increases. The observed value of $\alpha$ might vary from different measurements; however, it was estimated that $\alpha$ ranges from 0.64 to 0.83 based on the measurements of [135, 136]. To generate the synthetic content requests, we consider a library of 10,000 files with equal size of 20 MB. We randomly generate 100,000 requests following the Zipf-based popularity distribution with $\alpha \in [0.6, 0.7, 0.8]$.

Figure 3.7(a-c) depicts the performance of Octopus caching scheme over the baselines with different content popularity distributions. Observe that, as $\alpha$ increases, the performance of Octopus scheme in terms of cache hit ratio, average content access delay, and backhaul traffic load significantly improves. In all cases, Octopus always performs the best;

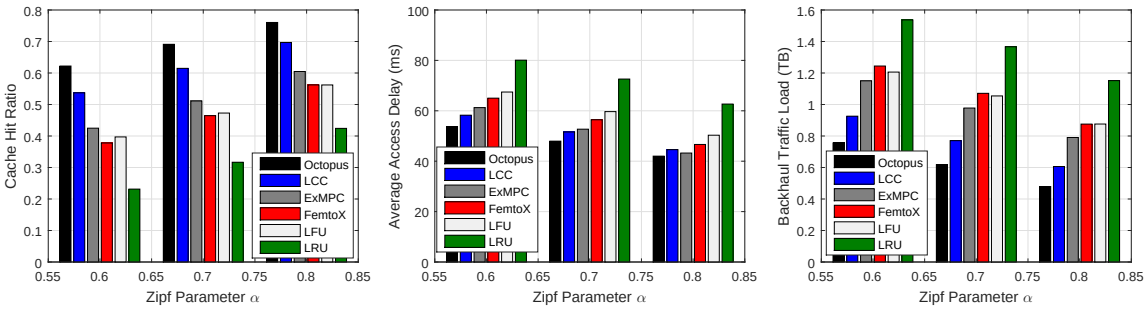



Figure 3.7: Performance of different cache management policies with synthetic content requests generated using the Zipf-based popularity distribution; $M = 30\%$ library size and Zipf parameter $\alpha \in [0.6, 0.7, 0.8]$.

however, as $\alpha$ increases, the performance gaps between Octopus and the baselines become smaller.

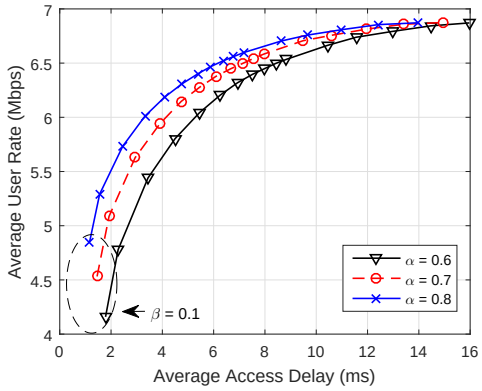



Figure 3.8: Performance of the proposed CARS policy with synthetic content requests generated using the Zipf-based popularity distribution; $M = 30\%$ library size, $\beta = 100$, $\lambda = 1$ reqs/min, Zipf parameter $\alpha \in [0.6, 0.7, 0.8]$, and the tradeoff factor $\beta = [0.1, 1, 2, ..., 9, 10, 15, 20, 30, 50, 100]$ (from bottom left to top right).

In Fig. 3.8, we compare the performance of the proposed CARS policy with different content popularity distributions. It can be seen that, as $\alpha$ increases, the average user rate and access delay tradeoff performance of CARS is improved. In other words, with the same average access delay, CARS yields higher average user rate when $\alpha$ is larger.

## 3.7 Summary

In this chapter, we introduced a cooperative hierarchical caching paradigm for Cloud Radio Access Networks (C-RAN) with the deployment of the cloud-cache at the Central Processing Unit (CPU) and the distributed edge-caches at the Base Stations (BSs). We proposed a low-complexity, efficient cache-management strategy, Octopus, comprising of two low-complexity, utility-based hierarchical caching algorithms. Furthermore, we proposed an online Cache-Aware Request Scheduling (CARS) algorithm that provides a competitive tradeoff between the content download rate and content access delay. We carried out extensive simulations using both a real-world YouTube video request trace and the Zipf-based synthetic content request model. We demonstrated that our cache-management strategy outperforms traditional caching deployment architectures and cache management algorithms in terms of cache hit ratio, average content access delay, and backhaul traffic load. Additionally, we illustrate that our CARS algorithm achieves superior tradeoff performance over existing approaches that optimize either users' rate or access delay alone.

**Discussion on Practical Implementation:** The low complexity of the cache management and online content request scheduling algorithms proposed in this work greatly facilitates the implementation of real C-RAN systems. In particular, the decision-making CCM can be implemented as a logical entity, co-located at the CPU, that periodically collects information from each cache node, such as user request logs and temporal channel gains. This information can be collected through the outband interface such as the IP/MPLS interface implemented in the commercial mobile caching system. In order to integrate efficiently the proposed caching system into C-RAN, however, there are several technical challenges that need to be addressed. First, due to the centralized nature of our cooperative hierarchical caching scheme, the issue of scalability is of concern. This could be addressed by implementing the CCM and the learning agent for content popularity estimation in the resourceful CPU. Second, not all BSs are suitable for cache deployment or the cache provisioned at each BS needs not be the same. Depending on the population of users at different locations, the number of edge-cache nodes and the cache capacities should be carefully selected. Last, but not least, the impact of user mobility should be taken into

account; for example, if the mobility pattern of users can be predicted, the CCM might be able to make informed decision and provision interested content at the cells where a user is moving to.

# Chapter 4

# Adaptive Bitrate Video Caching and Processing in MEC Networks

In this chapter, we propose a joint collaborative caching and processing framework that supports Adaptive Bitrate (ABR)-video streaming in MEC network. We formulate an Integer Linear Program (ILP) that determines the placement of video variants in the caches and the scheduling of video requests to the cache servers so as to minimize the expected backhaul cost of video retrieval. The considered problem is challenging due to its NP-completeness and to the lack of a-priori knowledge about video request arrivals. Our approach decomposes the original problem into a cache placement problem and a video request scheduling problem while preserving the interplay between the two. We then propose practically efficient solutions, including: (i) a novel ABR-aware proactive cache placement algorithm with provable approximation performance when video popularity is available, and (ii) an online low-complexity video request scheduling algorithm that performs very closely to the optimal solution. Simulation results show that our proposed solutions achieve significant increase in terms of cache hit ratio and decrease in backhaul traffic and content access delay compared to the traditional approaches.

## 4.1 Introduction

**Motivation:** Over the last few years, the proliferation of Over-The-Top (OTT) video content providers (YouTube, Amazon Prime, Netflix,...), coupled with the ever-advancing multimedia processing capabilities on mobile devices, have become the major driving factors for the explosion of on-demand mobile video streaming. According to the prediction of mobile data traffic by Cisco, mobile video streaming will account for 72% of the overall mobile data traffic by 2019 [137]. While such demands create immense pressure on mobile

network operators, distributed edge caching has been recognized as a promising solution to bring video contents closer to the users, to reduce data traffic going through the backhaul links and the time required for content delivery, as well as to help in smoothing the traffic during peak hours. In wireless edge caching, highly sought-after videos are cached in the cellular Base Stations (BSs) or wireless Access Points (APs) so that demands from users to the same content can be accommodated easily without duplicate transmissions from original content servers.

In wireless video streaming, users' preference and demand towards specific quality and/or format of a video might be different due to the high dynamics of network condition, coupled with the the heterogeneity of users' processing capabilities. For example, users with highly capable devices and fast network connection usually prefer high resolution videos while users with low processing capability or low-bandwidth connection may not enjoy high quality videos because the delay is large and the video may not fit within the device's display. By leveraging such behavior, Adaptive Bitrate (ABR)-streaming techniques [138, 139] have been widely used in commercial Content Delivery Network (CDN) to improve the users' Quality of Experience (QoE). In ABR-video streaming, different bitrate versions of a video, referred to as *video variants* hereafter, can be generated and transmitted to the users according to their devices' capabilities, network connection, and specific requests. While a video variant can be transcoded from another variant of the same video [140], existing video caching systems often overlook this dependency and offer each video variant as an independent stream (data file) to the user, resulting in the caching inefficiency.

**Our vision:** In contrast to most of the existing works on video caching (see related works in Sect. 4.2) which are not ABR-aware and mainly rely on the "store and transmit" mechanism without any processing, we propose a framework to utilize both caching and processing capabilities at the MEC servers to satisfy users' requests for videos with different bitrates. To the best of our knowledge, we are the first to introduce a collaborative joint caching and processing framework in MEC networks. Specifically, owing to their real-time computing capability, MEC servers can perform transcoding of a video to different variants to satisfy the user requests. Each variant is a bitrate version of the video and

we consider that a lower bitrate variant can be obtained from a higher bitrate variant via transcoding [141]. For example, a video at bit-rate of 5 Mbps (720p) can be transcoded from the same video at bitrate of 8 Mbps (1080p). Moreover, we extend the collaborative caching paradigm to a new dimension where the MEC servers can assist each other to not only provide the requested video via backhaul links but also transcode it to the desired variants (for example, when the requesting server is overloaded, the other servers can help to perform the transcoding tasks). In this way, the requested variant of a video can be transcoded by any MEC server on the delivery path from where the original video is located (data provider node) to the home MEC server (delivery node) of the end user. The potential benefits of this strategy is three-fold: (i) the original remote content server does not need to generate all different variants of the same video, (ii) users can receive videos that are suited for their network condition and multimedia processing capabilities as content adaptation is more appropriately done at the network edge, and (iii) collaboration among the MEC servers enhances cache hit ratio and balance processing load in the MEC network.

**Challenges and contributions:** The proposed framework, however, faces several challenges. Firstly, caching multiple variants of a videos incurs high overhead in terms of storage which is a limited resource at the network edge. In order to improve the cache efficiency, one needs to take into account the transcoding dependency among different video variants, which in turn adds up the complexity of traditional cache placement problem where different video files are treated independently. Secondly, real-time video transcoding is a computation-intensive task and transcoding of a large number of videos simultaneously might quickly exhaust the available processing resources on the MEC servers. Therefore, it is very important to design a caching and request scheduling scheme that efficiently *utilizes both the given cache and processing resources*. To this end, we make the following contributions in this work.

- We formulate the collaborative joint caching and processing problem as an Integer Linear Program (ILP) that minimizes the average backhaul network cost of delivering videos to all users, subject to the cache and processing capacity constraints at each MEC server.

- To overcome the complexity of the considered ILP and the lack of *a-priori* knowledge about video request arrivals, we follow the "divide and conquer" approach to decompose the original problem into two subproblems, i.e., a cache placement problem and a request-scheduling problem.

- We address the cache placement problem in two scenarios: (i) when content popularity is not available, we adopt the popular Least Recently Used (LRU) caching policy; and (ii) when content popularity is available, we propose a novel ABR-aware proactive cache placement algorithm with provable approximation performance.

- We propose a low-complexity online request scheduling algorithm while showing that no non-trivial competitive algorithms existed for the request scheduling problem in the general case. We show via numerical simulations that our proposed online algorithm performs very closely to the optimal solution using global optimization solver with exponential complexity.

- We carry out extensive simulations to evaluate the performance of the proposed solutions over traditional caching approaches, and show that our proposed collaborative joint caching and processing framework provides significant increase in terms of cache hit ratio and decrease in backhaul traffic and content access delay.

**Chapter Organization:** The remainder of this chapter is organized as follows. In Sect. 4.2, we review the related literature and highlight the novelties and significance of the proposed work. In Sect. 4.3, we present the caching system model and formulate the underlying optimization problem. In Sect. 4.4, we propose efficient solutions to the considered collaborative joint caching and processing problem. Sect. 4.5 presents our simulation results. Finally, we draw the conclusions in Sect. 4.6.

## 4.2 Related Work

In general, the use of content caching to reduce backhaul traffic and content access delay has been extensively studied and adopted in the CDN (cf. [142–147] and the references

therein). Differently from the CDN settings, considerable recent research efforts have focused on content caching in wireless networks [91, 92, 105]. For example, in [91], the notion of femtocaching is introduced, in which the femtocell-like BSs are used to form a distributed caching network that assists the macro BS to handle requests of popular files. To overcome the cache-size limitation at individual BSs, collaborative caching has been exploited in small-cell networks [94–96, 109] and among the operators [110]. Along this line, the authors in [95] propose online collaborative caching algorithms that aim at minimizing the total cost paid by the content providers without requiring prior knowledge about the content popularity. In the context of C-RAN, we proposed in [42, 148] a cooperative hierarchical caching strategy where the cloud-cache is introduced as a bridging layer between the edge-based and core-based caching schemes. In [149], the authors proposed a coordinated data assignment algorithm to minimize the network cost with respect to both the precoding matrix and the cache placement matrix in a C-RAN. In [150], a coded-caching scheme is proposed for C-RAN that takes into account user mobility in order to minimize the total energy consumption of the network including the transport and the caching energy consumptions. The energy-efficiency performance comparison of coded and uncoded caching in wireless network was investigated in [151].

Given the benefits of ABR streaming in improving the quality of delivered video over the rate-varying connections, several commercial techniques have been deployed including Apple HTTP Live Streaming (HLS) [152], Microsoft Smooth Streaming [153], and Dynamic Adaptive Streaming over HTTP (DASH) [138], etc. The experiment of ABR streaming using Software Defined Networking (SDN) was carried out in [154] where the proposed system utilizes the information on network condition and cache contents to guide video streaming clients for improved QoE. Lee at al. [155] investigated the bitrate oscillations problem caused by the interactions between a DASH client and a cache server and proposed a rate adaptation approach to address this problem.

In summary, most existing works on wireless caching and ABR streaming mainly focused on their respective problems of cache placement and rate adaptation while there was little study on the connection between the two technologies. To account for ABR video streaming, several works have focused on the use of caching with Scalable Video Coding (SVC) [110,

156, 157]. In these approaches, each video is encoded into multiple SVC layers that can be combined at the end user to render the requested video quality. In [156], the authors study multi-layer video streaming and propose a network coding scheme that combines inter-layer with intra-layer coding to determine the distribution of video packets to the cache nodes. The work in [157] proposes a heuristic-based solution to the problem of joint video caching (video-layer placement) and scheduling as a reward maximization problem. Furthermore, a collaborative caching scheme of layered videos is studied in [110] where the authors propose an approximate solution to the cache placement problem using a connection with the multiple-choice knapsack problem. However, SVC was not preferred in industry in the past, which is partly due to the lack of hardware decoding support in most traditional mobile devices. Another important factor that limits the applicability of SVC is that the decoding of multiple video layers may significantly increase power consumption on mobile devices whose battery capacity is always limited.

In contrast with the prior approaches using SVC, our work considers a multi-bitrate video caching framework in which different variants of the video are generated (via transcoding) and stored at the cache nodes, thus requiring minimal video decoding at the end users. While this use of multi-bitrate video caching has previously been considered by Shen et al. [158] and Ahlehagh et al. [141, 159, 160], they are fundamentally different from our work in three aspects. First, these works only study a caching system with one cache node, as opposed to the *collaborative* framework of multiple caching/processing servers in our work. Second, these works consider offline request scheduling approaches that resolve the optimization problem from scratch every time there is a new request arrival; this approach is inefficient and non-scalable due to the high complexity of the optimal solution while it results in re-directing large numbers of pre-scheduled requests. Third, only simple reactive cache placement algorithms were considered in these works; whereas we consider both reactive cache placement when content popularity is not available and proactive ABR-aware cache placement when content popularity is available.

## 4.3 System Model and Problem Formulation

In this section, we present the envisioned distributed caching system deployed on MEC networks, followed by the settings of the considered model; finally the joint caching and request scheduling optimization problem is formulated. Table 5.1 summarizes the key parameters used in this chapter.

Table 4.1: Summary of Key Parameters for MEC Caching System

| Symbol | Description |
|---|---|
| $\mathcal{K}$ | Set of $K$ BSs/MEC servers |
| $\mathcal{T}$ | Set of videos available for download |
| $L$ | Number of bitrate variants each video has |
| $\mathcal{V}$ | Set of all video variants |
| $v_l$ | $l$th variant of video $v$ |
| $r_l$ | Size of $v_l$ |
| $p_{hl}$ | Computing cost of transcoding $v_h$ to $v_l$ |
| $c_j^{v_l}$ | Indicator of placement of $v_l$ at BS $j$ |
| $M_j$ | Storage capacity of cache server at BS $j$ |
| $P_j$ | Processing capacity of cache server BS $j$ |
| $x_j^{v_l}$ | Indicator of service (IoS) for $v_l$ directly from cache of BS $j$ |
| $y_j^{v_l}$ | (IoS) for $v_l$ by transcoding at BS $j$ |
| $z_{jk}^{v_l}$ | (IoS) for $v_l$ at BS $j$ by retrieving from BS $k$ |
| $t_{jk}^{v_l}$ | (IoS) for $v_l$ at BS $j$ by transcoding at BS $k$ |
| $w_{jk}^{v_l}$ | (IoS) for $v_l$ at BS $j$ by retrieving from BS $k$ and transcoding at BS $j$ |
| $d_{jk}$ | Backhaul cost of retrieving unit-size video from BS $k$ to BS $j$ |
| $d_{j0}$ | Backhaul cost of retrieving unit-size video from remote server to BS $j$ |
| $\mathcal{N}_j^t$ | Set of video requests being served at BS $j$ at time $t$ |
| $f_j^{v_l}$ | Probability that a request sent to BS $j$ is for video $v_l$ |
| $\lambda_j$ | Average video request arrival rate at BS $j$ |

### 4.3.1 MEC-based Caching System Architecture

As shown in Fig. 4.1, a MEC network consists of multiple MEC servers connected via backhaul links. Each MEC server is deployed side-by-side with the BS in a cellular RAN, providing computation, storage and networking capabilities to support context-aware and delay-sensitive applications in close proximity to the users. In this work, we envisage the use of the MEC server for both caching (i.e., video storage) and processing (i.e., video

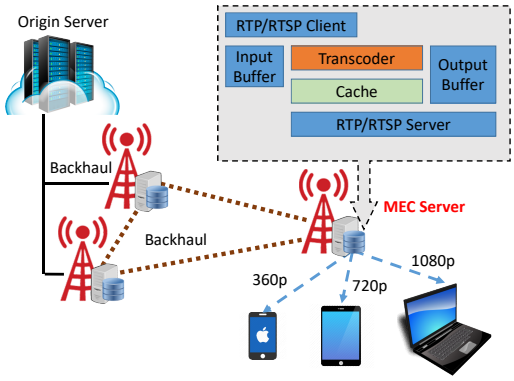



Figure 4.1: Illustration of collaborative video caching and processing framework deployed on MEC network. The cache server implemented on MEC server acts as both RTP/RTSP client and server.

transcoding). The use of MEC server as a caching server is similar to the use of cache proxy server in the Internet [158]; however, we consider here a MEC network where the connected MEC servers could share their cached content and assist each other for video transcoding. In particular, each MEC server is equipped with a cache unit and a transcoding unit (transcoder). Each cache unit in a MEC server acts as a client to the origin content server (in the Internet) and to other peer cache units (from other MEC servers). An RTP/RTSP client is built into the server to receive the streamed content from other servers via backhaul links and put it into the input buffer. If needed, the transcoder will transcode the input stream to a desired bitrate stream and pushes it out to the output buffer; otherwise the input buffer is directly moved to the cache and/or output buffer for transmitting to the end users. Here, an RTP/RTSP server is built within each MEC server to stream the video to the end users and to other servers. The data in the output buffer is obtained either from the transcoder or from the cache.

In Fig. 4.2, we illustrated the possible (exclusive) events that happen when a user request for a video; these events are then associated with different video request scheduling decisions as described later in this section. Here, we describe the event when the requested variant of the video exists in the cache as *exact hit*, and the event when the requested variant does not exist in the cache but the transcodable version does exist as *soft hit*. Video transcoding, i.e., compressing a higher bitrate video to a lower bitrate variant, can be done by various techniques [140]. Among those, compressed domain based approaches, such as

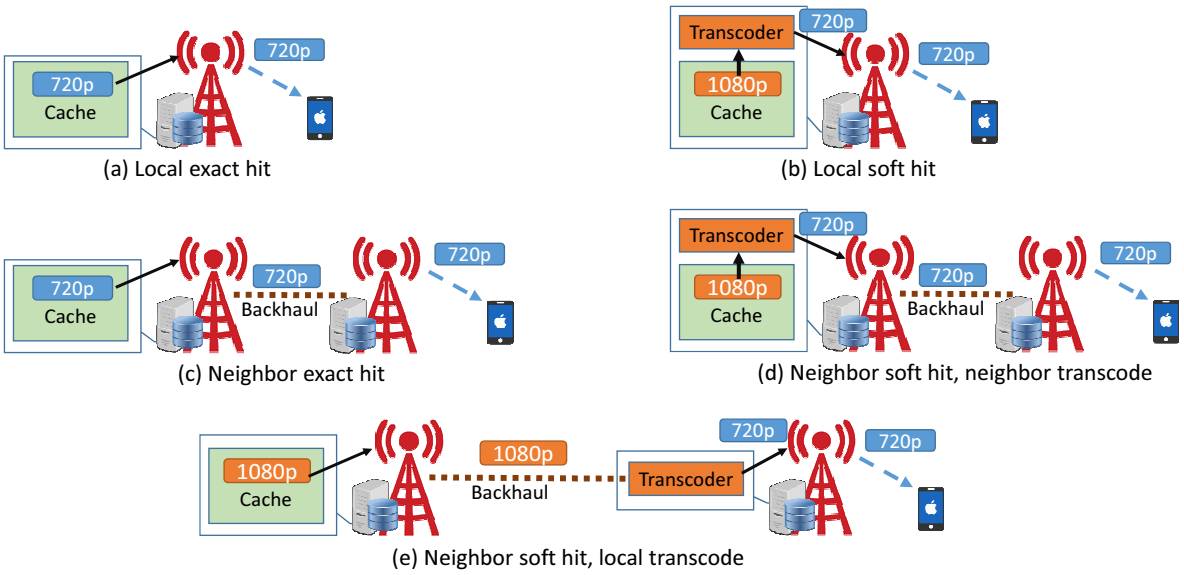



Figure 4.2: Illustration of possible (exclusive) events that happen when a user request for a video. (a) The video is obtained from cache of the home BS; (b) a higher bitrate version of the video from cache of the home BS is transrated to the desired bitrate version and deliver to the user; (c) the video is retrieved from cache of a neighboring BS or from the origin content server; (d) a higher bitrate version of the video from cache of a neighboring BS is transrated using the co-located transcoder and is then transfered to the home BS; (e) similar to (d) but the transcoding is done at the home BS's transcoder.

bitrate reduction and spatial resolution reduction, are the most favorable [158]. In general, video transcoding is a computation-intensive task. The cost of a transcoding task can be regarded as the CPU usage on the MEC cache server.

### 4.3.2 Settings

We consider a MEC network of $K$ cache servers, denoted as $\mathcal{K} = \{1, 2, ...K\}$. Each cache server is attached to a BS in the cellular RAN that spans $K$ cells. Additionally, $k = 0$ denotes the origin content server. The MEC servers are connected to each other via backhaul mesh network. The collection of videos available for download is indexed as $\mathcal{T} = \{1, 2, ...V\}$. Without loss of generality, we consider that all videos have the same length (playtime duration) and each has $L$ different bitrate variants. Hence, the size of each video variant $l$, denoted as $r_l$ [bytes], is proportional to its bitrate. The set of variants of video $v$ is indexed by an ordered list $\{v_1, v_2, ...v_L\}$ of increasing bitrates. Hence, the set of all video variants that a user can request is $\mathcal{V} = \{v_l \,|v \in \mathcal{T}, l = 1, ...L\}$. In the subsequent analysis, unless

otherwise stated, we will refer to video and video variant interchangeably. We consider that video $v_l$ can be transcoded from video $v_h$ if $l \leq h$ and the cost (CPU usage) of transcoding $v_h$ to $v_l$ is denoted as $p_{hl}$, $\forall v \in \mathcal{T}$ and $l, h = 1, ...L$. As considered in [141], we assume that $p_{hl}$ is proportional to $r_l$.

We consider that each user only connects to and receives data from the nearest BS (in terms of signal strength), which is later referred to as the user's *home BS*. Further extension to the system employing Coordinated Multi-Point transmission (CoMP), where each user can be served by multiple BSs, is a subject for future investigation.

To model the cache placement decision, we define the variables $c_j^{v_l} \in \{0,1\}, j \in \mathcal{K}, v_l \in \mathcal{V}$, in which $c_j^{v_l} = 1$ if $v_l$ is cached at server $j$ and $c_j^{v_l} = 0$ otherwise. Accordingly, the cache placement set is denoted by $\mathcal{C} = \left\{ c_j^{v_l} \middle| c_j^{v_l} = 1, j \in \mathcal{K}, v_l \in \mathcal{V} \right\}$. To account for the limited cache storage at each MEC server, we assume that each server is provisioned with a storage capacity of $M_j$ [bytes]. The cache storage capacity constraint at each server $j \in \mathcal{K}$ can then be expressed as,

$$\sum_{v_l \in \mathcal{V}} r_l c_j^{v_l} \leq M_j, \forall j \in \mathcal{K}. \tag{1}$$

To model the possible events that happen when a request for video $v_l$ arriving at server $j \in \mathcal{K}$, we introduce the binary variables $\left\{ x_j^{v_l}, y_j^{v_l}, z_{jk}^{vl}, t_{jk}^{v_l}, w_{jk}^{v_l} \right\} \in \{0,1\}$, which are explained as follows.

(a) $x_j^{v_l} = 1$ indicates that $v_l$ can be served directly from cache of BS $j$ (as illustrated in Fig. 4.2(a)); and $x_j^{v_l} = 0$ otherwise.

(b) $y_j^{v_l} = 1$ when $v_l$ is retrieved from cache at BS $j$ after being transcoded from a higher bitrate variant (as illustrated in Fig. 4.2(b)); and $y_j^{v_l} = 0$ otherwise.

(c) $z_{jk}^{v_l} = 1$ if $v_l$ is retrieved from cache of BS $k \neq j, k \in \mathcal{K} \cup \{0\}$ (including the remote server, as illustrated in Fig. 4.2(c)); and $z_{jk}^{v_l} = 0$ otherwise.

(d) $t_{jk}^{v_l} = 1$ when $v_l$ is obtained by transcoding a higher bitrate variant from cache of BS $k \neq j, k \in \mathcal{K}$ and the transcoding is performed at BS $k$ (as illustrated in Fig. 4.2(d)); and $t_{jk}^{v_l} = 0$ otherwise.

(e) $w_{jk}^{v_l} = 1$ when $v_l$ is obtained by transcoding a higher bitrate variant from cache of BS $k \neq j, k \in \mathcal{K}$ and the transcoding is performed at BS $j$ (as illustrated in Fig. 4.2(e)); and $w_{jk}^{v_l} = 0$ otherwise.

When a specific video quality is requested, the corresponding variant will be served following one of the event described above. To ensure there is one delivery for each request, we impose the following constraint ($\forall j \in \mathcal{K}, v_l \in \mathcal{V}$),

$$x_j^{v_l} + y_j^{v_l} + \sum_{k \neq j, k \in \mathcal{K}} \left( z_{jk}^{v_l} + t_{jk}^{v_l} + w_{jk}^{v_l} \right) + z_{j0}^{v_l} = 1. \tag{2}$$

### 4.3.3 Content Access Delay Cost

Let $d_{jk}$ denote the delay incurred when the $j$th cache server retrieves a video from the $k$th cache server, and let $d_{j0}$ denote the delay incurred when the $j$th cache server retrieves a video from the origin content server in the Internet. This delay cost model incorporates both the users' QoE via content access delay and the backhaul network cost via backhaul bandwidth consumption. In the best scenario, a user would like to receive its requested video from the local BS, which results in the lowest cost. Similar to [110], we consider this reference cost to be zero, i.e., $d_{jj} = 0, \forall j \in \mathcal{K}$. If we associate a cost between any two directly connected BSs, then for any two BSs $j$ and $k$ we can calculate $d_{jk}$ using the minimum cost path between $j$ and $k$. In practice, $d_{j0}$ is usually much greater than $d_{jk}$ as the backhaul link connecting a BS to the origin content server is of many-fold further than the backhaul links between the BSs. In the literature (cf. [42, 94, 95]), it is typically considered that $d_{j0} \gg d_{jk}, \forall j, k \in \mathcal{K}$. This makes it cost-effective to retrieve content from the in-network caches whenever possible rather than downloading them from the remote server. Nevertheless, the assumption that $d_{j0} \gg d_{jk}$ is not necessarily required for our analysis and proposed solutions in this work.

The incurred delay cost for video $v_l$ requested at BS $j$, $\forall j \in \mathcal{K}, v_l \in \mathcal{V}$, is calculated as,

$$D_j\left(v_l\right) = d_{j0} z_{j0}^{v_l} + \sum_{k \neq j, k \in \mathcal{K}} d_{jk} \left( z_{jk}^{v_l} + t_{jk}^{v_l} + w_{jk}^{v_l} \right). \tag{3}$$

The content access delay cost reflects the initial delay that the users have to wait before starting to play the requested videos. Reducing the content access delay cost (by retrieving content from shorter paths) directly translates into the decrease in backhaul usage. Therefore, it is very important to minimize the content average access delay cost of all video requests, which contribute to the improvement in users' QoE as well as network cost reduction.

### 4.3.4 Problem Formulation

To design the envisioned joint collaborative caching and processing in a MEC network, we now formulate the optimization problem that aims at minimizing the total delay cost of all the video requests. In particular, given the available resources (cache storage and processing capability), the objective is to jointly determine (i) a *cache placement policy* $\mathcal{C}$ and (ii) a *video request scheduling policy* $\mathcal{R} \triangleq \left\{x_j^{v_l}, y_j^{v_l}, z_{jk}^{v_l}, t_{jk}^{v_l}, w_{jk}^{v_l}\right\}$ in order to minimize the total content access delay cost, calculated as,

$$\Omega = \sum_{j \in \mathcal{K}} \sum_{v_l \in \mathcal{N}_j} D_j\left(v_l\right), \tag{4}$$

where $\mathcal{N}_j$ is the set of video request arrivals at BS $j$ over the time period under study. To take the user dynamics into account, we denote $\mathcal{N}_j^t$ as the set of video request being served at BS $j$ at every time $t$ whenever there is a new request arrival such that $\bigcup_t \mathcal{N}_j^t = \mathcal{N}_j$. Hence, the optimal solutions $\{\mathcal{C}, \mathcal{R}\}$ that minimize the total delay cost $\Omega$ can be determined by solving the static joint collaborative caching and processing problem (denoted as $\mathcal{J}^t$ in (5)) at every time $t$.

The constraints in problem (5b) can be explained as follows: constraints (5b) and (5c) ensure availability of the exact video variants; constraints (5d), (5e), and (5f) ensure the availability of the higher bitrate variants for transcoding; constraint (5g) ensures that there is one delivery for each request; constraint (5h) ensures the cache storage capacity; and constraint (5i) ensures the availability of processing resource (in terms of encoded bits that can be processed per second) for transcoding at each cache server.

$$(\mathcal{J}^t): \min_{\mathcal{C},\mathcal{R}} \sum_{j\in\mathcal{K}} \sum_{v_l\in\mathcal{N}_j^t} D_j\left(v_l\right), \tag{5a}$$

$$\text{s.t. } x_j^{v_l} \le c_j^{v_l}, \ \ \forall j \in \mathcal{K}, v_l \in \mathcal{V}, \tag{5b}$$

$$z_{jk}^{v_l} \le c_k^{v_l}, \ \ \forall j,k \in \mathcal{K}, \ v_l \in \mathcal{V}, \tag{5c}$$

$$y_j^{v_l} \le \min\left(1, \sum_{m=l+1}^{L} c_j^{v_m}\right), \ \ \forall j \in \mathcal{K}, v_l \in \mathcal{V}, \tag{5d}$$

$$t_{jk}^{v_l} \le \min\left(1, \sum_{m=l+1}^{L} c_k^{v_m}\right), \ \ \forall j \in \mathcal{K}, v_l \in \mathcal{V}, \tag{5e}$$

$$w_{jk}^{v_l} \le \min\left(1, \sum_{m=l+1}^{L} c_k^{v_m}\right), \ \ \forall j,k \in \mathcal{K}, v_l \in \mathcal{V}, \tag{5f}$$

$$x_j^{v_l} + y_j^{v_l} + \sum_{k\neq j, k\in\mathcal{K}} \left(z_{jk}^{v_l} + t_{jk}^{v_l} + w_{jk}^{v_l}\right) + z_{j0}^{v_l} = 1, \forall j \in \mathcal{K}, \tag{5g}$$

$$\sum_{v_l\in\mathcal{V}} r_l c_j^{v_l} \le M_j, \forall j \in \mathcal{K}, \tag{5h}$$

$$\sum_{v_l\in\mathcal{N}_j^t} p_l \left(y_j^{v_l} + \sum_{k\neq j, k\in\mathcal{K}} w_{jk}^{v_l}\right) + \sum_{k\neq j, k\in\mathcal{K}} \sum_{v_l\in\mathcal{N}_k^t} p_l t_{kj}^{v_l} \le P_j, \forall j \in \mathcal{K}, \tag{5i}$$

$$c_j^{v_l}, x_j^{v_l}, y_j^{v_l}, z_{jk}^{v_l}, t_{jk}^{v_l}, w_{jk}^{v_l} \in \{0,1\}, \ \ \forall j \in \mathcal{K}, v_l \in \mathcal{V}. \tag{5j}$$

Problem $\mathcal{J}^t$ in (5) is an ILP and is NP-complete, which can be shown by reduction from a multiple knapsack problem [161]. Thus, solving this problem to optimality in polynomial time is extremely challenging. Furthermore, while solving problem $\mathcal{J}^t$ for every time $t$ when there is a new request arrival will minimize the total backhaul network cost, this approach faces two critical challenges. First, optimizing the request scheduling at every time $t$ might result in re-directing on-going requests and thus wasting buffer data. Second, the *a-priori* information about the request arrival sets $\mathcal{N}_j^t$'s is rarely available in practice, or can only be estimated probabilistically.

To overcome the aforementioned drawbacks, our goal in this work is to design low-complexity algorithms that are easily implementable yet provide competitive performance for the considered objective function $\Omega$. Specifically, we follow the "divide and conquer" approach to decompose the original problem of solving a series of problems $\mathcal{J}^t$ into a *cache*

*placement problem* and a *request-scheduling problem*. We address the cache placement problem in two scenarios: (i) when content popularity is not available, the reactive LRU cache placement policy is employed; and (ii) when content popularity is available, we propose a novel ABR-aware Proactive Cache Placement (APCP) algorithm using the relation with the problem of maximizing a monotone submodular set function. In addition, *to preserve the interplay between the cache placement decision and the request scheduling decision, we introduce a scaling factor in the APCP algorithm that is adaptive to the processing capacity constraint at the BSs*. We show that the proposed APCP algorithm achieves an approximation performance guarantee. Furthermore, we propose a low-complexity online algorithm for the request scheduling problem, referred to as OnRS, using the relation with the online knapsack problem. We argue that no non-trivial competitive algorithm existed for the request scheduling problem in the general case and show via numerical simulations that our proposed OnRS algorithm yields performance that is very close to that of the optimal algorithm using global optimization solver. Specific details on the derivation of our proposed algorithms are presented in the following section.

## 4.4 Proposed Efficient Algorithms

In this section, we design low-complexity efficient algorithms to the cache placement problem and the request scheduling problem in order to minimize the total backhaul network cost $\Omega$.

### 4.4.1 Adaptive Bitrate Cache Placement

We consider both scenarios when the video popularity, i.e., the probability that each specific video variant is requested at each BS, is known and is not known.

**(a) Cache Placement with Unknown Video Popularity**

In this case, one has to make cache placement decision reactively following each cache miss, i.e., the requested video variant is neither available in the caches nor transcodable from other videos in the caches. The reactive cache placement policy has to be simple enough in order to quickly response to the large number of requests arriving to the system. As such,

we use the widely adopted LRU cache placement policy (LRU) [132, 141, 158]. Specifically, following each cache miss, the LRU policy fetches the requested video from the neighboring caches or the origin content server. It then saves the retrieved video file in the cache and if there is not enough space, the entries that have been least recently used are evicted to free up space for the newly added file.

It should be noted that, depending on the available processing resource at the MEC servers, one might choose to modify the pure LRU policy to take advantage of video transcoding. For example, if $P_j$ is very large at some server $j$, it is beneficial (in terms of cache storage efficiency) for this server to only cache the highest bitrate variant of each video. Whereas, if $P_j$ is small, multiple variants of popular videos should be stored to avoid processing-resource shortage for transcoding. Nevertheless, without the video popularity information, designing a reactive cache replacement algorithm that optimally adapts to the processing capacity at the MEC servers is a very challenging task that deserves a separate work.

**(b) Cache Placement with Known Video Popularity**

In MEC-enabled wireless systems, by leveraging sophisticated machine-learning and data-mining algorithms that can be implemented at the MEC servers, it is highly expected that estimation of the content popularity in each cell can be achieved. Such methods would involve analyzing data from popular websites, newspapers, and social networks to determine, around a specific BS, what kinds of contents people like, search for, and what the consumer profiles of these people are [121, 122]. By exploiting the availability of video popularity, we design a low-complexity APCP algorithm that determines the provisioning of cache contents at each server, which can be done during off-peak-traffic hours.

*Transcodability Factor and Request Scheduling Abstraction:* While the APCP algorithm does take into account the request scheduling decisions that happen at later times when the requests arrive, we aim at making the algorithm adaptive to the processing capacity constraints at the BSs by introducing the *transcodability factor* as follows. Firstly, denote the content popularity distribution at each BS $j$ as,

$$\mathcal{F}_j \;=\; \left\{ f_j^{v_l} \,\middle|\, v_l \in \mathcal{V}, \sum\nolimits_{v_l \in \mathcal{V}} f_j^{v_l} = 1 \right\},$$

in which $f_j^{v_l}$ is the probability that a request sent to BS $j$ is for video $v_l$. We consider that the video requests arrive one-by-one at each BS $j$ with average arrival rate of $\lambda_j$ [reqs/min]. Hence, the expected maximum processing load at BS $j$ due to transcoding, for any arbitrary cache placement policy, can be calculated as,

$$\bar{P}_j^{\max} = \lambda_j \sum\nolimits_{v \in \mathcal{V}} \sum\nolimits_{l=1}^{L-1} f_j^{v_l} p_l. \tag{6}$$

We define the *transcodability factor* at each BS $j$ to reflect the ratio between processing capacity at BS $j$ and the expected maximum processing load required, given by,

$$\tau_j \triangleq P_j / \bar{P}_j^{\max}. \tag{7}$$

As we are addressing the cache placement problem, the request scheduling variables in (5) are abstracted using the new variables $\alpha_{jk}^{v_l} \geq 0$, which indicates the availability of video $v_l$ at BS $k$ to serve request from BS $j$, such that,

$$\alpha_{jk}^{v_l} \triangleq \begin{cases} x_j^{v_l} + y_j^{v_l} & \text{if } k = j, \\ z_{jk}^{v_l} + t_{jk}^{v_l} + w_{jk}^{v_l} & \text{if } k \in \mathcal{K}, k \neq j, \\ z_{j0}^{v_l} & \text{if } k = 0. \end{cases} \tag{8}$$

From (8), we can recast the constraints in (5b)–(5f) w.r.t $\alpha_{jk}^{v_l}$ while incorporating the transcodability factor in the constraint as,

$$\alpha_{jk}^{v_l} \leq \min\left(1, c_k^{v_l} + \tau_k \sum\nolimits_{m=l+1}^{L} c_k^{v_m}\right), \forall j, k \in \mathcal{K}, v_l \in \mathcal{V}. \tag{9}$$

From the constraint above, we can see that when $\tau_j = 0$, BS $j$ does not have transcoding capability; and when $\tau_j \geq 1$, BS $j$ has infinite transcoding capacity. Let us denote $\boldsymbol{\alpha}_j^{v_l} = \left[\alpha_{j1}^{v_l}, ..., \alpha_{jK}^{v_l}\right]$. In addition, due to the constraint in (9), $\alpha_{jk}^{v_l}$ might take partial value when $k \neq 0$. Therefore, we recast constraint (5g) as,

$$\alpha_{j0}^{v_l} + \sum\nolimits_{k \in \mathcal{K}} \left\| \alpha_{jk}^{v_l} \right\|_0 = 1, \forall j \in \mathcal{K}, v_l \in \mathcal{V}, \tag{10}$$

where $\alpha_{j0}^{v_l} \in \{0, 1\}$ and $\|\cdot\|_0$ is the $l0$-norm.

*Cache Placement for Average Delay Cost Minimization:* Given the information about $\mathcal{F}_j$, we can calculate the average delay cost associated with requests for video $v_l$ from BS $j$ from (3) and (8) as,

$$\bar{D}_j\left(v_l\right) = f_j^{v_l} \sum_{k \in \mathcal{K} \cup \{0\}} d_{jk} \alpha_{jk}^{v_l}. \tag{11}$$

The ABR-aware proactive cache placement problem that aims at minimizing the expected content access delay cost can now be reduced from problem $\mathcal{J}^t$ as follows,

$$\min_{\mathcal{C}, \alpha_{jk}^{v_l}} \sum_{j \in \mathcal{K}} \sum_{v_l \in \mathcal{V}} \bar{D}_j\left(v_l\right), \tag{12a}$$

$$\text{s.t. } \alpha_{jk}^{v_l} \leq \min\left(1, c_k^{v_l} + \tau_k \sum_{m=l+1}^{L} c_k^{v_m}\right), \forall j, k \in \mathcal{K}, v_l \in \mathcal{V}, \tag{12b}$$

$$\alpha_{j0}^{v_l} + \sum_{k \in \mathcal{K}} \left\| \alpha_{jk}^{v_l} \right\|_0 = 1, \forall j \in \mathcal{K}, v_l \in \mathcal{V}, \tag{12c}$$

$$\alpha_{j0}^{v_l} \in \{0, 1\}, \forall j \in \mathcal{K}, v_l \in \mathcal{V}, \tag{12d}$$

$$\alpha_{jk}^{v_l} \geq 0, \forall j, k \in \mathcal{K}, k \neq j, v_l \in \mathcal{V}, \tag{12e}$$

$$\sum_{v_l \in \mathcal{V}} r_l c_j^{v_l} \leq M_j, \forall j \in \mathcal{K}. \tag{12f}$$

Problem (12) contains a special case of a traditional cooperative cache placement problem where each video only has one bitrate variant, i.e., $L = 1$, and different videos cannot be transcoded from each other. Such problem has been shown to be NP-hard using reduction from a set cover problem as in [42, 95]. Thus, *problem* (12) *is NP-hard.* To overcome the intractability of problem (12), we transform it into the problem of maximizing a monotone submodular set function in order to utilize the efficient techniques for this problem class. To proceed, we rewrite $\bar{D}_j\left(v_l\right)$ in (11) using (12c) as follows,

$$\bar{D}_j\left(v_l\right) = f_j^{v_l} \left[ \sum_{k \in \mathcal{K}} d_{jk} \alpha_{jk}^{v_l} + d_{j0} \left( 1 - \sum_{k \in \mathcal{K}} \left\| \alpha_{jk}^{v_l} \right\|_0 \right) \right] \tag{13a}$$

$$= f_j^{v_l} d_{j0} - f_j^{v_l} \sum_{k \in \mathcal{K}} \left( d_{j0} \left\| \alpha_{jk}^{v_l} \right\|_0 - d_{jk} \alpha_{jk}^{v_l} \right) \tag{13b}$$

$$\leq f_j^{v_l} d_{j0} - f_j^{v_l} \sum_{k \in \mathcal{K}} (d_{j0} - d_{jk}) \alpha_{jk}^{v_l}. \tag{13c}$$

The transition from (13b) to (13c) stems from the fact that $0 \leq \alpha_{jk}^{v_l} \leq \left\|\alpha_{jk}^{v_l}\right\|_0, \forall j,k \in \mathcal{K}$. It can be seen on the right hand side (RHS) of (13c) that the first term is a constant while the second term depends on the cache placement set $\mathcal{C}$ and it can be interpreted as the average backhaul network cost saving. Specifically, minimizing $\bar{D}_j\left(v_l\right)$ is equivalent to maximizing $\sum\limits_{k\in\mathcal{K}}\left(d_{j0}-d_{jk}\right)\alpha_{jk}^{v_l}$. Accordingly, we can recast problem (12) as an equivalent problem of maximizing the average content access delay cost saving, given by,

$$\max_{\mathcal{C}} \sum_{j\in\mathcal{K}}\sum_{v_l\in\mathcal{V}} f_j^{v_l} S_j^{v_l}\left(\mathcal{C}\right), \tag{14a}$$

$$\text{s.t.} \sum_{v_l\in\mathcal{V}} r_l c_j^{v_l} \leq M_j, \forall j \in \mathcal{K}, \tag{14b}$$

in which,

$$S_j^{v_l}\left(\mathcal{C}\right) \triangleq \max_{k,\alpha_{jk}^{v_l}} \left(d_{j0}-d_{jk}\right)\alpha_{jk}^{v_l} \tag{15a}$$

$$\text{s.t. } (12\text{b}),(12\text{c}),(12\text{d}),(12\text{e}). \tag{15b}$$

Note that for given $\mathcal{C}$, it is trivial to solve the simple Linear Program (LP) in (15) to obtain $S_j^{v_l}\left(\mathcal{C}\right)$. We exploit the structure of problem (14) via the following key lemma.

**Lemma 7.** *The objective function in problem* (14) *is monotone submodular over the cache placement ground set defined by* $\mathcal{G} = \left\{c_j^{v_l} \,|j \in \mathcal{K}, v_l \in \mathcal{V}\right\}$.

*Proof.* Since sum of monotone submodular functions is monotone submodular, it is enough to prove that for each BS $j$ and file $v_l$, the set function $S_j^{v_l}\left(\cdot\right)$ is monotone submodular over $\mathcal{G}$. The monotonicity and submodularity of $S_j^{v_l}\left(\cdot\right)$ are proven sequentially as follows.

*Monotonicity*. For a given cache placement set $\mathcal{C} \subset \mathcal{G}$ and an arbitrary new placement $c_i^{u_n} \in \mathcal{G}\backslash\mathcal{C}$, we denote $\mathcal{C}_{i,u_n} = \mathcal{C} \cup \{c_i^{u_n}\}$. According to (12b) we have,

$$\alpha_{jk}^{v_l}\left(\mathcal{C}\right) \leq \min\left(1, c_k^{v_l} + \tau_k \sum_{c_k^{v_m}\in\mathcal{C}, m=l+1}^{L} c_k^{v_m}\right), \tag{16}$$

$$\alpha_{jk}^{v_l}\left(\mathcal{C}_{i,u_n}\right) \leq \min\left(1, c_k^{v_l} + \tau_k \sum_{c_k^{v_m}\in\mathcal{C}_{i,u_n}, m=l+1}^{L} c_k^{v_m}\right). \tag{17}$$

Since $\mathcal{C} \subseteq \mathcal{C}_{i,u_n}$, the RHS of (17) is greater than or equal to the RHS of (16). Therefore, it can be easily verified in problem (15) that the feasible region of $\left\{\alpha_{jk}^{v_l}\left(\mathcal{C}_{i,u_n}\right)|k \in \mathcal{K}\right\}$ is greater than or equal to the feasible region of $\left\{\alpha_{jk}^{v_l}\left(\mathcal{C}\right)|k \in \mathcal{K}\right\}$. Hence, by solving (15), we will obtain $S_j^{v_l}\left(\mathcal{C}_{i,u_n}\right) \geq S_j^{v_l}\left(\mathcal{C}\right)$, which confirms the monotonicity of $S_j^{v_l}\left(\cdot\right)$.

*Submodularity*. Let us consider another cache placement decision $\mathcal{A}$ such that $\mathcal{A} \subseteq \mathcal{C}$, and denote $\mathcal{A}_{i,u_n} = \mathcal{A} \cup \{c_i^{u_n}\}$. The marginal value of adding $c_i^{u_n}$ to the cache placement set $\mathcal{C}$ and $\mathcal{A}$ can be calculated, respectively, as,

$$\Delta\mathcal{C}_i^{u_n} = S_j^{v_l}\left(\mathcal{C}_{i,u_n}\right) - S_j^{v_l}\left(\mathcal{C}\right) \tag{18}$$

$$\Delta\mathcal{A}_i^{u_n} = S_j^{v_l}\left(\mathcal{A}_{i,u_n}\right) - S_j^{v_l}\left(\mathcal{A}\right) \tag{19}$$

To prove that $S_j^{v_l}\left(\cdot\right)$ is submodular, we need to show that $\Delta\mathcal{A}_i^{u_n} \geq \Delta\mathcal{C}_i^{u_n}$, i.e., the marginal value decreases as the cache placement set growths. Notice that when $u \neq v$ or $u = v, n < l$ the request for video $v_l$ cannot be served using $u_n$. In this case, adding $c_i^{v_l}$ does not change the objective value, and thus, $\Delta\mathcal{A}_i^{u_n} = \Delta\mathcal{C}_i^{u_n} = 0$.

On the other hand, when $u = v$ and $n \geq l$, the placement $c_i^{u_n}$ can lead to an exact hit ($n = l$) or to a soft hit ($n > l$) for the request of $v_l$ at BS $j$. In this case, we have $\mathcal{A}_{i,u_n} = \mathcal{A}_{i,v_n}$ and $\mathcal{C}_{i,u_n} = \mathcal{C}_{i,v_n}$. Let us define,

$$\delta_{ji}^{v_l} \triangleq \begin{cases} 1 & \text{when } n = l, \\ \tau_i & \text{when } n > l. \end{cases} \tag{20}$$

Then, from (12b) and (15a), it is straightforward to verify that,

$$S_j^{v_l}\left(\mathcal{C}_{i,v_n}\right) = \max\left\{S_j^{v_l}\left(\mathcal{C}\right), \delta_{ji}^{v_l}\left(d_{j0} - d_{ji}\right)\right\}, \tag{21}$$

$$S_j^{v_l}\left(\mathcal{A}_{i,v_n}\right) = \max\left\{S_j^{v_l}\left(\mathcal{A}\right), \delta_{ji}^{v_l}\left(d_{j0} - d_{ji}\right)\right\}. \tag{22}$$

Additionally, since $S_j^{v_l}\left(\cdot\right)$ is monotone and $\mathcal{A} \subseteq \mathcal{C}$, it follows that $S_j^{v_l}\left(\mathcal{A}\right) \leq S_j^{v_l}\left(\mathcal{C}\right)$. We now distinguish three cases below,

(i) $\delta_{ji}^{v_l}\left(d_{j0} - d_{jk}\right) \leq S_j^{v_l}\left(\mathcal{A}\right)$: We have $S_j^{v_l}\left(\mathcal{C}_{i,v_n}\right) = S_j^{v_l}\left(\mathcal{C}\right)$ and $S_j^{v_l}\left(\mathcal{A}_{i,v_n}\right) = S_j^{v_l}\left(\mathcal{A}\right)$. Thus, $\Delta\mathcal{A}_n^{u_m} = \Delta\mathcal{C}_n^{u_m} = 0$.

(ii) $S_j^{v_l}(\mathcal{A}) \leq \delta_{ji}^{v_l}(d_{j0} - d_{jk}) \leq S_j^{v_l}(\mathcal{C})$: In this case, $S_j^{v_l}(\mathcal{C}_{i,v_n}) = S_j^{v_l}(\mathcal{C})$ and $S_j^{v_l}(\mathcal{A}_{i,v_n}) = \delta_{ji}^{v_l}(d_{j0} - d_{jk})$. It follows that $\Delta\mathcal{C}_n^{u_m} = 0 \leq \Delta\mathcal{A}_n^{u_m}$.

(iii) $S_j^{v_l}(\mathcal{C}) < \delta_{ji}^{v_l}(d_{j0} - d_{jk})$: We have $S_j^{v_l}(\mathcal{C}_{i,v_n}) = S_j^{v_l}(\mathcal{A}_{i,v_n}) = \delta_{ji}^{v_l}(d_{j0} - d_{jk})$. It follows from (18) and (19) that $\Delta\mathcal{C}_n^{u_m} \leq \Delta\mathcal{A}_n^{u_m}$.

In summary, we always have $\Delta\mathcal{C}_n^{u_m} \leq \Delta\mathcal{A}_n^{u_m}$, which confirms the submodularity of $S_j^{v_l}(\cdot)$. □

Lemma 7 provides valuable results as it shows the relationship between the cache placement problem in (14) with the problem of maximizing a monotone submodular set function subject to linear packing constraints. This allows us to exploit a wide range of efficient algorithms that have been proposed to deal with this problem, in which a greedy algorithm is preferable due to its low complexity, ease of implementation, and provable performance guarantee [162]. Let $\Gamma(\mathcal{C})$ denote the value of the objective function in (14a) for a given $\mathcal{C}$. The pseudo code for the greedy-based APCP algorithm for the cache placement problem is provided in Algorithm 7.

**Algorithm 7** ABR-Aware Proactive Cache Placement(APCP)

Initialize: $\mathcal{V}=\{v_l \,|v = 1, ...V; l = 1, ...L\}$; $\mathcal{G}=\left\{c_j^{v_l} \,|j \in \mathcal{K}, v_l \in \mathcal{V}\right\}$; $\mathcal{C} = \emptyset$; $\rho \in \mathbb{R}_+$
Set $\mu_j := 1/M_j, \forall j \in \mathcal{K}$
**while** $\sum_{j\in\mathcal{K}} M_j\mu_j \leq \rho$ and $\mathcal{C} \neq \mathcal{G}$ **do**
    $c_i^{u_n} = \arg\min_{c_j^{v_l}\in\mathcal{G}\setminus\mathcal{C}} \frac{r_n}{\Gamma(\mathcal{C}\cup\{c_j^{v_l}\})}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_i^{u_n}\}$
    $\mu_i \leftarrow \mu_i \rho^{c_i^{u_n} r_n/M_i}$
**if** $\sum_{v_l\in\mathcal{V}} r_l c_j^{v_l} \leq M_j, \forall j \in \mathcal{K}$ **then**
    Return $\mathcal{C}$
**else if** $\Gamma(\mathcal{C}\setminus\{c_i^{u_n}\}) \geq f(\{c_i^{u_n}\})$ **then**
    Return $\mathcal{C}\setminus\{c_i^{u_n}\}$
**else**
    Return $\{c_i^{u_n}\}$

**Lemma 8.** *Algorithm 7 outputs a feasible solution to the cache placement problem in* (14).

*Proof.* Let $\mathcal{C}$ be the cache placement set obtained after the main loop terminates, i.e., at line 7. We argue that: (i) if $\mathcal{C}$ satisfies the first condition in line 3, it is a also a feasible solution, i.e., satisfying the condition in line 8 (*argument 1*); otherwise (ii) $\mathcal{C}$ became infeasible only at the last iteration of the loop in which element $c_i^{u_n}$ was selected (*argument 2*).

At the end of iteration $t$ of the algorithm, denote $\mathcal{C}_t$ as the obtained cache placement set and $\mu_{jt}$ as the updated value of $\mu_j$. Note that $\mathcal{C}_0 = \emptyset$ and $\mu_{j0} = 1/M_j, \forall j \in \mathcal{K}$. We have,

$$M_j \mu_{jt} = M_j \mu_{j0} \prod_{c_j^{v_l} \in \mathcal{C}_t} \rho^{c_j^{v_l} r_n / M_j} = \rho^{\sum_{c_j^{v_l} \in \mathcal{C}_t} c_j^{v_l} r_n / M_j}. \tag{23}$$

Suppose that the loop in Algorithm 7 terminates at iteration $t$. *Argument 1* is easy to see. Assuming $\sum_{j \in \mathcal{K}} M_j \mu_{jt} \leq \rho$, it is obvious that $M_j \mu_{jt} \leq \rho, \forall j \in \mathcal{K}$. Using (23), we get,

$$\sum\nolimits_{c_j^{v_l} \in \mathcal{C}_t} c_j^{v_l} r_n / M_j \leq 1, \tag{24}$$

which confirms the feasibility of $\mathcal{C}_t$.

To establish *argument 2*, let $c_i^{u_n}$ be the first element that induces a violation in some constraint. Specifically, suppose $c_i^{u_n}$ induces a violation in constraint $i$ at iteration $t$. Accordingly, $\mathcal{C}_t = \mathcal{C}_{t-1} \cup \{c_i^{u_n}\}$ and $\sum_{c_i^{v_l} \in \mathcal{C}_t} c_i^{v_l} r_l > M_i$. Therefore, from (23), we have,

$$M_i \mu_{it} = \rho^{\sum_{c_i^{v_l} \in \mathcal{C}_t} c_i^{v_l} r_l / M_i} > \rho. \tag{25}$$

This implies that $\sum_{j \in \mathcal{K}} M_j \mu_{jt} > \rho$; hence, by checking the stopping condition of the main loop (line 3), we conclude that the loop must have terminated immediately after iteration $t$. Thus, *argument 2* holds; therefore, the returned solution must be feasible as it is either $\mathcal{C}_t \backslash \{c_i^{u_n}\}$ or $\{c_i^{u_n}\}$. □

Notice that constraint (14b) presents a linear packing constraint whose width can be defined as $W \triangleq \min\{M_j / r_l \,|\, j \in \mathcal{K}, l \in \mathcal{L}\}$. The approximation guarantee of Algorithm 7 is established via the following lemma.

**Lemma 9.** *Algorithm 7 achieves* $\Omega\left(K^{-1/W}\right)$*-approximation by choosing the update factor* $\rho = K \times \exp(W)$.

*Proof.* The proof follows the proof of Lemma 2.4 in [162], which establishes a lower bound on the value of the objective function using its monotonic and submodular properties. This result also matches the best known approximation guarantee for the case that the objective function is linear, achievable using continuous relaxation and randomized rounding techniques [163]. □

**Remark 7**: (Complexity of Algorithm 7) Recall that the minimum size of a video variant is $r_1$; hence, each server $j$ can store at most $M_j/r_1$ video files. Therefore, the main loop in Algorithm 7 will go over at most $\sum_{j\in\mathcal{K}} M_j/r_1$ iterations. Each iteration involves calculating the marginal value of at most $V \times L$ elements in the ground set $\mathcal{G}$ that have not been included in the cache placement solution $\mathcal{C}$. Evaluating each marginal value takes $\mathcal{O}(\mathcal{K})$ time; hence, the total running time of the algorithm is $\mathcal{O}\left(KVL\sum_{j\in\mathcal{K}} M_j/r_1\right)$. □

### 4.4.2 Video Request Scheduling

In the previous subsection, we have presented two cache placement algorithms: a reactive solution when content popularity is not known (LRU algorithm); and a proactive solution when content popularity is known (Algorithm 7). In this section, we address the request-scheduling decision that determines *how* and *where* to get the requested video to the users upon each request arrival.

The request-scheduling problem at each time $t$, denoted as $\mathcal{Q}^t$, is formulated by a reduction from problem $\mathcal{J}^t$ in (5) for a given cache placement decision $\mathcal{C}$ as follows,

$$(\mathcal{Q}^t): \min_{\mathcal{R}} \sum_{j\in\mathcal{K}} \sum_{v_l\in\mathcal{N}_j^t} D_j(v_l) \tag{26a}$$

$$\text{s.t. } (5b)-(5g), (5i), \tag{26b}$$

$$x_j^{vl}, y_j^{vl}, z_{jk}^{vl}, t_{jk}^{vl}, w_{jk}^{vl} \in \{0,1\}, \forall j \in \mathcal{K}, v_l \in \mathcal{V}. \tag{26c}$$

#### (a) Optimal Request Scheduling

The optimal request scheduling approach would be to solve problem $\mathcal{Q}^t$ from scratch at every time $t$ when a new video request arrives. This way, the obtained request scheduling decision would be *optimal* at every time $t$, yielding optimal solution in the long run. As $\mathcal{Q}^t$

is an ILP, the optimal solution can be obtained using standard optimization solver, such as MOSEK [164]. We refer to the optimal request scheduling approach that solves $\mathcal{Q}^t$ at every time $t$ using the ILP optimal solver as OptRS. The drawback of the OptRS approach, however, is that it might cause re-directing the existing (on going) video requests whenever the optimal request scheduling solution is re-calculated, thus wasting the buffered data at the BSs. Additionally, the complexity of solving problem $\mathcal{Q}$ scales with the number of request arrivals and number of caching servers and thus it is highly impractical to re-solve this problem when there is a large number of request arrivals in a very short time.

**(b) Online Video Request Scheduling**

To overcome the impracticality of the optimal approach, we propose an online algorithm for the video request scheduling problem. Specifically, we consider the *online* version of problem $\mathcal{Q}^t$, in which the set of video requests $\mathcal{N}_j^t$'s is not known at the beginning and video requests arrive one at a time. At each instance $t$, the request scheduling algorithm must decide what to do with the new incoming request (e.g., accepting the request as an exact hit, soft hit, or retrieving the video from remote server), and the decision once made is irrevocable—requests once scheduled cannot be re-directed. To simplify the notation, we drop the subscript $t$ from this point on. The online algorithm will make video request scheduling decision immediately upon each video request arrival at one of the BSs.

Firstly, given the current set of videos $\mathcal{N}_j$ being served at the BS $j$, we can compute the current processing load (due to transcoding) at BS $j$ as,

$$U_j\left(\mathcal{N}_j\right) = \sum_{v_l \in \mathcal{N}_j} p_l \left( y_j^{v_l} + \sum_{k \in \mathcal{K} \setminus \{j\}} w_{jk}^{v_l} \right) + \sum_{k \in \mathcal{K} \setminus \{j\}} \sum_{v_l \in \mathcal{N}_k} p_l t_{kj}^{v_l}. \tag{27}$$

Furthermore, we define the *closest* (in terms of bitrate) transcodable variant of video $v_l$ at BS $j$ as $T\left(j, v_l\right) = v_h$, in which,

$$h = \arg\min_{m > l} c_j^{v_m} \quad \text{s.t.} \quad \mathrm{c}_j^{v_m} = 1. \tag{28}$$

For each video request $v_l$ arriving at BS $j \in \mathcal{K}$, we outline the online request scheduling decision as in Algorithm 8 on page 112. In particular, if $v_l$ cannot be directly retrieved

(line 2) or transcoded (lines 3, 4) from cache of BS $j$, the algorithm will search for $v_l$ or its transcodable variant from other neighboring caches. Lines 5, 6, and 7 find the exactly requested video $v_l$ from the neighboring caches and, if that exists, $v_l$ will be retrieved from the cache with lowest backhaul cost. Otherwise, a transcodable variant of $v_l$ will be searched from neighboring caches in line 9. If the transcodable version exists in the cache of BS $k$, the algorithm will select the cache server (either server $k$ or the requesting server $j$) with most available processing resource to perform transcoding. Finally, if $v_l$ cannot be satisfied by the cache system, it will be fetched from the origin content server (in line 17), which incurs the highest backhaul cost.

**Algorithm 8** Online Video Request Scheduling (OnRS)

1: For each video request $v_l$ arriving at BS $j \in \mathcal{K}$, proceed.
2: **if** $c_j^{v_l} = 1$ **then** stream $v_l$ from cache of BS $j$ to the user.
3: **else if** $T(j, v_l) \neq \emptyset$ and $P_j - U_j(\mathcal{N}) - p_l \geq 0$ **then**
4: Transcode $T(j, v_l)$ from cache of BS $j$ to $v_l$ and then stream it to the user.
5: **else if** $\sum_{k \in \mathcal{K} \setminus \{j\}} c_k^{v_l} \geq 1$ **then**
6: Find $f = \arg\min_{k \in \mathcal{K} \setminus \{j\}} d_{jk}$ s.t. $c_k^{v_l} = 1$
7: Retrieve $v_l$ from cache of BS $f$, send it to BS $j$, and then stream it to the user.
8: **else if** $\bigcup_{k \in \mathcal{K} \setminus \{j\}} T(k, v_l) \neq \emptyset$ **then**
9: **else if** $\tilde{\mathcal{K}} \triangleq \{k \in \mathcal{K} \setminus \{j\} \,|T(k, v_l) \neq \emptyset\} \neq \emptyset$ **then**
10: Calculate the computing-resource availability

$$Q_k(\mathcal{N}) = P_k - U_k(\mathcal{N}) - p_l, \forall k \in \tilde{\mathcal{K}} \cup \{j\} \tag{29}$$

11: Find $f = \arg\max_{k \in \tilde{\mathcal{K}} \cup \{j\}} Q_k(\mathcal{N})$ s.t. $Q_k(\mathcal{N}) \geq 0$
12: **if** $f = j$ **then**
13: Retrieve $T(k, v_l)$ from cache of BS $k \in \tilde{\mathcal{K}}$ with the least backhaul cost.
14: Transcode $T(f, v_l)$ to $v_l$ at BS $j$ and then stream it to the user.
15: **else if** $f \neq \emptyset$ **then**
16: Transcode $T(f, v_l)$ from cache of BS $f$ to $v_l$.
17: Send $v_l$ from BS $f$ to BS $j$ and then stream it to the user.
18: **else** Go to line 21.
19: **else**
20: Retrieve $v_l$ from the origin content server and then stream it to the user.

The online request scheduling problem can be seen as an instance of the Online Multiple-Choice Knapsack Problem (ON-MCKP) where the items are the video requests that require

transcoding, the weight of each item is the transcoding cost of the video and the value of each item is the backhaul network cost saving associated with the scheduling decision. In a general case, it has been shown that the ON-MCKP is NP-hard and there are no non-trivial competitive algorithms existed for this problem [165]. Due to this impossibility result, we will compare performance of Algorithm 8 with the optimal counter part—the OptRS method using global optimization solver—via numerical simulations. As we will see in Sect. 4.5, our proposed OnRS method in Algorithm 8 yields performance that is very close to that of the OptRS method.

**Remark 8**: (Complexity of Algorithm 8) The complexity of Algorithm 8 mainly comes from lines 6 and 10, which are simple sorting processes; thus, it scales linearly with the number of BSs, i.e., $\mathcal{O}(\mathcal{K})$. □

## 4.5 Performance Evaluation

In this section, we evaluate the performance of the proposed adaptive bitrate collaborative video caching and processing solutions under various cache sizes, processing capacities and video request arrival rates. We consider a MEC networks consisting of 3 MEC servers, each deployed on a BS of a cellular RAN. We assume the video library $\mathcal{V}$ that consists of $V = 1000$ unique videos, each having $L = 4$ bitrate variants. Like in [141], we set the relative bitrates of the four variants $(r_1, ..., r_4)$ to be $0.45, 0.55, 0.67$, and $0.82$ of the original video bitrate (2 Mbps), respectively. We assume that all video variants have equal length of 10 minutes. The popularity of the videos being requested at each BS is generated following a Zipf distribution with the skew parameter $\alpha = 0.6$, i.e., the probability that an incoming request is for the $i$-th most popular video is given as $q_i = \frac{1/i^{\alpha}}{\sum_{j=1}^{V} 1/j^{\alpha}}$.

In order to obtain a scenario where the same video can have different popularities at different locations, we randomly shuffle the distributions at different BSs. Video requests arrive one-by-one at each BS $j$ following a Poisson distribution with rate $\lambda_j$ [reqs/min]. Note that our solutions are applicable for any request arrival distribution. For each simulation, we randomly generate $10,000$ requests at each BS. The end-to-end latency of fetching video content from the local BS, from a neighboring BS, and from the origin content server are

Table 4.2: Summary of Competing Schemes for Video Caching and Processing

| ***Reactive* Caching and Processing** | | ***Proactive* Caching and Processing** | |
|---|---|---|---|
| **LRU-OnRS** | This scheme uses the reactive caching algorithm LRU (presented in Sect. 4.4.1(a)) and the proposed online request scheduling OnRS (Algorithm 8). This scheme was referred to as Online-JCCP in our previous work [39]. | **APCP-OnRS** | This scheme uses the proposed APCP algorithm (Algorithm 7) for cache placement and the OnRS algorithm (Algorithm 8) for request scheduling. |
| **LRU-OptRS** | This scheme uses the LRU caching algorithm and the optimal request scheduling algorithm OptRS presented in Sect. 4.4.2(a). | **APCP-OptRS** | This scheme uses the cache placement algorithm APCP and the optimal request scheduling algorithm OptRS. |
| **LRU-CachePro** | A joint caching and processing scheme without collaboration among the cache servers, as proposed in [141]. The LRU cache placement algorithm is used. | **APCP-CachePro** | This scheme is similar to LRU-CachePro, but it uses the APCP algorithm for cache placement. |
| **LRU-CoCache** | A collaborative caching scheme without transcoding, i.e., using Algorithm 8 with zero processing capacity at the BS, and the LRU cache placement policy is employed. | **APCP-CoCache** | This scheme is similar to LRU-CoCache, but it uses the APCP algorithm for cache placement. |

randomly assigned following the uniform distribution in the ranges $[5, 10]$(ms), $[20, 50]$(ms), and $[100, 200]$(ms), respectively [166]. The per-unit-data backhaul cost $d_{j0}$'s and $d_{jk}$'s are set equal to the corresponding delays. In terms of resources, we set the cache storage capacity relative to the total size of the video library, and the processing capacity is regard as the number of encoded bits that can be processed per second.

In our performance evaluation, we consider the following three important metrics: *(i) cache hit ratio*—the fraction of requests that can be satisfied either by retrieving from the cache or by transcoding; *(ii) average access delay* [ms]—average latency of the contents traveling from the caches or the origin server to the requesting user; *(iii) external backhaul traffic load* [TB]—the volume of data traffic going through the backhaul network due to users downloading videos from the origin server. In the simulation results, we compare performance of the competing schemes as summarized in Table 4.2, in which LRU-OnRS and APCP-OnRS are our proposed algorithms for the two scenarios: (1) *reactive* caching for unknown content popularity, and (2) *proactive* caching for known content popularity, respectively.

### 4.5.1 Impact of Cache Capacity

We compare performance of the considered schemes in terms of cache hit ratio, average access delay, and external backhaul traffic load at different relative cache capacities, given by the ratio between the cache size at each BS and the total size of the video library. The results are reported in Fig. 4.3(a, b, c) for the case of reactive caching and in Fig. 4.3(d, e, f) for the case of proactive caching, in which $P_j = 10$ Mbps and $\lambda_j = 8$ reqs/minute, $\forall$j $\in \mathcal{K}$. From the figures, we can see that increasing the cache size results in performance improvement for all schemes. Notice that our proposed schemes, LRU-OnRS and APCP-OnRS, always significantly outperform the baselines in the corresponding scenarios. At small and moderate cache sizes, the performance of LRU-OnRS and APCP-OnRS schemes are slightly lower than that of the optimal request scheduling schemes LRU-OptRS and APCP-OptRS, respectively. On the other hand, when the cache size is high (greater than 50% and 30% of the library size for reactive and proactive caching, respectively), the performance of LRU-OnRS and APCP-OnRS approaches that of LRU-OptRS and APCP-OptRS, respectively.

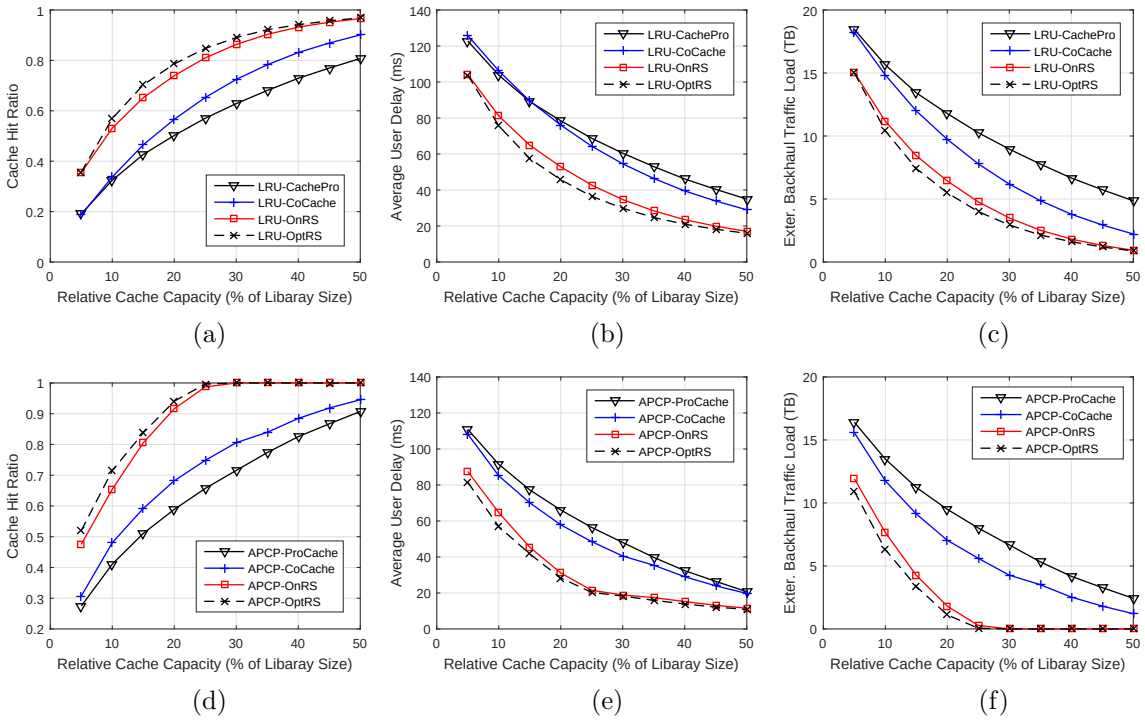



Figure 4.3: Performance comparison of different caching and processing schemes when increasing relative cache capacity at each server.

### 4.5.2 Impact of Processing Capacity

To evaluate the impact of processing capacity, we compare performance of the considered scheme at different processing capacities as in Fig. 4.4(a, b, c) for reactive caching, and in Fig. 4.4(d, e, f) for proactive caching, in which $M_j = 20\%$ [Library Size] and $\lambda_j = 8$ reqs/minute, $\forall$j $\in \mathcal{K}$. As the two schemes LRU-CoCache and APCP-CoCache do not involve transcoding, their performance does not change with processing capacity. On the other hand, the performance of the other schemes is observed as follows. When processing capacity at the MEC servers is small, increasing processing capacity always results in performance improvement to all schemes. Performance of the LRU-OnRS scheme approaches that of the LRU-OptRS scheme when processing capacity is high, whereas performance of the APCP-OnRS scheme approaches that of the APCP-OptRS scheme when processing capacity is low. Notice that the performance of LRU-OnRS and APCP-OnRS always significantly outperform that of the baseline schemes in the corresponding scenarios.

### 4.5.3 Impact of User Dynamics

User dynamics in each cell, i.e., the frequency that potential video-requesting users enter a cell and the duration that they stay in each cell, result in how frequently the cache server needs to make cache updates (in case of reactive caching) and request scheduling decisions. Hence, we want to study the impact of high user dynamics on the performance of our proposed caching and request scheduling schemes. Suppose video-requesting users arrive at each BS $j$ with average arrival rate $\lambda_j^u$ [users/min] and their average active time is $W$ [min]. Additionally, we assume the mean video request arrival rate per active user is $\lambda^v$. Thus, the video request arrival rate at each BS $j$ can be calculated as $\lambda_j = \lambda_j^u \lambda^v W$. Therefore, high user dynamics will result in high video request arrival rate, and vice versa.

In Fig. 4.5(a, b), we illustrate the cache hit ratio performance of LRU-OnRS scheme and APCP-OnRS scheme at different values of video request arrival rate and processing capacity, in which $M_j = 20\%[\text{Library Size}], \forall \text{j} \in \mathcal{K}$. It can be seen that, when the request arrival rate increases, the cache hit performance of both schemes decreases. On the other

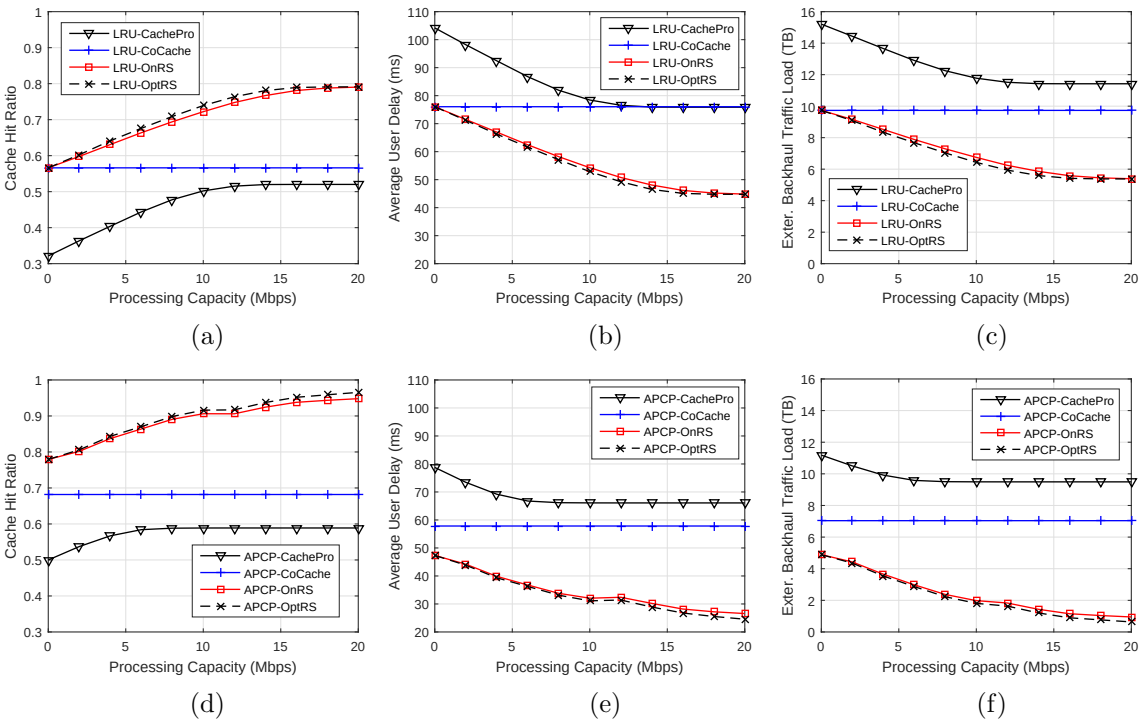
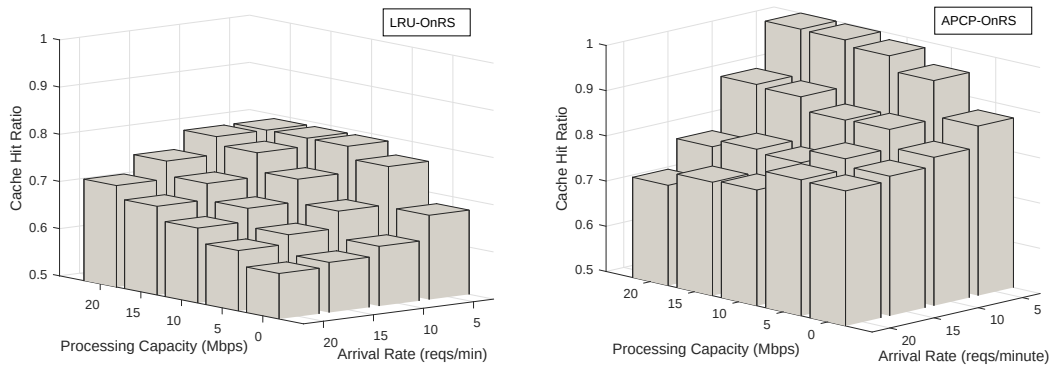



Figure 4.4: Performance comparison of different caching and processing schemes when increasing processing capacity at each server.

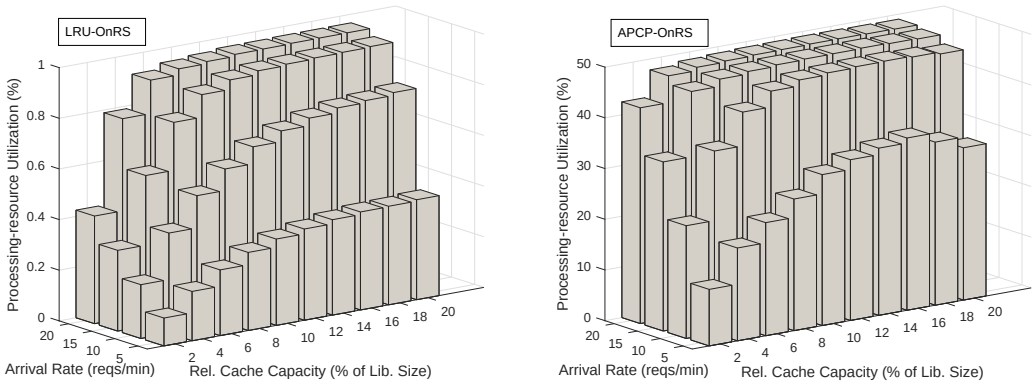



(a) (b)

Figure 4.5: Hit ratio performance of (a) LRU-OnRS and (b) APCP-OnRS.

hand, when the processing capacity increases, the cache hit performance of LRU-OnRS scheme increases while that of APCP-OnRS scheme decreases at high request arrival rate and increases at low request arrival rate. The behavior of the APCP-OnRS scheme can be explained as follows. The cache placement strategy in APCP-OnRS depends on the transcodability factor $\tau_j$ (defined in (7)) at each BS $j$, which increases with high processing capacity and decreases with high request arrival rate. Thus, when the processing capacity is dominating, we might get high value of $\tau_j$ which results in a cache placement strategy that prefers to cache less number of variants for each video and relies more on video transcoding. However, such adaptation might not be optimal at very high processing capacity and high request arrival rate regime, and thus it leads to degradation in cache hit ratio in such regime as shown in Fig. 4.5(b). In future work, it is worth investigating a proactive cache placement strategy that better utilizes the high processing capacity to improve cache hit performance.

In Fig. 4.6(a, b), we illustrate the processing resource utilization of our proposed LRU-OnRS and APCP-OnRS schemes versus different video request arrival rates and cache capacities, in which $P_j = 20$ Mbps, $\forall$j $\in \mathcal{K}$. We observe that the processing utilization of both schemes increases with arrival rate and moderate cache capacity; however, it decreases at high cache capacity. This can be explained as follows: when the cache capacity is high, the MEC servers can store a large number of video variants and thus there are fewer requests requiring transcoding.




(a) (b)

Figure 4.6: Average processing resource utilization at the cache servers using (a) LRU-OnRS and (b) APCP-OnRS.

## 4.6 Summary

In this chapter, we proposed the idea of deploying a collaborative caching and processing framework in a multi-cell Mobile-Edge Computing (MEC) networks, whereby the MEC servers attached to the Base Stations (BSs) can assist each other for both caching and transcoding of Adaptive Bitrate (ABR) videos. The problem of joint collaborative caching and processing is formulated as an Integer Linear Program (ILP) aiming at minimizing the total cost of retrieving video contents over backhaul links. Due to the NP-completeness of the problem and to the absence of the request arrival information in practice, we design low-complexity algorithms that are easily implementable and provide solutions with competitive performance. Specifically, we decompose the original problem into two subproblems, a cache placement problem and a request-scheduling problem. Our main contributions include a novel ABR-aware proactive cache placement algorithm with provable approximation performance when video popularity is available and a low-complexity online request scheduling algorithm that performs very closely to the optimal solution using optimization solver. Extensive simulation results showed that our proposed joint caching and processing framework provides significant performance improvement in terms of cache hit ratio, backhaul traffic load, and content access delay, over the traditional approaches.

# Chapter 5

# Joint Task Offloading and Resource Allocation for Multi-Server MEC

In this chapter, a MEC enabled multi-cell wireless network is considered where each BS is equipped with a MEC server that assists mobile users in executing computation-intensive tasks via task offloading. The problem of Joint Task Offloading and Resource Allocation (JTORA) is studied in order to maximize the users' task offloading gains, which is measured by a weighted sum of reductions in task completion time and energy consumption. The considered problem is formulated as a Mixed Integer Non-linear Program (MINLP) that involves jointly optimizing the task offloading decision, uplink transmission power of mobile users, and computing resource allocation at the MEC servers. Due to the NP-hardness of this problem, solving for optimal solution is difficult and impractical for a large-scale network. To overcome this drawback, we propose to decompose the original problem into (i) a Resource Allocation (RA) problem with fixed task offloading decision and (ii) a Task Offloading (TO) problem that optimizes the optimal-value function corresponding to the RA problem. We address the RA problem using convex and quasi-convex optimization techniques, and propose a novel heuristic algorithm to the TO problem that achieves a sub-optimal solution in polynomial time. Simulation results show that our algorithm performs closely to the optimal solution and that it significantly improves the users' offloading utility over traditional approaches.

## 5.1 Introduction

**Motivation:** With the emergence of MEC, the ability of resource-constrained mobile devices to offload computation tasks to the MEC servers is expected to support a myriad

of new services and applications such as augmented/virtual reality [167, 168], IoTs, autonomous vehicles, and image processing [169]. Example applications such as face detection and recognition for airport security and surveillance [170], video crowd-sourcing that reconstructs multi-view live videos of an event [171], or real-time healthcare data analytics [172] can highly benefit from the collaboration between mobile devices and MEC platform. In the former case, a central authority such as the Federal Bureau of Investigation (FBI) would extend their Amber alerts such that all available cell phones in the area where a missing child was last seen that opt-in to the alert would actively capture images. In the latter case, the MEC servers collect individual input videos (views) captured for the same event from multiple attendees and combine them into multi-view videos, allowing viewers to watch the event from various angles. In both applications, the user devices only provide input data (images, videos) and the computation-intensive tasks (face recognition, multi-view video construction) are then performed at the MEC servers.

Task offloading, however, incurs extra overheads in terms of delay and energy consumption due to the communication required between the devices and the MEC server in the uplink wireless channels. Additionally, in a system with a large number of offloading users, the finite computing resources at the MEC servers considerably affect the task execution delay [173]. Therefore, offloading decisions and performing resource allocation become a critical problem toward enabling efficient computation offloading. Previously, this problem has been partially addressed by optimizing either the offloading decision [173,174], communication resources [34,175], or computing resources [176,177]. Recently, Sardellitti et al. [36] addressed the joint allocation of radio and computing resources, while the authors in [178] considered the joint task offloading and resources optimization in a multi-user system. Both of these works, however, only concentrate on a system with a single MEC server.

**Our Vision:** Unlike the traditional approaches mentioned above, our objective is to design a holistic solution for joint task offloading and resource allocation in a multi-server MEC-assisted network so as to maximize the users' offloading gains. Specifically, we consider a multi-cell ultra-dense network where each BS is equipped with a MEC server to provide computation offloading services to the mobile users. The distributed deployment of

the MEC servers along with the densification of (small cell) BSs—as foreseen in the 5G standardization roadmap [179]—will pave the way for real proximity, ultra-low latency access to cloud functionalities. Additionally, the benefits brought by a multi-server MEC system over the single-server MEC (aka single-cloud) system are multi-fold: (i) firstly, as each MEC server may be overloaded when serving a large number of offloading users, one can release the burdens on that server by directing some users to offload to the neighboring servers from the nearby BSs, thus preventing the limited resources on each MEC server from becoming the bottle neck; (ii) secondly, each user can choose to offload its task to the BS with more favorable uplink channel condition, thus saving transmission energy consumption; (iii) finally, coordination of resource allocation to offload users across multiple neighboring BSs can help mitigate the effect of interference and resource contention among the users and hence, improve offloading gains when multiple users offload their tasks simultaneously.

The envisioned scheme requires global knowledge about the computation tasks at the users, the computational resources at the MEC servers, and the channel condition between users and BSs. The mechanism to gather these information can be done similarly as in the CoMP transmission systems [45] where the BSs exchange their channel information with each other using the logical X2 interface. Alternatively, in C-RAN [14] where all the BSs are connected to the same BBU via high-capacity backhaul links, the global system state can be easily accessible at the BBU. Hence, the proposed scheme can be implemented at a central entity which might be located at an aggregation point in the traditional RAN hierarchy [26] or at the BBU pool in C-RAN. Similar to the system that performs centralized joint transmission for interference management [45] and/or joint user scheduling [103], our proposed scheme comes with the cost of increased demand on backhaul due to additional signaling overhead. However, it is anticipated that this issue can be overcome when different BSs are connected together in the form of C-RAN.

**Challenges and Contributions:** To exploit in full the benefits of computation offloading in the considered multi-cell, multi-server MEC network, there are several key challenges that need to be addressed. Firstly, the radio resource allocation is much more challenging than the special cases studied in the literature (cf. [178]) due to the presence of inter-cell interference that introduces the coupling among the achievable data rate of different users,

which makes the problem nonconvex. Secondly, the complexity of the task-offloading decision is high as, for each user, one needs to decide not only whether it should offload the computation task but also which BS/server to offload the task to. Thirdly, the optimization model should take into account the inherent heterogeneity in terms of mobile devices' computing capabilities, computation task requirements, and availability of computing resources at different MEC servers. In this context, the main contributions of this work are summarized as follows.

- We model the offloading utility of each user as the weighted-sum of the improvement in task-completion time and device energy consumption; we formulate the problem of Joint Task Offloading and Resource Allocation (JTORA) as a Mixed Integer Nonlinear Program (MINLP) that jointly optimizes the task offloading decisions, users' uplink transmit power, and computing resource allocation to offloaded users at the MEC servers, so as to maximize the system offloading utility.
- Given the NP-hardness of the JTORA problem, we propose to decompose the problem into (i) a Resource Allocation (RA) problem with fixed task offloading decision and (ii) a Task Offloading (TO) problem that optimizes the optimal-value function corresponding to the RA problem.
- We further show that the RA problem can be decoupled into two independent problems, namely the Uplink Power Allocation (UPA) problem and the Computing Resource Allocation (CRA) problem; the resulting UPA and CRA problems are addressed using quasi-convex and convex optimization techniques, respectively.
- We propose a novel low-complexity heuristic algorithm to tackle the TO problem and show that it achieves a suboptimal solution in polynomial time.
- We carry out extensive numerical simulations to evaluate the performance of the proposed solution, which is shown to be near-optimal and to improve significantly the users' offloading utility over traditional approaches.

**Chapter Organization:** The remainder of this chapter is organized as follows. In Sect. 5.2, we review the related works. In Sect. 5.3, we present the system model. The

joint task offloading and resource allocation problem is formulated in Sect. 5.4, followed by the decomposition of the problem itself. We present our proposed solution in Sect. 5.5 and numerical results in Sect. 5.6. Finally, in Sect. 5.7 we conclude the chapter.

## 5.2 Related Work

A number of solutions have also been proposed to exploit the potential benefits of MEC in the context of the IoTs and 5G. For instance, our previous work in [180] proposed to explore the synergies among the connected entities in the MEC network and presented three representative use-cases to illustrate the benefits of MEC collaboration in 5G networks. In [112], we proposed a collaborative caching and processing framework in a MEC network whereby the MEC servers can perform both caching and transcoding so as to facilitate Adaptive Bit-Rate (ABR) video streaming. Similar approach was also considered in [181] which combined the traditional client-driven dynamic adaptation scheme, DASH, with network-assisted adaptation capabilities. In addition, MEC is also seen as a key enabling technique for connected vehicles by adding computation and geo-distributed services to the roadside BSs so as to analyze the data from proximate vehicles and roadside sensors and to propagate messages to the drivers in very low latency [182].

Recently, several works have focused on exploiting the benefits of computation offloading in MEC network [183]. Note that similar problems have been investigated in conventional Mobile Cloud Computing (MCC) systems [184]. However, a large body of existing works on MCC assumed an infinite amount of computing resources available in a cloudlets, where the offloaded tasks can be executed with negligible delay [31,32,185]. The problem of offloading scheduling was then reduced to radio resource allocation in [34] where the competition for radio resources is modeled as a congestion game of selfish mobile users. In the context of MEC, the problem of joint task offloading and resource allocation was studied in a single-user system with energy harvesting devices [35], and in a multi-cell multi-user systems [36]; however the congestion of computing resources at the MEC server was omitted. Similar problem is studied in [178] considering the limited edge computing resources in a single-server MEC system.

Computation offloading in MEC can also utilizes the heterogeneous resource pool constituted by the end-user devices. In our previous work [180], a novel resource management framework was envisioned to orchestrate both the horizontal collaboration at the end-user layer and at the MEC layer as well as the vertical collaboration between end-users, edge nodes, and cloud nodes. Along this line, Chen et al. [186] proposed a Device-to-Device (D2D) Crowd framework where a massive crowd of devices at the network edge leverage network-assisted D2D collaboration for computation and communication resource sharing. Using game theoretic approach, the work in [187] considered a computation offloading problem that involves selfish mobile users that want to maximize their individual performance. However, the difference between [186, 187] and our work is multi-fold. First, the focus of [186, 186] is on the task assignment problem in which the data rates of the wireless links are considered as constants, whereas our work considers the joint design of task assignment and resource allocation policy for which the varying wireless channels and multi-user interference are taken into account when calculating the data rates. Second, the objective in [186] is to minimize the total energy consumption on mobile devices while we consider a parameterized objective function that can be tuned to optimize both the energy consumption and task execution time. Third, it is assumed in [187] that the users always use a constant transmit power while our approach aims at optimizing user transmit power.

In summary, most of the existing works did not consider a holistic approach that jointly determines the task offloading decision and the radio and computing resource allocation in a multi-cell, multi-server system as considered in this work.

## 5.3 System Model

We consider a multi-cell, multi-server MEC system as illustrated in Fig. 5.1, in which each BS is equipped with a MEC server to provide computation offloading services to the resource-constrained mobile users such as smart phones, tablets, and wearable devices. In general, each MEC server can be either a physical server or a virtual machine with moderate computing capabilities provisioned by the network operator and can communicate with the mobile devices through wireless channels provided by the corresponding BS. Each mobile user can choose to offload computation tasks to a MEC server from one of the nearby BSs

it can connect to. We denote the set of users and MEC servers in the mobile system as $\mathcal{U} = \{1, 2, ..., U\}$ and $\mathcal{S} = \{1, 2, ..., S\}$, respectively. For ease of presentation, we will refer to the MEC server $s$ and BS $s$ interchangeably. The modeling of user computation tasks, task uploading transmissions, MEC computation resources, and offloading utility are presented here below. For ease of reference, the key parameters used in this chapter are summarized in Table 5.1.

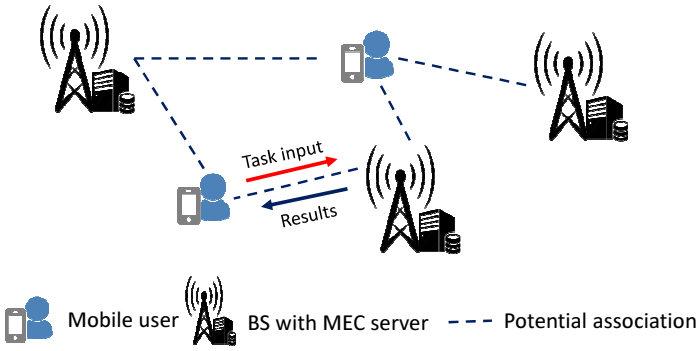



Figure 5.1: Example of a cellular system with MEC servers deployed at the BSs.

### 5.3.1 User Computation Tasks

We consider that each user $u \in \mathcal{U}$ has one computation task at a time, denoted as $T_u$, that is atomic and cannot be divided into subtasks. Each computation task $T_u$ is characterized by a tuple of two parameters, $\langle d_u, c_u \rangle$, in which $d_u$ [bits] specifies the amount of input data necessary to transfer the program execution (including system settings, program codes, and input parameters) from the local device to the MEC server, and $c_u$ [cycles] specifies the workload, i.e., the amount of computation to accomplish the task. The values of $d_u$ and $c_u$ can be obtained through carefully profiling of the task execution [173, 188]. Each task can be performed locally on the user device or offloaded to a MEC server. By offloading the computation task to the MEC server, the mobile user would save its energy for task execution; however, it would consume additional time and energy for sending the task input in the uplink.

Let $f_u^l > 0$ denote the local computing capability of user $u$ in terms of CPU cycles/s. Hence, if user $u$ executes its task locally, the task completion time is $t_u^l = \frac{c_u}{f_u^l}$ [seconds]. To calculate the energy consumption of a user device when executing its task locally, we use the

Table 5.1: Summary of Key Parameters for MEC Offloading System

| Notation | Description |
|---|---|
| $\mathcal{U}$ | Set of $U$ users |
| $\mathcal{S}$ | Set of $S$ BSs/MEC servers |
| $T_u$ | Computation task of user $u$ |
| $d_u$ | Input data of computation task $T_u$ |
| $c_u$ | Workload of computation task $T_u$ |
| $f_u^l$ | Local computing capability of user $u$ |
| $E_u^l$ | Energy consumption of user $u$ when executing its task locally |
| $E_u$ | Energy consumption of user $u$ when offloading its task |
| $\kappa$ | Energy coefficient of user device |
| $t_u^l$ | Local execution time of task $T_u$ |
| $t_{\text{up}}^u$ | Transmission time of task $T_u$ to the MEC server |
| $t_{\text{exe}}^u$ | Execution time of task $T_u$ at the MEC server |
| $t_u$ | Delay experienced by user $u$ when offloading its task |
| $B$ | Uplink system bandwidth |
| $N$ | Number of orthogonal sub-bands in the uplink |
| $\mathcal{N}$ | Set of $N$ orthogonal sub-bands |
| $x_{us}^j$ | Task offloading indicator, $\forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N}$ |
| $\mathcal{G}$ | Task offloading ground set |
| $\mathcal{X}$ | Task offloading policy |
| $\mathcal{U}_{\text{Off}}$ | Set of all users that offload their tasks |
| $\mathcal{U}_s$ | Set of users that offload their tasks to server $s$ |
| $h_{us}^j$ | Uplink channel gain between user $u$ and BS $s$ on sub-band $j$ |
| $p_u$ | Transmission power of user $u$ |
| $P_u$ | Maximum transmission power of user $u$ |
| $\gamma_{us}^j$ | SINR from user $u$ to BS $s$ on sub-band $j$ |
| $R_{us}$ | Uplink data rate from user $u$ to BS $s$ |
| $f_s$ | Computing capacity of server $s$ |
| $f_{su}$ | Computing resource that server $s$ allocates to task of user $u$ |
| $\mathcal{F}$ | Computing resource allocation policy |
| $J_u$ | Offloading utility of user $u$ |
| $\beta_u^t$ | Preference of user $u$ on task completion time |
| $\beta_u^e$ | Preference of user $u$ on energy consumption |
| $\lambda_u$ | Resource provider's preference towards user $u$ |

widely adopted model of the energy consumption per computing cycle as $\mathcal{E} = \kappa f^2$ [34, 189], where $\kappa$ is the energy coefficient depending on the chip architecture and $f$ is the CPU frequency. Thus, the energy consumption, $E_u^l\,[J]$, of user $u$ when executing its task $T_u$ locally, is calculated as,

$$E_u^l = \kappa \Big(f_u^l\Big)^2 c_u. \tag{1}$$

### 5.3.2 Task Uploading

In case user $u$ offloads its task $T_u$ to one of the MEC servers, the incurred delay comprises: (i) the time $t^u_{\text{up}}$ [s] to transmit the input to the MEC server on the uplink, (ii) the time $t^u_{\text{exe}}$ [s] to execute the task at the MEC server, and (iii) the time to transmit the output from the MEC server back to the user on the downlink. Since the size of the output is generally much smaller than the input, plus the downlink data rate is much higher than that of the uplink, we omit the delay of transferring the output in our computation, as also considered in [34, 178, 187]. Note that when the delay of downlink transmission of output data is non-negligible, our proposed algorithm can still be directly applied for a given downlink rate allocation scheme and known output data size.

In this work, we consider the system with OFDMA as the multiple access scheme in the uplink [190], in which the operational frequency band $B$ is divided into $N$ equal sub-bands of size $W = B/N$ [Hz]. To ensure the orthogonality of uplink transmissions among users associated with the same BS, each user is assigned to one sub-band. Thus, each BS can serve at most $N$ users at the same time. Let $\mathcal{N} = \{1, ..., N\}$ be the set of available sub-band at each BS. We define the task offloading variables, which also incorporate the uplink sub-band scheduling, as $x^j_{us}, u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N}$, where $x^j_{us} = 1$ indicates that task $T_u$ from user $u$ is offloaded to BS $s$ on sub-band $j$, and $x^j_{us} = 0$ otherwise. We define the ground set $\mathcal{G}$ that contains all the task offloading variables as $\mathcal{G} = \left\{ x^j_{us} \,|u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N} \right\}$ and the task offloading policy $\mathcal{X}$ expressed as $\mathcal{X} = \left\{ x^j_{us} \in \mathcal{G} \,\middle|\, x^j_{us} = 1 \right\}$. As each task can be either executed locally or offloaded to at most one MEC server, a feasible offloading policy must satisfy the constraint below,

$$\sum\nolimits_{s\in\mathcal{S}} \sum\nolimits_{j\in\mathcal{N}} x^j_{us} \leq 1, , \forall u \in \mathcal{U}. \tag{2}$$

Additionally, we denote $\mathcal{U}_s = \left\{ u \in \mathcal{U} \,\middle|\, \sum_{j\in\mathcal{N}} x^j_{us} = 1 \right\}$ as the set of users offloading their tasks to server $s$, and $\mathcal{U}_{\text{off}} = \bigcup_{s\in\mathcal{S}} \mathcal{U}_s$ as the set of users that offload their tasks.

Furthermore, we consider that each user and BS have a single antenna for uplink transmissions (as also considered in [178,191]). Extension to the case where each BS uses multiple antennas for receiving uplink signals will be addressed in a future work. Denote $h^j_{us}$ as the

uplink channel gain between user $u$ and BS $s$ on sub-band $j$, which captures the effect of path-loss, shadowing, and antenna gain. Note that the user-BS association usually takes place in a large time scale (duration of an offloading session) that is much larger than the time scale of small-scale fading. Hence, similar to [103], we consider that the effect of fast-fading is averaged out during the association. Let $\mathcal{P} = \{p_u \,|0 < p_u \leq P_u, u \in \mathcal{U}_{\text{off}}\}$ denote the users' transmission power, where $p_u$ [W] is the transmission power of user $u$ when uploading its task's input $d_u$ to the BS, subject to a maximum budget $P_u$. Note that $p_u = 0, \forall u \notin \mathcal{U}_{\text{off}}$. As the users transmitting to the same BS use different sub-bands, the uplink intra-cell interference is well mitigated; still, these users suffer from the inter-cell interference. In this case, the Signal-to-Interference-plus-Noise Ratio (SINR) from user $u$ to BS $s$ on sub-band $j$ is given by,

$$\gamma_{us}^{j} = \frac{p_u h_{us}^{j}}{\sum\limits_{r \in \mathcal{S} \setminus \{s\}} \sum\limits_{k \in \mathcal{U}_r} x_{kr}^{j} p_k h_{ks}^{j} + \sigma^2}, \forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N}, \tag{3}$$

where $\sigma^2$ is the background noise variance and the first term at the denominator is the accumulated intra-cell interference from all the users associated with other BSs on the same sub-band $j$. Since each user only transmits on one sub-band, the achievable rate [bits/s] of user $u$ when sending data to BS $s$ is given as,

$$R_{us}\left(\mathcal{X}, \mathcal{P}\right) = W \log_2\left(1 + \gamma_{us}\right), \tag{4}$$

where $\gamma_{us} = \sum_{j \in \mathcal{N}} \gamma_{us}^{j}$. Moreover, let $x_{us} = \sum_{j \in \mathcal{N}} x_{us}^{j}, \forall u \in \mathcal{U}, s \in \mathcal{S}$. Hence, the transmission time of user $u$ when sending its task input $d_u$ in the uplink can be calculated as,

$$t_{\text{up}}^{u} = \sum\nolimits_{s \in \mathcal{S}} \frac{x_{us} d_u}{R_{us}\left(\mathcal{X}, \mathcal{P}\right)}, \forall u \in \mathcal{U}. \tag{5}$$

### 5.3.3 MEC Computing Resources

The MEC server at each BS is able to provide computation offloading service to multiple users concurrently. The computing resources made available by each MEC server to be shared among the associating users are quantified by the computational rate $f_s$, expressed

in terms of number of CPU cycles/s. After receiving the offloaded task from a user, the server will execute the task on behalf of the user and, upon completion, will return the output result back to the user. We define the computing resource allocation policy as $\mathcal{F} = \{f_{us} \,|u \in \mathcal{U}, s \in \mathcal{S}\}$, in which $f_{us}$ [cycles/s] $> 0$ is the amount of computing resource that BS $s$ allocates to task $T_u$ offloaded from user $u \in \mathcal{U}_s$. Hence, clearly $f_{us} = 0, \forall u \notin \mathcal{U}_s$. In addition, a feasible computing resource allocation policy must satisfy the computing resource constraint, expressed as,

$$\sum\nolimits_{u \in \mathcal{U}} f_{us} \leq f_s, \forall s \in \mathcal{S}. \tag{6}$$

Given the computing resource assignment $\{f_{us}, s \in \mathcal{S}\}$, the execution time of task $T_u$ at the MEC servers is,

$$t^u_{\text{exe}} = \sum\nolimits_{s \in \mathcal{S}} \frac{x_{us} c_u}{f_{us}}, \forall u \in \mathcal{U}. \tag{7}$$

### 5.3.4 User Offloading Utility

Given the offloading policy $\mathcal{X}$, the transmission power $p_u$, and the computing resource allocation $f_{us}$'s, the total delay experienced by user $u$ when offloading its task is given by,

$$t_u = t^u_{\text{up}} + t^u_{\text{exe}} = \sum_{s \in \mathcal{S}} x_{us} \left( \frac{d_u}{R_{us}\left(\mathcal{X}, \mathcal{P}\right)} + \frac{c_u}{f_{us}} \right), \forall u \in \mathcal{U}. \tag{8}$$

The energy consumption of user $u$, $E_u\,[J]$, due to uploading transmission is calculated as $E_u = \frac{p_u t^u_{\text{up}}}{\xi_u}, \forall u \in \mathcal{U}$, where $\xi_u$ is the power amplifier efficiency of user $u$. Without loss of generality, we assume that $\xi_u = 1, \forall u \in \mathcal{U}$. Thus, the uplink energy consumption of user $u$ simplifies to,

$$E_u = p_u t^u_{\text{up}} = p_u d_u \sum_{s \in \mathcal{S}} \frac{x_{us}}{R_{us}\left(\mathcal{X}, \mathcal{P}\right)}, \forall u \in \mathcal{U}. \tag{9}$$

In a mobile cloud computing system, the users' QoE is mainly characterized by their task completion time and energy consumption. In the considered scenario, the relative improvement in task completion time and energy consumption are characterized by $\frac{t^l_u - t_u}{t^l_u}$ and

$\frac{E_u^l - E_u}{E_u^l}$, respectively [178]. Therefore, we define the offloading utility of user $u$ as,

$$J_u = \left(\beta_u^t \frac{t_u^l - t_u}{t_u^l} + \beta_u^e \frac{E_u^l - E_u}{E_u^l}\right) \sum_{s \in \mathcal{S}} x_{us}, \forall u \in \mathcal{U}, \tag{10}$$

in which $\beta_u^t, \beta_u^e \in [0,1]$, with $\beta_u^t + \beta_u^e = 1, \forall u \in \mathcal{U}$, specify user $u$'s preference on task completion time and energy consumption, respectively. For example, a user $u$ with short battery life can increase $\beta_u^e$ and decrease $\beta_u^t$ so as to save more energy at the expense of longer task completion time. In practice, the value of $\beta_u^e$ can be set by the mobile user through different power saving modes; for instance, $\beta_u^e = 1$ at "extreme power saving" mode and $\beta_u^e = 0$ at "maximum performance" mode. Alternatively, $\beta_u^t$ can be set proportional to the battery percentage level of the device. Note that offloading too many tasks to the MEC servers will cause excessive delay due to the limited bandwidth and computing resources at the MEC servers, and consequently degrade some users' QoE compared to executing their tasks locally. Hence, clearly user $u$ should not offload its task to the MEC servers if $J_u \leq 0$.

The expressions of the task completion time and energy consumption in (10) clearly shows the interplay between radio access and computational aspects, which motivates a joint optimization of offloading scheduling, radio, and computing resources so as to optimize users' offloading utility.

## 5.4 Problem Formulation

We formulate here the problem of joint task offloading and resource allocation, followed by the outline of our decomposition approach.

### 5.4.1 Joint Task Offloading and Resource Allocation Problem

For a given offloading decision $\mathcal{X}$, uplink power allocation $\mathcal{P}$, and computing resource allocation $\mathcal{F}$, we define the system utility as the weighted-sum of all the users' offloading utilities,

$$J(\mathcal{X}, \mathcal{P}, \mathcal{F}) = \sum\nolimits_{u \in \mathcal{U}} \lambda_u J_u, \tag{11}$$

with $J_u$ given in (10) and $\lambda_u \in (0, 1]$ specifying the resource provider's preference towards user $u$, $\forall u \in \mathcal{U}$. For instance, depending on the payments offered by the users, the resource provider could prioritize users with higher revenues for offloading by increasing their corresponding preferences. Additionally, $\lambda_u$ can also be set based on user type and the criticality of the computation task. For example, computation tasks from mobile devices involved in public safety operations (e.g., devices carried by police officers and first responders) should be prioritized with high value of $\lambda_u$. We now formulate the Joint Task Offloading and Resource Allocation (JTORA) problem as a system utility maximization problem, i.e.,

$$\max_{\mathcal{X},\mathcal{P},\mathcal{F}} \; J\left(\mathcal{X},\mathcal{P},\mathcal{F}\right) \tag{12a}$$

$$\text{s.t. } x_{us}^{j} \in \{0,1\}, \forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N}, \tag{12b}$$

$$\sum_{s\in\mathcal{S}}\sum_{j\in\mathcal{N}} x_{us}^{j} \leq 1, \forall u \in \mathcal{U}, \tag{12c}$$

$$\sum_{u\in\mathcal{U}} x_{us}^{j} \leq 1, \forall s \in \mathcal{S}, j \in \mathcal{N}, \tag{12d}$$

$$0 < p_u \leq P_u, \forall u \in \mathcal{U}_{\text{off}}, \tag{12e}$$

$$f_{us} > 0, \forall u \in \mathcal{U}_s, s \in \mathcal{S}, \tag{12f}$$

$$\sum\nolimits_{u\in\mathcal{U}} f_{us} \leq f_s, \forall s \in \mathcal{S}. \tag{12g}$$

The constraints in the formulation above can be explained as follows: constraints (12b) and (12c) imply that each task can be either executed locally or offloaded to at most one server on one sub-band; constraint (12d) implies that each BS can serve at most one user per sub-band; constraint (12e) specifies the transmission power budget of each user; finally, constraints (12f) and (12g) state that each MEC server must allocate a positive computing resource to each user associated with it and that the total computing resources allocated to all the associated users must not excess the server's computing capacity. The JTORA problem in (12) is a Mixed Integer Nonlinear Program (MINLP), which can be shown to be NP-hard; hence, finding the optimal solution usually requires exponential time complexity [192]. The NP-hardness proof of the JTORA problem is deferred to Sect. 5.5-C where we show that this problem contains a special case that is equivalent to a set

of submodular maximization problems. Given the large number of variables that scale linearly with the number of users, MEC servers, and sub-bands, our goal is to design a low-complexity, suboptimal solution that achieves competitive performance while being practical to implement.

### 5.4.2 Problem Decomposition

By exploiting the structure of the objective function and constraints in the formulation of JTORA problem in (12), we observe that by temporarily fixing the binary variables $\{x_{us}\}$, problem (12) can be decomposed into multiple subproblems with separated objective and constraints. Leveraging this characteristic and motivated by the approach in [193], we can employ Tammer decomposition method [194] to transform the original problem with high complexity into an equivalent *master problem* and a set of *subproblems* with lower complexity. Firstly, we rewrite the JTORA problem in (12) as,

$$\max_{\mathcal{X}} \left( \max_{\mathcal{P},\mathcal{F}} J\left(\mathcal{X}, \mathcal{P}, \mathcal{F}\right) \right) \tag{13a}$$

$$\text{s.t. } (12\text{b}) - (12\text{g}). \tag{13b}$$

Note that the constraints on the offloading decision, $\mathcal{X}$, in (12b), (12c), (12d), and the RA policies, $\mathcal{P}, \mathcal{F}$, in (12e), (12f), (12g), are decoupled from each other; therefore, solving the problem in (13) is equivalent to solving the following Task Offloading (TO) problem,

$$\max_{\mathcal{X}} J^*\left(\mathcal{X}\right) \tag{14a}$$

$$\text{s.t. } (12\text{b}), (12\text{c}), (12\text{d}), \tag{14b}$$

in which $J^*\left(\mathcal{X}\right)$ is the optimal-value function corresponding to the RA problem, written as,

$$J^*\left(\mathcal{X}\right) = \max_{\mathcal{P},\mathcal{F}} J\left(\mathcal{X}, \mathcal{P}, \mathcal{F}\right) \tag{15a}$$

$$\text{s.t. } (12\text{e}), (12\text{f}), (12\text{g}), \tag{15b}$$

Note that the decomposition from problem (12) to problems (14) and (15) does not change the optimality of the solution [194]. In the next section, we will present our solutions

to both the RA problem and the TO problem so as to finally obtain the solution to the original JTORA problem.

## 5.5 Proposed Low-complexity Algorithm

We present now our low-complexity approach to solve the JTORA problem by solving first the RA problem in (15) and then using its solution to derive the solution of the TO problem in (14).

Firstly, given a feasible task offloading decision $\mathcal{X}$ that satisfies constraints (12b), (12c), and (12d), and using the expression of $J_u$ in (10), the objective function in (15a) can be rewritten as,

$$J\left(\mathcal{X},\mathcal{P},\mathcal{F}\right)=\sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\lambda_u\left(\beta_u^t+\beta_u^e\right)-V\left(\mathcal{X},\mathcal{P},\mathcal{F}\right), \tag{16}$$

where

$$V\left(\mathcal{X},\mathcal{P},\mathcal{F}\right)=\sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\lambda_u\left(\frac{\beta_u^t t_u}{t_u^l}+\frac{\beta_u^e E_u}{E_u^l}\right). \tag{17}$$

We observe that the first term on the right hand side (RHS) of (16) is constant for a particular offloading decision, while $V\left(\mathcal{X},\mathcal{P},\mathcal{F}\right)$ can be seen as the total offloading overheads of all offloaded users. Hence, we can recast (15) as the problem of minimizing the total offloading overheads, i.e.,

$$\min_{\mathcal{P},\mathcal{F}} V\left(\mathcal{X},\mathcal{P},\mathcal{F}\right) \tag{18a}$$

$$\text{s.t. } (12e),(12f),(12g). \tag{18b}$$

Furthermore, from (8), (9), and (17), we have,

$$V\left(\mathcal{X},\mathcal{P},\mathcal{F}\right)=\sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\frac{\phi_u+\psi_u p_u}{\log_2\left(1+\gamma_{us}\right)}+\sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\frac{\eta_u}{f_{us}}, \tag{19}$$

in which, for simplicity, $\phi_u=\frac{\lambda_u\beta_u^t d_u}{t_u^l W}$, $\psi_u=\frac{\lambda_u\beta_u^e d_u}{E_u^l W}$, and $\eta_u=\lambda_u\beta_u^t f_u^l$. Notice from (18b) and (19) that the problem in (18) has a *separable structure*, i.e., the objectives and constraints corresponding to the power allocation $p_u$'s and computing resource allocation $f_{us}$'s can be

decoupled from each other. Leveraging this property, we can decouple problem (18) into two independent problems, namely the *Uplink Power Allocation (UPA)* and the *Computing Resource Allocation (CRA)*, and address them separately in the following subsections.

### 5.5.1 Uplink Power Allocation

The UPA problem is decoupled from problem (18) by considering the first term on the RHS of (19) as the objective function. Specifically, the UPA problem is expressed as,

$$\min_{\mathcal{P}} \sum\nolimits_{s\in\mathcal{S}} \sum\nolimits_{u\in\mathcal{U}_s} \frac{\phi_u + \psi_u p_u}{\log_2(1+\gamma_{us})} \tag{20a}$$

$$\text{s.t. } 0 < p_u \leq P_u, \forall u \in \mathcal{U}. \tag{20b}$$

Problem (20) is non-convex and difficult to solve because the uplink SINR $\gamma_{us}^j$ corresponding to user $u \in \mathcal{U}_s$ depends on the transmit power of the other users associated with other BSs on the same sub-band $j$ through the inter-cell interference $I_{us}^j = \sum\limits_{w\in\mathcal{S}\setminus\{s\}} \sum\limits_{k\in\mathcal{U}_w} x_{ks}^j p_k h_{ks}^j$, as seen in (3). Our approach is to find an approximation for $I_{us}^j$ and thus for $\gamma_{us}^j$ such that problem (20) can be decomposed into sub-problems that, in turn, can be efficiently solved. The optimal uplink power allocation $\mathcal{P}^*$ still generates small objective value for (20). Suppose each BS $s \in \mathcal{S}$ calculates its uplink power allocation independently, i.e., without mutual cooperation, and informs its associated users about the uplink transmit power; then, an achievable upper bound for $I_{us}^j$ is given by,

$$\tilde{I}_{us}^j \triangleq \sum_{w\in\mathcal{S}\setminus\{s\}} \sum_{k\in\mathcal{U}_w} x_{ks}^j P_k h_{ks}^j, \forall u \in \mathcal{U}_s, s \in \mathcal{S}, j \in \mathcal{N}. \tag{21}$$

Similar to [195], we argue that $\tilde{I}_{us}^j$ is a good estimate of $I_{us}^j$ since our offloading decision $\mathcal{X}$ is geared towards choosing the appropriate user-BS associations so that $\tilde{I}_{us}^j$ is small in the first place. This means that a small error in $I_{us}^j$ should not lead to large bias in $\gamma_{us}^j$ [195].

By replacing $I_{us}^j$ with $\tilde{I}_{us}^j$, we get the approximation for the uplink SINR for user $u$ uploading to BS $s$ on sub-band $j$ as,

$$\tilde{\gamma}_{us}^j = \frac{p_u h_{us}^j}{\tilde{I}_{us}^j + \sigma^2}, \forall u \in \mathcal{U}_s, s \in \mathcal{S}, j \in \mathcal{N}. \tag{22}$$

Let $\vartheta_{us} = \sum_{j\in\mathcal{N}} h_{us}^{j} / \left(\tilde{I}_{us}^{j} + \sigma^2\right)$ and $\Gamma_s(p_u) = \frac{\phi_u + \psi_u p_u}{\log_2(1+\vartheta_{us} p_u)}$. The objective function in (20a) can now be approximated by $\sum_{s\in\mathcal{S}} \sum_{u\in\mathcal{U}_s} \Gamma_s(p_u)$. With this position, it can be seen that the objective function and the constraint corresponding to each user's transmit power is now decoupled from each other. Therefore, the UPA problem in (20) can be approximated by $\sum_{s\in\mathcal{S}} |\mathcal{U}_s|$ sub-problems, each optimizing the transmit power of a user $u \in \mathcal{U}_s, s \in \mathcal{S}$, and can be written as,

$$\min \sum\nolimits_{u\in\mathcal{U}_s} \Gamma_s(p_u) \tag{23a}$$

$$\text{s.t. } 0 < p_u \leq P_u. \tag{23b}$$

Problem (23) is still non-convex as the second-order derivative of the objective function with respect to (w.r.t) $p_u$, i.e., $\Gamma_s''(p_u)$, is not always positive. However, we can employ quasi-convex optimization technique to address problem (23) based on the following lemma.

**Lemma 10.** *$\Gamma_s(p_u)$ is strictly quasi-convex in the domain defined in* (23b).

*Proof.* Firstly, it is straightforward to verify that $\Gamma_s(p_u)$ is twice differentiable on $\mathbb{R}$. We now check the second-order condition of a strictly quasi-convex function, which requires that a point $p$ satisfying $\Gamma_s'(p) = 0$ also satisfies $\Gamma_s'(p) > 0$ [78].

The first-order and second-order derivatives of $\Gamma_s(p_u)$ can be calculated, respectively, as,

$$\Gamma_s'(p_u) = \frac{\psi_u C_u(p_u) - \frac{\vartheta_{us} D_u(p_u)}{A_u(p_u)\ln 2}}{C_u^2(p_u)}, \tag{24}$$

and

$$\Gamma_s''(p_u) = \frac{\vartheta_{us}\left[G_{us}(p_u) C_{us}(p_u) + 2\vartheta_{us} D_{us}(p_u)/\ln 2\right]}{A_{us}^2(p_u) C_{us}^3(p_u)\ln 2}, \tag{25}$$

in which,

$$A_{us}(p_u) = 1 + \vartheta_{us} p_u, \tag{26a}$$

$$C_{us}(p_u) = \log_2(1 + \vartheta_{us} p_u), \tag{26b}$$

$$D_{us}(p_u) = \phi_u + \psi_u p_u, \tag{26c}$$

$$G_{us}(p_u) = \vartheta_{us} D_{us}(p_u) - 2\psi_u A_{us}(p_u). \tag{26d}$$

Suppose that $\bar{p}_u \in (0, P_u]$; to satisfy $\Gamma'_s(\bar{p}_u) = 0$, it must hold that,

$$\Omega_s(\bar{p}_u) = \psi_u \log_2(1 + \vartheta_{us}\bar{p}_u) - \frac{\vartheta_{us}(\phi_u + \psi_u \bar{p}_u)}{(1 + \vartheta_{us}\bar{p}_u)\ln 2} = 0. \tag{27}$$

By substituting $\bar{p}_u$ into (25), we obtain,

$$\Gamma''_s(\bar{p}_u) = \frac{\vartheta_{us}^3 D_{us}^2(\bar{p}_u)}{A_{us}^2(\bar{p}_u) C_{us}^3(\bar{p}_u) \psi_u \ln^2 2}. \tag{28}$$

It can be easily verified that both $\vartheta_{us}$ and $D_{us}^2(\bar{p}_u)$ are strictly positive $\forall \bar{p}_u \in (0, P_u]$. Hence, $\Gamma''_s(\bar{p}_u) > 0$, which confirms that $\Gamma_s(p_u)$ is a strictly quasi-convex function in $(0, P_u]$. □

In general, a quasi-convex problem can be solved using the bisection method, which solves a convex feasibility problem in each iteration [78]. However, the popular interior cutting plane method for solving a convex feasibility problem requires $\mathcal{O}\left(n^2/\epsilon^2\right)$ iterations, where $n$ is the dimension of the problem. We now propose to further reduce the complexity of the bisection method.

Firstly, notice that a quasi-convex function achieves a local optimum at the diminishing point of the first-order derivative, and that any local optimum of a strictly quasi-convex function is the global optimum [196]. Therefore, based on Lemma 10, we can confirm that the optimal solution $p_u^*$ of problem (23) either lies at the constraint border, i.e., $p_u^* = P_u$ or satisfies $\Gamma'_s(p_u^*) = 0$. It can be verified that $\Gamma'_s(p_u) = 0$ when,

$$\Omega_s(p_u) = \psi_u \log_2(1 + \vartheta_{us}p_u) - \frac{\vartheta_{us}(\phi_u + \psi_u p_u)}{(1 + \vartheta_{us}p_u)\ln 2} = 0. \tag{29}$$

Moreover, we have, $\Omega'_s(p_u) = \frac{\vartheta_{us}^2(\phi_u + \psi_u p_u)}{(1+\vartheta_{us}p_u)^2 \ln 2} > 0$, and $\Omega_s(0) = -\frac{\vartheta_{us}\phi_u}{\ln 2} < 0$. This implies that $\Omega_s(p_u)$ is a monotonically increasing function and is negative at the starting point $p_u = 0$. Therefore, we can design a low-complexity bisection method that evaluates $\Omega_s(p_u)$ in each iteration instead of solving a convex feasibility problem, so as to obtain the optimal solution $p_u^*$, as presented in Algorithm 9.

In Algorithm 9, if $\Omega_s(P_u) > 0$, the algorithm will terminate in exactly $\lceil \log_2(P_u/\xi) \rceil$ iterations, where $\xi$ is the convergence threshold in line 14. Let $\mathcal{P}^* = \{p_u^*, u \in \mathcal{U}\}$ denote the

**Algorithm 9** Bisection Method for Uplink Power Allocation

Calculate $\Omega_s(P_u)$ using (29)
**if** $\Omega_s(P_u) \leq 0$ **then**
    $p_u^* = P_u$
**else**
    Set optimality tolerance $\epsilon > 0$
    Initialize $p_u' = 0$ and $p_u'' = P_u$
    **repeat**
        Set $p_u^* = (p_u' + p_u'')/2$
        **if** $\Omega_s(p_u^*) \leq 0$ **then**
            Set $p_u' = p_u^*$
        **else**
            Set $p_u'' = p_u^*$
    **until** $p_u'' - p_u' \leq \xi$
    Set $p_u^* = (p_u' + p_u'')/2$

optimal uplink transmit power policy for a given task offloading policy $\mathcal{X}$. In addition, we denote now as $\Gamma(\mathcal{X}, \mathcal{P}^*)$ the objective value of problem (20) corresponding to $\mathcal{P}^*$.

### 5.5.2 Computing Resource Allocation

The CRA problem optimizes the second term on the RHS of (19) and is expressed as follows,

$$\min_{\mathcal{F}} \sum\nolimits_{s\in\mathcal{S}} \sum\nolimits_{u\in\mathcal{U}_s} \eta_u / f_{us} \tag{30a}$$

$$\text{s.t.} \ \sum\nolimits_{u\in\mathcal{U}} f_{us} \leq f_s, \forall s \in \mathcal{S}, \tag{30b}$$

$$f_{us} > 0, \forall u \in \mathcal{U}_s, s \in \mathcal{S}. \tag{30c}$$

Notice that the constraints in (30b) and (30c) are convex. Denote the objective function in (30a) as $\Lambda(\mathcal{X}, \mathcal{F})$; by calculating the second-order derivatives of $\Lambda(\mathcal{X}, \mathcal{F})$ w.r.t. $f_{us}$, we have,

$$\frac{\partial^2 \Lambda(\mathcal{X}, \mathcal{F})}{\partial f_{us}^2} = \frac{2\eta_u}{f_{us}^3} > 0, \forall s \in \mathcal{S}, u \in \mathcal{U}_s, \tag{31a}$$

$$\frac{\partial^2 \Lambda(\mathcal{X}, \mathcal{F})}{\partial f_{us} \partial f_{vw}} = 0, \forall (u,s) \neq (v,w). \tag{31b}$$

It can be seen that the Hessian matrix of the objective function in (30a) is diagonal with the strictly positive elements, thus it is positive-definite. Hence, (30) is a convex

optimization problem and can be solved using Karush-Kuhn-Tucker (KKT) conditions. We have the following Lemma.

**Lemma 11.** *The optimal computing resource allocation $f_{us}^*$ for problem (30) and the corresponding optimal objective function $\Lambda(\mathcal{X}, \mathcal{F}^*)$ are given, respectively, as,*

$$f_{us}^* = \frac{f_s\sqrt{\eta_u}}{\sum_{u\in\mathcal{U}_s}\sqrt{\eta_u}}, \forall s \in \mathcal{S}, u \in \mathcal{U}_s, \tag{32}$$

$$\Lambda(\mathcal{X}, \mathcal{F}^*) = \sum\nolimits_{s\in\mathcal{S}} \frac{1}{f_s}\Big(\sum\nolimits_{u\in\mathcal{U}_s}\sqrt{\eta_u}\Big)^2. \tag{33}$$

*Proof.* The Lagrangian of problem (30) can be calculated as,

$$\mathcal{L}(\Lambda(\mathcal{X}, \mathcal{F}), \nu) = \sum_{s\in\mathcal{S}}\sum_{u\in\mathcal{U}_s}\frac{\eta_u}{f_{us}} + \sum_{s\in\mathcal{S}}\nu_s\left(\sum_{u\in\mathcal{U}} f_{us} - f_s\right), \tag{34}$$

where $\boldsymbol{\nu} = [v_1, ...v_S]$ is the vector of Lagrangian multipliers. Taking the derivatives of the Lagrangian w.r.t. $f_{us}$'s, we get,

$$\frac{\partial\mathcal{L}(\Lambda(\mathcal{X}, \mathcal{F}), \nu)}{\partial f_{us}} = -\frac{\eta_u}{f_{us}^2} + \nu_s, \forall s \in \mathcal{S}, u \in \mathcal{U}_s. \tag{35}$$

By equating the gradient of the Lagrangian to zero and solving for $f_{us}$, the optimal computing resource allocation solution for problem (30) is obtained as,

$$f_{us}^* = \sqrt{\eta_u/\nu_s^*}, \forall s \in \mathcal{S}, u \in \mathcal{U}_s, \tag{36}$$

in which $\nu_s^* > 0$ is the constant satisfying,

$$\sum\nolimits_{u\in\mathcal{U}} f_{us}^* = f_s, \forall s \in \mathcal{S}. \tag{37}$$

By substituting (36) into (37) and noting that $f_{us}^* = 0,\ \forall u \notin \mathcal{U}_s$, we obtain the optimal Lagrangian multiplier $\nu_s^*$ as,

$$\nu_s^* = \left(\frac{1}{f_s}\sum\nolimits_{u\in\mathcal{U}_s}\sqrt{\eta_u}\right)^2, \forall s \in \mathcal{S} \tag{38}$$

Finally, by substituting (38) into (36), we can obtain the optimal solution to problem (30) in closed form as,

$$f_{us}^{*} = \frac{f_s \sqrt{\eta_u}}{\sum_{u \in \mathcal{U}_s} \sqrt{\eta_u}}, \forall s \in \mathcal{S}, u \in \mathcal{U}_s, \tag{39}$$

and the optimal objective function of problem (30) is then calculated as,

$$\Lambda\left(\mathcal{X}, \mathcal{F}^{*}\right) = \sum\nolimits_{s \in \mathcal{S}} \frac{1}{f_s} \Big( \sum\nolimits_{u \in \mathcal{U}_s} \sqrt{\eta_u} \Big)^2. \tag{40}$$

□

### 5.5.3 Joint Task Offloading Scheduling and Resource Allocation

In the previous sections, for a given task offloading decision $\mathcal{X}$, we obtained the solutions for the radio and computing resources allocation. In particular, according to (15), (16), (19), and (33), we have,

$$J^{*}\left(\mathcal{X}\right) = \sum_{s \in \mathcal{S}} \sum_{u \in \mathcal{U}_s} \lambda_u \left(\beta_u^t + \beta_u^e\right) - \Gamma\left(\mathcal{X}, \mathcal{P}^{*}\right) - \Lambda\left(\mathcal{X}, \mathcal{F}^{*}\right), \tag{41}$$

where $\mathcal{P}^{*}$ can be obtained through Algorithm 9 and $\Lambda\left(\mathcal{X}, \mathcal{F}^{*}\right)$ can be calculated using the closed-form expression in (33). Now, using (41), we can rewrite the TO problem in (14) as,

$$\max_{\mathcal{X}} \sum_{s \in \mathcal{S}} \sum_{u \in \mathcal{U}_s} \lambda_u \left(\beta_u^t + \beta_u^e\right) - \Gamma\left(\mathcal{X}, \mathcal{P}^{*}\right) - \Lambda\left(\mathcal{X}, \mathcal{F}^{*}\right) \tag{42a}$$

$$\text{s.t. } x_{us}^{j} \in \{0,1\}, \forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N}, \tag{42b}$$

$$\sum_{s \in \mathcal{S}} \sum_{j \in \mathcal{N}} x_{us}^{j} \leq 1, \forall u \in \mathcal{U}, \tag{42c}$$

$$\sum_{u \in \mathcal{U}} x_{us}^{j} \leq 1, \forall s \in \mathcal{S}, j \in \mathcal{N}. \tag{42d}$$

Problem (42) consists in maximizing a set function $J^{*}\left(\mathcal{X}\right)$ w.r.t $\mathcal{X}$ over the ground set $\mathcal{G}$ defined by (42b), and the constraints in (42c) and (42d) define two matroids over $\mathcal{G}$[1]. We now show the intractability of problem (42) in the following theorem.

**Theorem 2.** *Problem* (42) *is NP-hard.*

[1]For detailed definition of matroid constraint on set function, refer to [197].

*Proof.* We will show the NP-hardness of problem (42) by reduction from a submodular maximization problem subject to matroid constraints, which is known to be NP-hard [197]. Firstly, we reduce problem (42) to a special case where each user can only offload its task to the closest server by setting $h_{us}^j > 0$ if $s$ is the closest BS to user $u$, and $h_{us}^j = 0$ otherwise. In this case, problem (42) can be decomposed into $S$ independent Single-Server Task Offloading (SSTO) problems, each corresponding to a server $s$. Let $\tilde{\mathcal{U}}_s$ be the set of users that can only offload tasks to server $s$; additionally, denote $\mathcal{G}_s = \left\{ x_{us}^j \middle| u \in \tilde{\mathcal{U}}_s, j \in \mathcal{N} \right\}$, and $\mathcal{X}_s = \left\{ x_{us}^j \in \mathcal{G}_s \middle| x_{us}^j = 1 \right\}$. The $s$-th SSTO problem can be expressed as follows.

$$\max_{\mathcal{X}_s} \sum_{u \in \mathcal{U}_s} \lambda_u \left( \beta_u^t + \beta_u^e \right) - \Gamma \left( \mathcal{X}_s, \mathcal{P}^* \right) - \Lambda \left( \mathcal{X}_s, \mathcal{F}^* \right) \tag{43a}$$

$$\text{s.t. } x_{us}^j \in \{0, 1\}, \forall u \in \tilde{\mathcal{U}}_s, j \in \mathcal{N}, \tag{43b}$$

$$\sum_{j \in \mathcal{N}} x_{us}^j \leq 1, \forall u \in \tilde{\mathcal{U}}_s, \tag{43c}$$

$$\sum_{u \in \tilde{\mathcal{U}}_s} x_{us}^j \leq 1, \forall s \in \mathcal{S}, j \in \mathcal{N}. \tag{43d}$$

We now show that the objective function in (43a), denoted as $J_s^* \left( \mathcal{X}_s \right)$, is submodular over the ground set $\mathcal{G}_s$. For any $x_{vs}^k \in \mathcal{G}_s$ and $x_{vs}^k \notin \mathcal{X}_s$, the marginal value of the objective function corresponding to the addition of $x_{vs}^k$ to the current set $\mathcal{X}_s$ is calculated as,

$$J_{s,vk}^* \left( \mathcal{X}_s \right) = J_s^* \left( \mathcal{X}_s \cup \left\{ x_{vs}^k \right\} \right) - J_s^* \left( \mathcal{X}_s \right). \tag{44}$$

Using (20a) and (33) we have,

$$J_{s,vk}^* \left( \mathcal{X}_s \right) = \lambda_v \left( \beta_v^t + \beta_v^e \right) - \frac{\phi_v + \psi_v p_v^*}{\log_2 \left( 1 + \gamma_{vs} \right)} - \left( \eta_v + 2\sqrt{\eta_v} \sum\nolimits_{u \in \mathcal{U}_s} \sqrt{\eta_u} \right) / f_s. \tag{45}$$

Recall that $\mathcal{U}_s = \left\{ u \in \mathcal{U} \middle| \sum_{j \in \mathcal{N}} x_{us}^j = 1 \right\}$; it is straightforward to verify that, on the RHS of (45), the last term increases with $\mathcal{X}_s$ while the first two terms do not depend on $\mathcal{X}_s$. Therefore, the marginal value $J_{s,vk}^* \left( \mathcal{X}_s \right)$ monotonically decreases with the offloading decision set $\mathcal{X}_s$. In other words, we get diminishing return in offloading utility as the offloading set gets bigger. Thus, the objective function in (43a) is a submodular function; and hence, each

SSTO problem is a submodular maximization problem over matroid constraints, which is NP-hard according to [197]. This confirms that the TO problem in (42), which contains as special case a set of SSTO problems, is also NP-hard. □

Given the NP-hardness of the TO problem, solving for an optimal solution in polynomial time is extremely challenging. One straightforward approach to solve problem (42) is to use exhaustive search method over all possible task offloading decisions. However, the total number of candidate task offloading decisions would be $2^n$ where $n = S \times U \times N$. Hence, the exhaustive search method is clearly impractical. Hence, we propose a low-complexity heuristic algorithm that can find a local optimum to problem (42) in polynomial time. Specifically, our algorithm starts with an empty set $\mathcal{X} = \emptyset$ and repeatedly performs one of the local operations, namely the *remove* operation or the *exchange* operation, as described in Routine 4, if it improves the set value $J^*(\mathcal{X})$. As we are dealing with two matroid constraints, the *exchange* operation involves adding one element from outside of the current set and dropping up to 2 elements from the set, so as to comply with the constraints. Our proposed heuristic algorithm for task offloading scheduling is presented in Algorithm 10.

**Routine 4** *remove* and *exchange* operations

*remove* $\left(\mathcal{X}, x_{us}^j\right)$

Set $\mathcal{X} \leftarrow \mathcal{X} \backslash \left\{x_{us}^j\right\}$

Output: $\mathcal{X}$

*exchange* $\left(\mathcal{X}, x_{us}^j\right)$

**for** $w \in \mathcal{S}, i \in \mathcal{N}$ **do**

    $\mathcal{X} \leftarrow \mathcal{X} \backslash \left\{x_{uw}^i\right\}$

**for** $v \in \mathcal{U}$ **do**

    $\mathcal{X} \leftarrow \mathcal{X} \backslash \left\{x_{vs}^j\right\}$

Set $\mathcal{X} \leftarrow \mathcal{X} \cup \left\{x_{us}^j\right\}$

Output: $\mathcal{X}$

**Remark 9**: (Complexity Analysis of Algorithm 10) Parameter $\epsilon > 0$ in Algorithm 10 is any value such that $\frac{1}{\epsilon}$ is at most a polynomial in $n$. Let $\mathrm{Opt}(\mathcal{G})$ be the optimal value of problem (42) over the ground set $\mathcal{G}$. It is easy to see that $\mathrm{J}^*\left(\left\{x_{kw}^i\right\}\right) \leq \mathrm{Opt}(\mathcal{G})/n$ where $x_{kw}^i$ is the element with the maximum $\mathrm{J}^*\left(\left\{x_{us}^j\right\}\right)$ over all elements of $\mathcal{G}$. Let $t$ be the number of iterations for Algorithm 10. Since after each iteration the value of the function

**Algorithm 10** Heuristic Task Offloading Scheduling

1: Initialize: $\mathcal{X} = \emptyset$
2: Find $x_{kw}^i = \underset{x_{us}^j, j\in\mathcal{N}, s\in\mathcal{S}, u\in\mathcal{U}}{\arg\max} \quad J^*\left(\left\{x_{us}^j\right\}\right)$
3: Set $\mathcal{X} \leftarrow \left\{x_{kw}^i\right\}$
4: **if** there exists $x_{us}^j \in \mathcal{X}$ such that $J^*\left(remove\left(\mathcal{X}, x_{us}^j\right)\right) > \left(1+\frac{\epsilon}{n^2}\right) J^*\left(\mathcal{X}\right)$ **then**
5: Set $\mathcal{X} \leftarrow remove\left(\mathcal{X}, x_{us}^j\right)$
6: Go back to step 4
7: **else if** there exists $x_{us}^j \in \mathcal{G}\backslash\mathcal{X}$ such that $J^*\left(exchange\left(\mathcal{X}, x_{us}^j\right)\right) > \left(1+\frac{\epsilon}{n^2}\right) J^*\left(\mathcal{X}\right)$ **then**
8: Set $\mathcal{X} \leftarrow exchange\left(\mathcal{X}, x_{us}^j\right)$
9: Go back to step 4
10: Output: $\mathcal{X}$

increases by a factor of at least $\left(1+\frac{\epsilon}{n^2}\right)$, we have $\left(1+\frac{\epsilon}{n^2}\right)^t \leq n$, and thus $t = \mathcal{O}\left(\frac{1}{\epsilon}n^2\log n\right)$. Note that the number of queries needed to calculate the value of the objective function in each iteration is at most $n$. In each query, one needs to solve an UPA problem and a CRA problem. Since the CRA problem has a closed-form solution, the running time complexity in each query mainly comes from solving the UPA problem, which takes $\lceil\log_2\left(P_u/\xi\right)\rceil$ iterations. Therefore, the running time of Algorithm 10 is $\mathcal{O}\left(\frac{1}{\epsilon}\lceil\log_2\left(P_u/\xi\right)\rceil n^3\log n\right)$, which is polynomial in $n$. Furthermore, since the queries in each iteration can be computed independently, the runtime can be greatly reduced by utilizing parallel computing. □

**Remark 10**: (JTORA solution) Let $\mathcal{X}^*$ be the output of Algorithm 10. The corresponding solutions $\mathcal{P}^*$ for the uplink power allocation and $\mathcal{F}^*$ for computing resource sharing can be obtained using Algorithm 9 and the closed-form expression in (32), respectively, by setting $\mathcal{X} = \mathcal{X}^*$. Thus, the local optimal solution for the JTORA problem is $(\mathcal{X}^*, \mathcal{P}^*, \mathcal{F}^*)$. While characterizing the degree of suboptimality of the proposed solution is a non-trivial task—mostly due to the combinatorial nature of the task offloading decision and the nonconvexity of the original UPA problem—in the next section we will show via numerical results that our heuristic algorithm performs closely to the optimal solution using exhaustive search method. □

## 5.6 Performance Evaluation

Simulation results are presented to evaluate the performance of our proposed heuristic joint task offloading scheduling and resource allocation strategy, referred to as hJTORA. We consider a multi-cell cellular system consisting of multiple hexagonal cells with a BS in the center of each cell. The neighboring BSs are set 1 km apart from each other. We assume that both the users and BSs use a single antenna for uplink transmission and reception, respectively. The uplink channel gains are generated using a distance-dependent path-loss model given as $L\,[\text{dB}] = 140.7 + 36.7\log_{10} d_{[\text{km}]}$ [198], and the log-normal shadowing variance is set to 8 dB. In most simulations, if not stated otherwise, we consider $S = 7$ cells and the users' maximum transmit power set to $P_u = 20$ dBm. In addition, the system bandwidth is set to $B = 20$ MHz and the background noise variance is assumed to be $\sigma^2 = -100$ dBm.

In terms of computing resources, we assume the CPU capability of each MEC server and of each user to be $f_s = 20$ GHz and $f_u^l = 1$ GHz, respectively. According to the realistic measurements in [188], we set the energy coefficient $\kappa$ as $5 \times 10^{-27}$. For computation task, we consider the face detection and recognition application for airport security and surveillance [170], which can highly benefit from the collaboration between mobile devices and MEC platform. Unless otherwise stated, we choose the default task input size as $d_u = 420$ KB (following [170, 175]), and the preference parameters as $\beta_u^t = 0.2$, $\beta_u^e = 0.8$, and $\lambda_u = 1$, $\forall u \in \mathcal{U}$. In addition, the users are dropped in random locations, with uniform distribution, within the coverage area of the network, and the number of sub-bands $N$ is set equal to the number of users per cell. We compare the system utility performance of our proposed hJTORA strategy against the following baselines.

- **Exhaustive**: This is a brute-force method that finds the optimal offloading scheduling solution via exhaustive search over $2^n$ possible decisions; since the computational complexity of this method is very high, we only evaluate its performance in a small network setting.

- *Greedy Offloading and Joint Resource Allocation (***GOJRA***)*: All tasks (up to the maximum number that can be admitted by the BSs) are offloaded, as in [36]. In each cell, offloading users are *greedily* assigned to sub-bands that have the highest channel

gains until all users are admitted or all the sub-bands are occupied; we then apply joint joint resource allocation across the BSs as proposed in Sect. 5.5-A, B.

- *Independent Offloading and Joint Resource Allocation (***IOJRA***)*: Each user is randomly assigned a sub-band from its home BS, then the users independently make offloading decision [31]; joint resource allocation is employed.

- *Distributed Offloading and Resource Allocation (***DORA***)*: Each BS independently makes joint task offloading decisions and resource allocation for users within its cell [178].

### 5.6.1 Suboptimality and Convergence Behavior

Firstly, to characterize the suboptimality of our proposed hJTORA solution obtained using Algorithm 10, we compare its performance with the optimal solution obtained by the *Exhaustive* method, and then with the three other described baselines. Since the *Exhaustive* method searches over all possible offloading scheduling decisions, its runtime is extremely long for a large number of variables; hence, we carry out the comparison in a small network setting with $U = 6$ users uniformly placed in the area covered by $S = 4$ cells, each having $N = 2$ sub-bands. We randomly generate 500 large-scale fading (shadowing) realizations and the average system utilities, with 95% Confidence Interval (CI), of different schemes are reported in Fig. 5.2 when we set $c_u = 1000, 1500$, and 2000 Megacycles, respectively. It can be seen that the proposed hJTORA algorithm performs very closely to that of the optimal *Exhaustive* algorithm while it significantly outperforms the other baselines. We also observe that the performance of all schemes increases with the task workload. In all cases, hJTORA achieves an average system utility within 1% of that of the *Exhaustive* algorithm, while providing average gains of 31%, 38%, and 91% over DORA, GOJRA, and IOJRA schemes, respectively.

In Table 5.2, we report the average runtime per simulation drop of different algorithms, running on a Windows 7 desktop PC with 3.6 GHz quad-core CPU and 16 GB RAM. It can be seen that the *Exhaustive* method takes very long time, about 100× longer than the hJTORA algorithm for such a small network. The DORA algorithm runs slightly faster

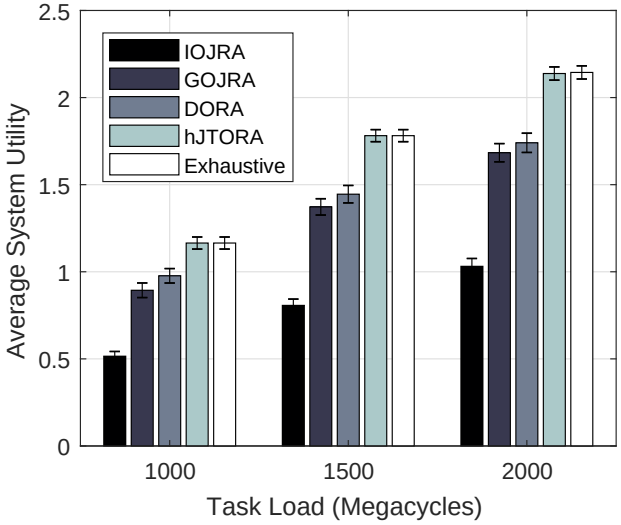



Figure 5.2: Comparison of average system utility.

than hJTORA, while IOJRA and GOJRA requires the lowest runtimes.

Table 5.2: Runtime Comparison Among Competing Schemes for MEC Offloading

| | **IOJRA** | **GOJRA** | **DORA** | **hJTORA** | **Exhaustive** |
|---|---|---|---|---|---|
| **Runtime** [ms] | $0.2 \pm 0.03$ | $1.8 \pm 0.2$ | $6.8 \pm 0.22$ | $19.3 \pm 0.7$ | $1,923 \pm 1.4$ |

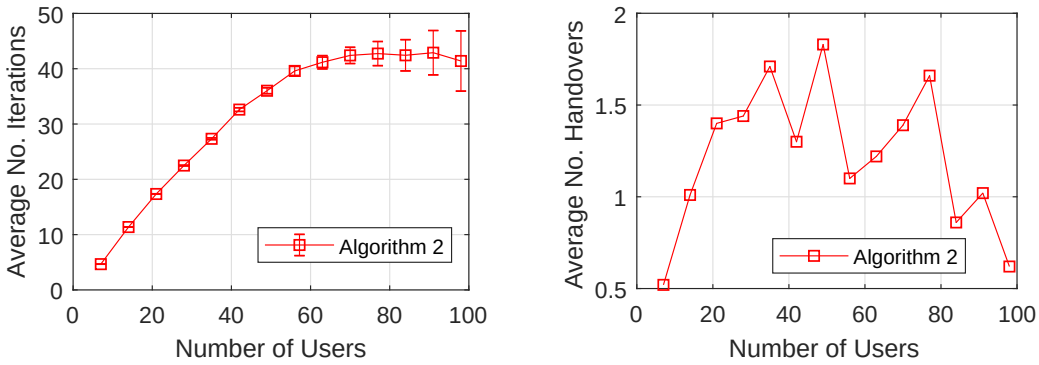
Furthermore, in order to evaluate the runtime of Algorithm 10 when more BSs cooperate with each other, we consider three deployment scenarios: *denser*, *very dense*, and *ultra dense* networks for which the BS spacing (distance between two neighboring BSs) are set to 237 m, 209 m, and 112 m according to [199]. The number of BSs and users are set as in Table 5.3, and the average runtime of Algorithm 10 (with 95% CI) are reported therein. We can see that the runtime increases considerably when there are many BSs cooperating with each other (e.g., 16 or 25). Therefore, in order to make the proposed solution practical, it is advisable to divide the network region into groups of cooperating BSs, each with fewer than 10 BSs.

### 5.6.2 Impact of Number of Users

In Fig. 5.3(a), we evaluate the convergence behavior of Algorithm 10 against different number of users by showing the average number of iterations with 95% CI. It can be seen that

Table 5.3: Runtime of Algorithm 10 Versus Network Size

| **Scenario** | **No. BSs** | **BS spacing** | **No. Users** | **Runtime** [s] |
|---|---|---|---|---|
| Dense | 9 | 237 m | 18 | $0.59 \pm 0.02$ |
| Very dense | 16 | 209 m | 32 | $3.29 \pm 0.12$ |
| Ultra dense | 25 | 112 m | 50 | $12.46 \pm 0.35$ |

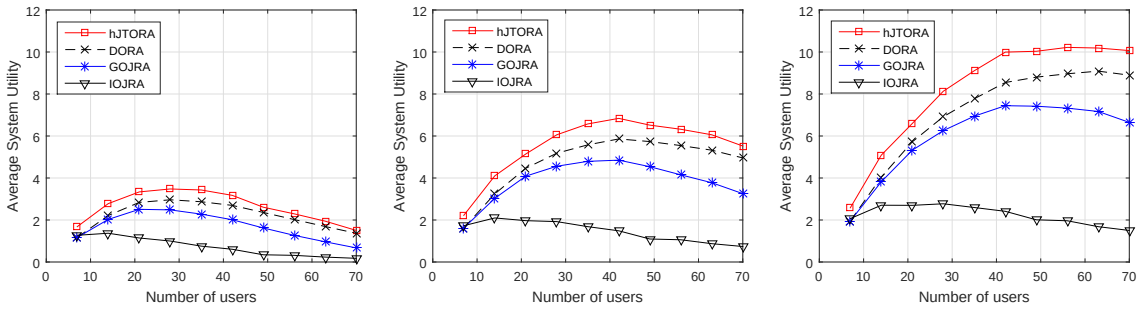



(a) (b)

Figure 5.3: (a) Average number of iterations versus number of users, and (b) Average number of handovers per offloading decision versus number of users; $c_u = 2000$ Megacycles.

with a small and moderate number of users the number of iterations grows linearly (but less than) the number of users and the variation across different simulation runs is small. When the number of users is high, e.g., greater than 60, there is greater variation in the number of iterations but the growing rate of the average number of iteration gets lower. This shows that Algorithm 10 can scale well with the number of users. In Fig. 5.3(b), we plot the average number of handovers per simulation drop. While it can be seen that the average number of handovers (over 500 drops) varies with different number of users, it does not appear to follow a consistent trend.

We now evaluate the system utility performance against different number of users wishing to offload their tasks, as shown in Fig. 5.4(a,b,c) with different task workloads. The number of users per cell is increased from 1 to 10. Note that the number of sub-bands $N$ is set equal to the number of users per cell, thus the bandwidth allocated for each user decreases when there are more users in the system. Observe from Fig. 5.4(a,b,c) that hJTORA always performs the best, and that the performance of all schemes significantly increases




(a) $c_u = 1000$ Megacycles (b) $c_u = 1500$ Megacycles (c) $c_u = 2000$ Megacycles

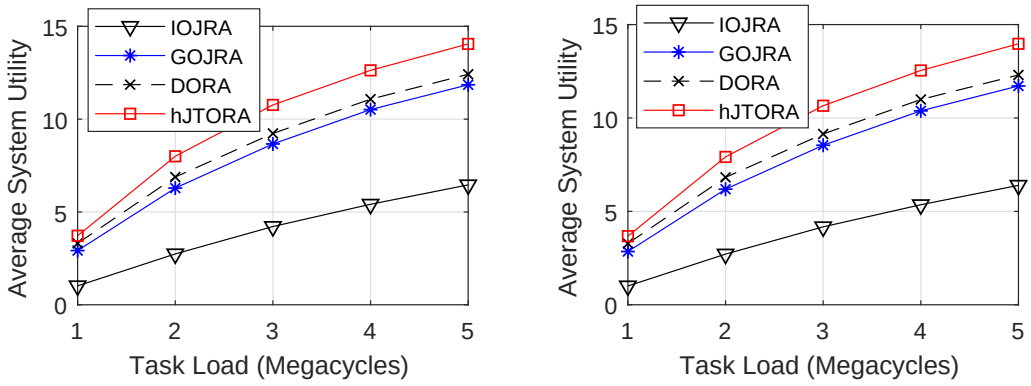
Figure 5.4: Comparison of average system utility against different number of users, evaluated on three different task workloads: (a) $c_u = 1000$ Megacycles, (b) $c_u = 1500$ Megacycles, and (c) $c_u = 2000$ Megacycles, $\forall u \in \mathcal{U}$.

when the tasks' workload increases. This is because when the tasks require more computation resources, the users will benefit more from offloading their tasks to the MEC servers. We also observe that, when the number of users is small, the system utility increases with the number of users; however, when the number of users exceeds some thresholds, the system utility starts to decrease. This is because, when there are many users competing for radio and computing resources for task offloading, the overheads of sending the tasks and executing them at the MEC servers will be higher, thus degrading the offloading utility.

### 5.6.3 Impact of Task Profile

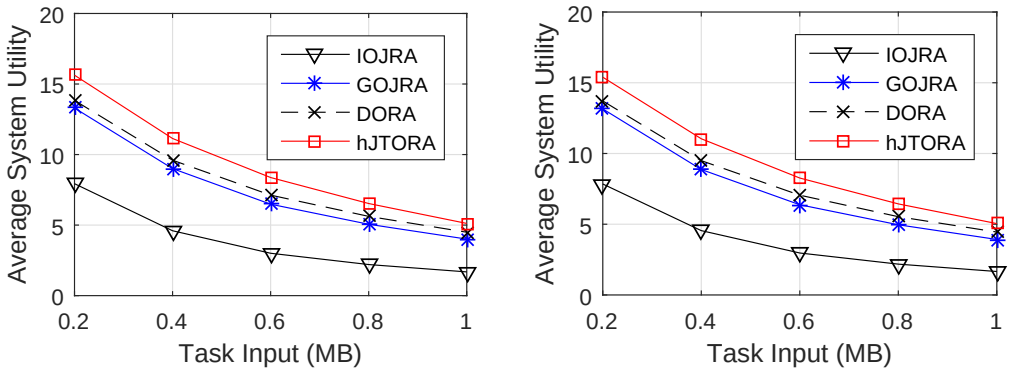
Here, we evaluate the system utility performance w.r.t. the computation tasks' profiles in terms of input size $d_u$'s and workload $c_u$'s. We consider two configuration of the MEC servers: (i) *homogeneous servers*—where all servers have the same CPU speed of 20 GHz, and (ii) *heterogeneous servers*—where the servers' CPU speeds are randomly selected from $\{10, 20, 30\}$ GHz. The average system utility of the four competing schemes are plotted in Fig. 5.5(a, b) with different values of $c_u$; and in Fig. 5.6(a, b) with different values of $d_u$. It can be seen that the average system utilities of all schemes increase with the task workload and decrease with the task input size. This implies that the tasks with small input sizes and high workloads benefit more from offloading than those with large input sizes and low workloads do. Moreover, we observe that the performance gains of the proposed hJTORA scheme over the baselines also follow the similar trend, i.e., they increase with task workloads

and decrease with task input size. Notice that the difference in performance of all schemes in the homogeneous server setting versus in the heterogeneous server setting is marginal.




(a) Homogeneous Servers (b) Heterogeneous Servers

Figure 5.5: Comparison of average system utility against different task workloads, with $U = 28$ and $d_u = 420$ KB.



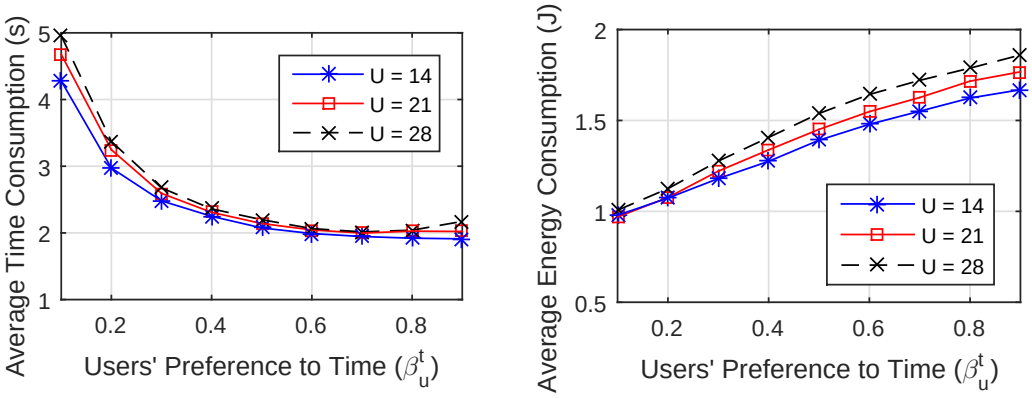

(a) Homogeneous Servers (b) Heterogeneous Servers

Figure 5.6: Comparison of average system utility against different task input size, with $U = 28$ and $c_u = 3000$ Megacycles.

### 5.6.4 Impact of Users' Preferences

Figure 5.7(a,b) show the average time and energy consumption of all the users when we vary the users' preference to time, $\beta_u^t$'s, from 0.1 to 0.9 while changing the users' preference to energy accordingly as $\beta_u^e = 1 - \beta_u^t, \forall u \in \mathcal{U}$. It can be seen that the average time consumption decreases when $\beta_u^t$ increases, at the cost of higher energy consumption. In addition, the users experience a larger average time and energy consumption when there are more users in the system. This is because when there are more users competing for the limited resources,

the chance that a user can benefit from offloading its task is lower.




(a) (b)

Figure 5.7: Average time consumption—(a) and energy consumption—(b) of all users obtained using hJTORA; with $c_u = 2000$ Megacycles, $\forall u \in \mathcal{U}$.

### 5.6.5 Impact of Inter-cell Interference Approximation

To test the effect of the approximation to model the inter-cell interference as in (21) in Sect. 5.5-A, we compare the results of the hJTORA solution to calculate the system utility using the approximated expression versus using the exact expression of the inter-cell interference. Figure 5.8 shows the system utility when the users' maximum transmit power $P_u$'s vary between 0 and 35 dBm. It can be seen that the performance obtained using the approximated interference is almost identical to that of the exact interference when $P_u$ is below 25 dBm, while an increasing gap appears when $P_u > 25$ dBm. However, as specified in the LTE standard, 3GPP TS36.101 section 6.2.3[2], the maximum UE transmit power is 23 dBm; hence, we can argue that our approximation can work well in practical systems.

## 5.7 Summary

We proposed a holistic strategy for a joint task offloading and resource allocation in a multi-cell Mobile-Edge Computing (MEC) network. The underlying optimization problem was formulated as a Mixed-Integer Non-linear Program (MINLP), which is NP-hard. Our approach decomposes the original problem into a Resource Allocation (RA) problem with fixed

[2]Refer to: 3GPP TS36.101, V14.3.0, Mar. 2017

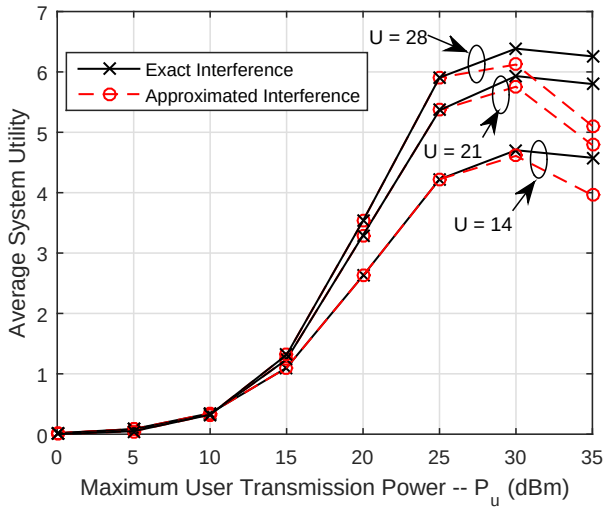



Figure 5.8: Average system utility obtained by hJTORA with exact expression and approximation of the inter-cell interference; with $c_u = 1000$ Megacycles, $\forall u \in \mathcal{U}$.

task offloading decision and a Task Offloading (TO) problem that optimizes the optimal-value function corresponding to the RA problem. We further decouple the RA problem into two independent subproblems, namely the uplink power allocation and the computing resource allocation, and address them using quasi-convex and convex optimization techniques, respectively. Finally, we proposed a novel heuristic algorithm that achieves a suboptimal solution for the TO problem in polynomial time. Simulation results showed that our heuristic algorithm performs closely to the optimal solution and significantly improves the average system offloading utility over traditional approaches.

To further reduce the runtime of the proposed hJTORA algorithm, two approaches can be exploited: (i) *parallel computing*—since the queries in each iteration of Algorithm 10 are independent, they can be run in parallel to reduce the runtime in each iteration; (ii) *pre-disassociation*—for each user, the task offloading variables corresponding to the BSs that are far away can be set to zeros, thus reducing the search space of the task offloading decision before running Algorithm 10. In addition, as future work, it is worth characterizing the approximation ratio of the proposed algorithm with regard to the approximation of the interference term used in the uplink power allocation problem. Furthermore, it is also desirable to design a mistreatment-free solution where each user cannot improve its own performance by not following the network-wide solution.

# Chapter 6

# Conclusion and Future Directions

This chapter summarizes the main contributions of this dissertation and discusses future research directions that are worth investigation and can leverage the frameworks proposed in this dissertation.

## 6.1 Summary of Dissertation Contributions

This dissertation describes novel cooperative frameworks that exploit the synergies between communications, caching, and computing in cloud-assisted wireless networks. The proposed innovations leverage the emerging 5G paradigms—C-RAN and MEC—to design optimized control policies aimed at making best use of the resources available to satisfy the data- and computation-service requests from mobile users. Firstly, given the high degree of cooperation provided by the centralized nature of C-RAN, we proposed a novel joint user-centric radio clustering and beamforming scheme that maximizes the downlink sum throughput. The proposed algorithm, with reasonably low-complexity, was demonstrated to significantly improve the WSR performance over traditional approaches. Secondly, we focused on data offloading mechanisms that take advantage of context and content information to design content caching solutions that decide "what" (content) and "where" to store within the wireless access network. Specifically, we proposed: (i) a cooperative hierarchical caching and request scheduling in a C-RAN that exploits both the vertical collaboration between the edge-caches and the cloud-cache and the horizontal collaboration among the edge-caches to form a heterogeneous cache storage pool; and (ii) a joint cooperative caching and processing framework in a MEC network where the MEC servers perform both cache storage and video transcoding in order to enhance the performance of ABR video streaming. Numerical results showed the significant advantages of the proposed caching solutions in terms of

improvement in cache hit ratio as well as reduction in content access latency and backhaul network usage. Finally, we studied an ultra-dense MEC network where mobile devices can offload computation-intensive tasks to multiple nearby MEC servers. We proposed a cross-layer optimization framework that jointly optimizes the task offloading decision and radio as well as computation resource allocation in order to maximize the computation offloading gain of mobile devices.

## 6.2 Future Directions

The emergence of cloud-assisted wireless networks provides a multitude of benefits to the 5G evolution. Research on the two complementary cloud-assisted wireless network paradigms, C-RAN and MEC, lies at the intersection of wireless communications and cloud computing which has resulted in many interesting research opportunities and challenges. The spectrum of research required to fully achieve the promises of these paradigms in 5G systems require significant investigation along many directions. In the following, we highlight and discuss the key open directions for future research.

### 6.2.1 Edge Caching for User-generated Content Uploading

Edge caching has been recognized as a promising solution, by which popular videos are cached in the BSs or wireless Access Points (APs) so that demands from users to the same content can be accommodated easily without duplicating transmissions from remote servers; this way backhaul usage and content access delay can be substantially reduced. However, traditional caching systems are geared towards supporting data transfer from the servers to users, i.e., focusing on the downlink. They are not meant to support mobile users to upload and share real-time captured/collected data to their peers and to the remote servers. Motivated by this limitation, we propose as future work to design an edge caching strategy to enable the asynchronous upload of user-generated multimedia content in a bandwidth- and energy-efficient manner. The development of such mechanism will bring benefits to both (i) the mobile users, as it reduces the transmission time for users to send their content in the uplink; and (ii) to the network, as it helps smoothen the uplink traffic in peak-traffic

hours or during crowd events by slightly shifting the uploading timing without incurring much extra delay to the delivery of important and time-sensitive content.

There has been a vast body of works focusing on the development of advanced mobile content caching and delivery techniques (see, e.g., [42, 92] and references therein); however, these works rarely address the problem of uploading user-generated content. A recent study [200] shows that the traffic uploading for user-generated content accounts for a large portion of the current Internet traffic and brings new challenges to the network design. In light of this, it is highly desirable to design an edge caching and scheduling mechanism in which mobile users selectively send their content to the edge caching servers instead of directly uploading all of the content to the remote cloud servers. The design and problem at hand differ from the design of CDN in several aspects. Firstly, while the source content of the CDN is static, the content-generating devices in our context are usually mobile; thus it makes the scheduling in the cache pre-fetching phase highly challenging. Secondly, the cache placement in CDN is mostly based on *content popularity*, which can be estimated offline using historical data; however, the notion of content popularity is not relevant for the considered context because the content is generated from mobile users, which are very different from time to time. Therefore, the decision on "what" content to be cached needs to take into account the classification of live content so as to decide (1) which content should be transmitted to the remote server in real-time; and (2) which content is less time-sensitive and could be uploaded at a later time for storage/post-event investigation.

### 6.2.2 Exploiting Collaborative Caching and Multicasting

Due to the broadcast/shared nature of the wireless medium, multicast transmissions can be exploited to serve multiple requests for identical contents occurring in certain time window. This technique has been employed by many network operators to utilize efficiently the available spectrum of their networks. For example, multicast is often used for the delivery of sponsored contents in certain locations such as advertisements, weather updates, stock market reports, and sporting events. Recently, multicast technology has been incorporated in 3GPP LTE standard under the term Evolved Multimedia Broadcast and Multicast

Services (eMBMS) [201]. This technology can be used across multiple cells where the transmissions are carried over the same frequency. As a consequence, the radio resources needed for multicast are a fraction of those needed for multiple unicast transmissions, and thus the network spectral efficiency is greatly enhanced. *Unfortunately, current approaches from academia and industry consider caching and multicast independently as they often need to achieve different purposes.* On the one hand, caching is leveraged to temporally shift the traffic demand from peak to off-peak hours by proactive content provisioning; on the other hand, multicast is used to reduce energy and bandwidth consumption by serving concurrent user requests for the same content via single point-to-multipoint transmission rather than many point-to-point (unicast) transmissions. Intuitively, caching is more effective when there are many duplicated requests for a few content files over time. Conversely, multicast is more effective when there are more concurrent requests for the same contents across multiple users. Such scenarios would be more common in a dense cellular network with a large number of co-located users that are interested in the same contents, such as in sporting events, stock market updates, etc. In the next generation 5G systems, where demands for mobile data is often massive and a variety of new services will arise (such as those using social networking and those employing the one-to-many communication paradigm, e.g., updates in Tweeter, Facebook Live), it is expected that multicast will become very popular.

Given the potential of caching and multicasting, it is interesting to exploit the synergy between the two technologies. Specifically, a joint design of caching and multicasting strategy could enable the possibility of serving users' requests with minimized backhaul usage and spectrum resources. The design problem at hand involves the interplay between three problems: (i) the content popularity prediction and cache placement problem, (ii) the cache-aware multicasting problem, and (iii) the content encryption design, which allows the secure and privacy-preserving implementation of caching within the wireless network.

### 6.2.3 MEC-based Low-latency Smart Healthcare Monitoring

With the emergence of edge-computing paradigm and advanced mobile sensing technologies, *smart* (or *connected*) *healthcare* is getting significant attention from academia, governments, and healthcare industry. It is envisioned that a smart healthcare system will provide greater

accountability, enhance care services, and enable patient monitoring anytime, anywhere, on any device. In such systems, the adoption of emergent mobile sensing solutions coupled with MEC platform can play a key role in supporting reliable and scalable connectivity as well as intelligent services in close-proximity to the patients and caregivers. For instance, healthcare providers can port powerful deep learning and decision-support tools to the MEC servers for assisting clinicians with synthesis of data from multiple sources and low-latency, context-aware decision making. While numerous research efforts have been devoted to improve performance of mobile sensing solutions and healthcare services individually, little attention has been given to make cost-effective and affordable smart healthcare solutions by integrating these technologies and leveraging their synergies. Moreover, to exploit the potential of such integration, there are several challenges that need to be addressed, including: (i) how to exploit the emerging mobile wireless and edge-computing technologies to develop rich and real-time services and applications aiming at assisting patients with the right care, at the right time, and in the right place; and (ii) how to design a network infrastructure that facilitates the representation, storage, integration, and analysis of heterogeneous medical data for effective healthcare solutions. Therefore, it is imperative to develop a wireless networking and computing framework that supports highly connected healthcare environments and to address each of these challenges effectively.

Given the need and potential benefits of smart healthcare, it is a promising approach to design a MEC-based framework that effectively integrates mobile sensing, edge computing, and deep learning to enable a smart healthcare system. In particular, a potential architecture could employ emergent mobile solutions (e.g., novel biosensors and wearable devices) for data collection, MEC platform for data integration and real-time analysis using deep learning, and remote cloud computing for long-term medical data storage and statistical research. The realization of such framework has unprecedented potential for delivering automated, intelligent, and sustainable healthcare services. By exploring the synergies among the three layers—sensing, edge computing, and cloud computing—the envisioned smart healthcare framework will enable the transformation of heterogeneous multimodal raw physiological data into valuable information and then into knowledge that is critical for patients and care givers.

# References


[1] C. V. N. Index, "Global mobile data traffic forecast update, 2015–2020 White Paper," *link: http://goo.gl/ylTuVx*, 2016.

[2] T. Wen and P. Zhu, "5G: A technology vision," *Huawei. http://www. huawei. com/en/abouthuawei/publications/winwin-magazine/hw-329304.htm*, 2013.

[3] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, 2014.

[4] N. Zhang, N. Cheng, A. T. Gamage, K. Zhang, J. W. Mark, and X. Shen, "Cloud assisted HetNets toward 5G wireless networks," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 59–65, 2015.

[5] China Mobile Research Institute, "C-RAN: The Road Towards Green RAN," *White Paper*, Sept. 2013.

[6] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): a primer," *IEEE Network*, vol. 29, no. 1, pp. 35–41, 2015.

[7] S. Bhaumik, S. P. Chandrabose, M. K. Jataprolu, G. Kumar, A. Muralidhar, P. Polakos, V. Srinivasan, and T. Woo, "Cloudiq: a framework for processing base stations in a data center," in *Proc. ACM Annual Int. Conf. Mobile Comput. and Netw. (MobiCom)*, pp. 125–136, 2012.

[8] K. Sundaresan, M. Y. Arslan, S. Singh, S. Rangarajan, and S. V. Krishnamurthy, "Fluidnet: a flexible cloud-based radio access network for small cells," in *Proc. ACM Annual Int. Conf. Mobile Comput. and Netw. (MobiCom)*, pp. 99–110, 2013.

[9] I. Alyafawi, E. Schiller, T. Braun, D. Dimitrova, A. Gomes, and N. Nikaein, "Critical issues of centralized and cloudified LTE-FDD radio access networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 5523–5528, 2015.

[10] N. Nikaein, "Processing radio access network functions in the cloud: Critical issues and modeling," in *Proc. Int. Workshop on Mobile Cloud Computing and Services*, pp. 36–43, ACM, 2015.

[11] P. Chanclou, A. Pizzinat, F. Le Clech, T.-L. Reedeker, Y. Lagadec, F. Saliou, B. Le Guyader, L. Guillo, Q. Deniel, S. Gosselin, *et al.*, "Optical fiber solution for mobile fronthaul to achieve cloud radio access network," in *Proc. Future Network and Mobile Summit*, pp. 1–11, 2013.

[12] Y. Liao and L. Song, "Computing resource constraint in wireless m2m communications," in *Proc. IEEE Int. Wksp. Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–6, 2016.

[13] D. Pompili, A. Hajisami, and T. X. Tran, "Elastic resource utilization framework for high capacity and energy efficiency in Cloud RAN," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 26–32, 2016.

[14] T. X. Tran and D. Pompili, "Dynamic radio cooperation for user-centric cloud-RAN with computing resource sharing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2379–2393, 2017.

[15] A. Hajisami, H. Viswanathan, and D. Pompili, ""Cocktail Party in the Cloud": Blind Source Separation for Co-Operative Cellular Communication in Cloud RAN," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 37–45, 2014.

[16] A. Hajisami and D. Pompili, "Cloud-CFFR: Coordinated fractional frequency reuse in cloud radio access network (c-ran)," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 46–54, 2015.

[17] A. Hajisami, T. X. Tran, and D. Pompili, "Dynamic provisioning for high energy efficiency and resource utilization in Cloud RANs," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 471–472, 2015.

[18] D. Pompili, A. Hajisami, and H. Viswanathan, "Dynamic provisioning and allocation in cloud radio access networks (C-RANs)," *Ad Hoc Networks*, vol. 30, pp. 128–143, 2015.

[19] A. Hajisami and D. Pompili, "DJP: Dynamic joint processing for interference cancellation in cloud radio access networks," in *Proc. IEEE Veh. Technol. Conf. (VTC)*, pp. 1–5, 2015.

[20] A. Hajisami and D. Pompili, "Dynamic joint processing: Achieving high spectral efficiency in uplink 5G cellular networks," *Computer Networks*, vol. 126, pp. 44–56, 2017.

[21] A. Hajisami and D. Pompili, "Joint virtual edge-clustering and spectrum allocation scheme for uplink interference mitigation in C-RAN," *Ad Hoc Networks*, vol. 72, pp. 91–104, 2018.

[22] A. Hajisami, T. X. Tran, and D. Pompili, "Elastic-Net: Boosting Energy Efficiency and Resource Utilization in 5G C-RANs," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 466–470, Oct 2017.

[23] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, 2017.

[24] Intel and Nokia Siemens Networks, "Increasing mobile operators' value proposition with edge computing," *Technical Brief*, 2013.

[25] Saguna and Intel, "Using mobile edge computing to improve mobile network performance and profitability," *White paper*, 2016.

[26] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing-A Key technology towards 5G," *ETSI White Paper*, vol. 11, 2015.

[27] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu, "Concert: a cloud-based architecture for next-generation cellular systems," *IEEE Commun. Mag.*, vol. 21, no. 6, pp. 14–22, 2014.

[28] F. E. Project, "Distributed computing, storage and radio resource allocation over cooperative femtocells (TROPIC)." [Online]: http://www.ict-tropic.eu/, 2012.

[29] S. Wang, G.-H. Tu, R. Ganti, T. He, K. Leung, H. Tripp, K. Warr, and M. Zafer, "Mobile micro-cloud: Application classification, mapping, and deployment," in *Proc. Annual Fall Meeting of ITA (AMITA)*, 2013.

[30] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[31] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, 2015.

[32] Z. Cheng, P. Li, J. Wang, and S. Guo, "Just-in-time code offloading for wearable computing," *IEEE Trans. on Emerging Topics in Computing*, vol. 3, no. 1, pp. 74–83, 2015.

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

[34] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, 2015.

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

[36] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[37] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, 2017.

[38] T. X. Tran and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," *IEEE Trans. Veh. Technol., under revision*, Feb. 2018.

[39] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *Proc. IEEE/IFIP Conf. Wireless On-demand Network Systems and Services (WONS)*, pp. 165–172, 2017.

[40] A. Reznik et al., "Cloud RAN and MEC: A Perfect Pairing," *ETSI White Paper No. 23*, Feb. 2018.

[41] T. X. Tran and D. Pompili, "Dynamic Radio Cooperation for Downlink Cloud-RANs with Computing Resource Sharing," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 118–126, 2015.

[42] T. X. Tran and D. Pompili, "Octopus: A Cooperative Hierarchical Caching Strategy for Cloud Radio Access Networks," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 154–162, Oct. 2016.

[43] T. X. Tran, D. V. Le, G. Yue, and D. Pompili, "Cooperative Hierarchical Caching and Request Scheduling in a Cloud Radio Access Network," *IEEE Trans. Mobile Comput.*, vol. PP, pp. 1–15, Mar. 2018.

[44] T. X. Tran and D. Pompili, "Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks," *IEEE Trans. Mobile Comput., under revision*, Mar. 2018.

[45] D. Lee, H. Seo, B. Clerckx, E. Hardouin, D. Mazzarese, S. Nagata, and K. Sayana, "Coordinated multipoint transmission and reception in LTE-advanced: deployment scenarios and operational challenges," *IEEE Commun. Mag.*, vol. 50, no. 2, 2012.

[46] S. Mosleh, L. Liu, and J. Zhang, "Proportional-Fair Resource Allocation for Coordinated Multi-Point Transmission in LTE-Advanced," *IEEE Trans. Wireless Commun.*, vol. 15, pp. 5355–5367, Aug 2016.

[47] J. Gong, S. Zhou, Z. Niu, L. Geng, and M. Zheng, "Joint scheduling and dynamic clustering in downlink cellular networks," in *Proc. IEEE Global Commun. Conf. (Globecom)*, pp. 1–5, Dec. 2011.

[48] H. Huang, M. Trivellato, A. Hottinen, M. Shafi, P. J. Smith, and R. Valenzuela, "Increasing downlink cellular throughput with limited network MIMO coordination," *IEEE Trans. Wireless Commun.*, vol. 8, no. 6, pp. 2983–2989, 2009.

[49] P. Baracca, F. Boccardi, and V. Braun, "A dynamic joint clustering scheduling algorithm for downlink CoMP systems with limited CSI," in *Proc. IEEE ISWCS*, pp. 830–834, Aug. 2012.

[50] S. Mosleh, J. D. Ashdown, J. D. Matyjas, M. J. Medley, J. Zhang, and L. Liu, "Interference alignment for downlink multi-cell LTE-advanced systems with limited feedback," *IEEE Trans. Wireless Commun.*, vol. 15, no. 12, pp. 8107–8121, 2016.

[51] S. Mosleh, L. Liu, Y. Li, and J. Zhang, "Interference alignment and leakage-based iterative coordinated beam-forming for multi-user MIMO in LTE-advanced," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, 2015.

[52] T. X. Tran, A. Younis, and D. Pompili, "Understanding the Computational Requirements of Virtualized Baseband Units using a Programmable Cloud Radio Access Network Testbed," in *Proc. IEEE Int. Conf. Autonomic Computing (ICAC)*, pp. 221–226, 2017.

[53] E. K. Lee, H. Viswanathan, and D. Pompili, "Vmap: Proactive thermal-aware virtual machine allocation in HPC cloud datacenters," in *Proc. IEEE HiPC*, pp. 1–10, Dec. 2012.

[54] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas in Commun.*, vol. 24, no. 8, pp. 1439–1451, 2006.

[55] A. Liu and V. K. Lau, "Joint power and antenna selection optimization in large cloud radio access networks," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1319–1328, 2014.

[56] B. Dai and W. Yu, "Sparse beamforming and user-centric clustering for downlink cloud radio access network," *IEEE Access*, vol. 2, pp. 1326–1339, 2014.

[57] V. N. Ha, D. H. Nguyen, and L. B. Le, "Sparse precoding design for cloud-RANs sum-rate maximization," in *Proc. IEEE Wireless Commun. and Netw. Conf. (WCNC)*, pp. 1648–1653, 2015.

[58] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical programming*, vol. 95, no. 1, pp. 3–51, 2003.

[59] A. Beck, A. Ben-Tal, and L. Tetruashvili, "A sequential parametric convex approximation method with applications to nonconvex truss topology design problems," *Journal of Global Optimization*, vol. 47, no. 1, pp. 29–51, 2010.

[60] R. Schooler, "Transforming networks with NFV and SDN," *Intel Architecture Group*, 2013.

[61] "Amarisoft LTE software base station." Available: http://www.amarisoft.com/?p=amarilte.

[62] EURECOM, "Open air interface." Available: http://www.openairinterface.org/, Oct. 2014.

[63] C.-L. I, J. Huang, R. Duan, C. Cui, J. Jiang, and L. Li, "Recent Progress on C-RAN Centralization and Cloudification," *IEEE Access*, vol. 2, pp. 1030–1039, 2014.

[64] V. N. Ha, L. B. Le, and N.-D. Dao, "Coordinated multipoint (CoMP) transmission design for Cloud-RANs with limited fronthaul capacity constraints," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7432–7447, 2016.

[65] Y. Shi, J. Zhang, and K. Letaief, "Group sparse beamforming for green Cloud-RAN," *IEEE Trans. Wireless Commun.*, vol. 13, no. 5, pp. 2809–2823, 2014.

[66] S. Luo, R. Zhang, and T. J. Lim, "Downlink and Uplink Energy Minimization Through User Association and Beamforming in C-RAN," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 494–508, 2015.

[67] J. Tang, W. P. Tay, and T. Q. Quek, "Cross-layer resource allocation with elastic service scaling in cloud radio access network," *IEEE Trans. Wireless Commun.*, vol. 14, no. 9, pp. 5068–5081, 2015.

[68] V. N. Ha, L. B. Le, and N.-D. Dao, "Energy-efficient coordinated transmission for cloud-RANs: Algorithm design and trade-off," in *Proc. IEEE Conf. Information Sciences and Systems (CISS)*, pp. 1–6, Mar. 2014.

[69] J. Li, M. Peng, A. Cheng, Y. Yu, and C. Wang, "Resource Allocation Optimization for Delay-Sensitive Traffic in Fronthaul Constrained Cloud Radio Access Networks," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2267–2278, 2017.

[70] T. X. Tran, A. Hajisami, and D. Pompili, "QuaRo: A Queue-Aware Robust Coordinated Transmission Strategy for Downlink C-RANs," in *Proc. IEEE Int. Conf. Sensing, Commun., and Netw. (SECON)*, pp. 1–9, 2016.

[71] T. X. Vu, H. D. Nguyen, and T. Q. Quek, "Adaptive compression and joint detection for fronthaul uplinks in cloud radio access networks," *IEEE Trans. Commun.*, vol. 63, no. 11, pp. 4565–4575, 2015.

[72] S.-H. Park, O. Simeone, O. Sahin, and S. Shamai, "Robust and efficient distributed compression for cloud radio access networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 2, pp. 692–703, 2013.

[73] Z. Yin, F. R. Yu, S. Bu, and Z. Han, "Joint cloud and wireless networks operations in mobile cloud computing environments with telecom operator cloud," *IEEE Trans. Wireless Commun.*, vol. 14, pp. 4020–4033, July 2015.

[74] Y. Liao, L. Song, Y. Li, and Y. A. Zhang, "How much computing capability is enough to run a cloud radio access network?," *IEEE Commun. Letters*, vol. 21, pp. 104–107, Jan 2017.

[75] X. Huang, G. Xue, R. Yu, and S. Leng, "Joint Scheduling and Beamforming Coordination in Cloud Radio Access Networks with QoS Guarantees," *IEEE Trans. Veh. Technol.*, vol. 65, no. 7, pp. 5449–5460, 2016.

[76] L. Tran, M. F. Hanif, A. Tolli, and M. Juntti, "Fast converging algorithm for weighted sum rate maximization in multicell MISO downlink," *IEEE Signal Process. Letters*, vol. 19, no. 12, pp. 872–875, 2012.

[77] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear algebra and its applications*, vol. 284, no. 1, pp. 193–228, 1998.

[78] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[79] Y. Ye, *Interior point algorithms: theory and analysis*, vol. 44. John Wiley & Sons, 2011.

[80] S. K. Joshi, P. C. Weeraddana, M. Codreanu, and M. Latva-Aho, "Weighted sum-rate maximization for MISO downlink cellular networks via branch and bound," *IEEE Trans. Signal Process.*, vol. 60, no. 4, pp. 2090–2095, 2012.

[81] S. Boyd and J. Mattingley, "Branch and bound methods," *Notes for EE364b, Stanford University*, pp. 2006–07, 2007.

[82] V. Balakrishnan, S. Boyd, and S. Balemi, "Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems," *Int. J. of Robust and Nonlinear Control*, vol. 1, no. 4, pp. 295–317, 1991.

[83] O. Tervo, L.-N. Tran, and M. Juntti, "Optimal energy-efficient transmit beamforming for multi-user MISO downlink," *IEEE Trans. Signal Process.*, vol. 63, no. 20, pp. 5574–5588, 2015.

[84] Z. Chen, X. Hou, and C. Yang, "Training resource allocation for user-centric base-station cooperation networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2729–2735, 2016.

[85] 3GPP R1-074068, "Summary of reflector discussions on E-UTRA UL RS," *3GPP TSG RAN WG1 Meeting 50bis*, Oct. 2007.

[86] V. Kotzsch and G. Fettweis, "Interference analysis in time and frequency asynchronous network MIMO OFDM systems," in *Proc. IEEE Wireless Commun. and Netw. Conf. (WCNC)*, pp. 1–6, 2010.

[87] S. Jagannathan, H. Aghajan, and A. Goldsmith, "The effect of time synchronization errors on the performance of cooperative MISO systems," in *Proc. IEEE Global Commun. Conf. (Globecom)*, pp. 102–107, 2004.

[88] S. S. Christensen, R. Agarwal, E. Carvalho, and J. M. Cioffi, "Weighted sum-rate maximization using weighted MMSE for MIMO-BC beamforming design," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, pp. 4792–4799, 2008.

[89] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4331–4340, 2011.

[90] Cisco Visual Networking Index, "Global mobile data traffic forecast update, 2015-2020," *White paper*, 2016.

[91] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1107–1115, 2012.

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

[93] H. Sarkissian, "The business case for caching in 4G LTE networks." LSI-Wireless Technical Report, 2012.

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

[95] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1863–1876, 2016.

[96] H. Ahlehagh and S. Dey, "Hierarchical video caching in wireless cloud: Approaches and algorithms," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 7082–7087, 2012.

[97] L. Gkatzikis, V. Sourlas, C. Fischione, I. Koutsopoulos, and G. Dan, "Clustered content replication for hierarchical content delivery networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 5872–5877, 2015.

[98] F. Pantisano, M. Bennis, W. Saad, and M. Debbah, "In-network caching and content placement in cooperative small cell networks," in *Proc. Int. Conf. on 5G for Ubiquitous Connectivity (5GU)*, pp. 128–133, 2014.

[99] A. Khreishah and J. Chakareski, "Collaborative caching for multicell-coordinated systems," in *Proc. IEEE Conf. on Computer Commun. Workshops (INFOCOM WKSHPS)*, pp. 257–262, 2015.

[100] S. Sesia, I. Toufik, and M. Baker, *LTE: the UMTS long term evolution*. Wiley Online Library, 2009.

[101] J. Robson, "Small cell backhaul requirements," *NGMN White Paper*, pp. 1–40, 2012.

[102] J. Bartelt, P. Rost, D. Wubben, J. Lessmann, B. Melis, and G. Fettweis, "Fronthaul and backhaul requirements of flexibly centralized radio access networks," *IEEE Wireless Commun.*, vol. 22, no. 5, pp. 105–111, 2015.

[103] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2706–2716, 2013.

[104] W. C. Ao and K. Psounis, "An efficient approximation algorithm for online multi-tier multi-cell user association," in *Proc. ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 281–290, 2016.

[105] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, 2014.

[106] W. C. Ao and K. Psounis, "Distributed caching and small cell cooperation for fast content delivery," in *Proc. ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 127–136, 2015.

[107] J. Sung, M. Kim, K. Lim, and J.-K. K. Rhee, "Efficient cache placement strategy in two-tier wireless content delivery network," *IEEE Trans. on Multimedia*, vol. 18, no. 6, pp. 1163–1174, 2016.

[108] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Sel. Areas in Commun.*, vol. 35, pp. 1046–1061, May 2017.

[109] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE J. Sel. Areas in Commun.*, vol. 34, no. 8, pp. 2275–2284, 2016.

[110] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Caching and operator cooperation policies for layered video content delivery," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 874–882, 2016.

[111] P. Ostovari, J. Wu, and A. Khreishah, "Efficient online collaborative caching in cellular networks with multiple base stations," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 136–144, 2016.

[112] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative Multi-bitrate Video Caching and Processing in Mobile-Edge Computing Networks," in *Proc. of the IEEE Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Feb. 2017.

[113] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2995–3007, 2016.

[114] Z. Zhao, M. Peng, Z. Ding, W. Wang, and H. V. Poor, "Cluster content caching: An energy-efficient approach to improve quality of service in cloud radio access networks," *IEEE J. Sel. Areas in Commun.*, vol. 34, no. 5, pp. 1207–1221, 2016.

[115] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud RAN," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6118–6131, 2016.

[116] D. Bethanabhotla, O. Y. Bursalioglu, H. C. Papadopoulos, and G. Caire, "Optimal user-cell association for massive MIMO wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 3, pp. 1835–1850, 2016.

[117] A. Thangaraj and R. Vaze, "Online algorithms for basestation allocation," *IEEE Trans. Wireless Commun.*, vol. 13, no. 5, pp. 2966–2975, 2014.

[118] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1078–1086, 2014.

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

[120] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal routing and content caching in heterogeneous networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 936–944, 2015.

[121] D. A. Soysa, D. G. Chen, O. C. Au, and A. Bermak, "Predicting YouTube content popularity via Facebook data: A network spread model for optimizing multimedia delivery," in *Proc. of IEEE Symp. on Computational Intelligence and Data Mining (CIDM)*, pp. 214–221, 2013.

[122] E. Baştuğ, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *IEEE J. Commun. Netw.*, vol. 17, no. 6, pp. 549–557, 2015.

[123] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[124] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions - I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[125] L. A. Wolsey and G. L. Nemhauser, *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[126] J. Ribas-Corbera, P. A. Chou, and S. L. Regunathan, "A generalized hypothetical reference decoder for h. 264/avc," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 674–687, 2003.

[127] W. C. Ao and K. Psounis, "Approximation algorithms for online user association in multi-tier multi-cell mobile networks," *IEEE/ACM Trans. Netw.*, 2017.

[128] B. Lehmann, D. Lehmann, and N. Nisan, "Combinatorial auctions with decreasing marginal utilities," in *Proc. 3rd ACM Conf. Electron. Commerce*, pp. 18–28, 2001.

[129] http://traces.cs.umass.edu/index.php/Network.

[130] S. Borst, V. Gupt, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1–9, 2010.

[131] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1357–1370, 2009.

[132] D. Lee, J. Choi, J. H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, 2001.

[133] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 126–134, 1999.

[134] L. A. Adamic and B. A. Huberman, "Zipf's law and the Internet," *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[135] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4, pp. 333–344, 2006.

[136] A. Mahanti, C. Williamson, and D. Eager, "Traffic analysis of a web proxy caching hierarchy," *IEEE Network*, vol. 14, no. 3, pp. 16–23, 2000.

[137] Cisco Visual Networking Index, "Global mobile data traffic forecast update, 2014–2019," *White Paper c11-520862*, 2014.

[138] T. Stockhammer, "Dynamic adaptive streaming over http–: standards and design principles," *In Proc. Annual ACM Conference on Multimedia Systems*, pp. 133–144, 2011.

[139] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proc. Annual ACM Conference on Multimedia Systems*, pp. 157–168, 2011.

[140] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Process. Mag.*, vol. 20, no. 2, pp. 18–29, 2003.

[141] H. A. Pedersen and S. Dey, "Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 996–1010, 2016.

[142] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," *in Proc. IEEE INFOCOM WKSHPS*, pp. 310–315, 2012.

[143] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Telecom ParisTech Relatório técnico*, 2011.

[144] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," *in Proc. Workshop on Information-Centric Networking (ICN)*, pp. 55–60, 2012.

[145] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 2426–2434, 2012.

[146] M. Hajimirsadeghi, N. B. Mandayam, and A. Reznik, "Joint caching and pricing strategies for popular content in information centric networks," *IEEE J. Sel. Areas in Commun.*, vol. 35, no. 3, pp. 654–667, 2017.

[147] M. Hajimirsadeghi, N. B. Mandayam, and A. Reznik, "Joint caching and pricing strategies for information centric networks," in *Proc. IEEE Global Commun. Conf. (Globecom)*, pp. 1–6, 2015.

[148] T. X. Tran, A. Hajisami, and D. Pompili, "Cooperative hierarchical caching in 5G cloud radio access networks," *IEEE Network*, vol. 31, no. 4, pp. 35–41, 2017.

[149] S. Mosleh, L. Liu, H. Hou, and Y. Yi, "Coordinated Data Assignment: A Novel Scheme for Big Data Over Cached Cloud-RAN," in *Proc. IEEE Global Commun. Conf. (Globecom)*, Dec. 2016.

[150] T. X. Tran, F. Kazemi, E. Karimi, and D. Pompili, "Mobee: Mobility-Aware Energy-Efficient Coded Caching in Cloud Radio Access Networks," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 461–465, 2017.

[151] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Edge-caching wireless networks: Performance analysis and optimization," *IEEE Trans. Wireless Commun.*, vol. PP, no. 99, 2018.

[152] Apple Inc., "HTTP Live Streaming (HLS)." [Online]: https://developer.apple.com/streaming/, 2018.

[153] A. Zambelli, "IIS smooth streaming technical overview," *Microsoft Corporation*, pp. 1–17, Mar. 2009.

[154] D. Bhat, A. Rizk, M. Zink, and R. Steinmetz, "Network assisted content distribution for adaptive bitrate video streaming," in *Proc. 8th ACM Conf. Multimedia Systems*, pp. 62–75, 2017.

[155] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in HTTP adaptive streaming: Friend or foe?," in *Proc. ACM Wksh. Network and Operating System Support on Digital Audio and Video*, pp. 31–36, ACM, 2014.

[156] P. Ostovari, A. Khreishah, and J. Wu, "Multi-layer video streaming with helper nodes using network coding," in *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems (MASS)*, pp. 524–532, 2013.

[157] R. Yu, S. Qin, M. Bennis, X. Chen, G. Feng, Z. Han, and G. Xue, "Enhancing software-defined RAN with collaborative caching and scalable video coding," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1–6, 2016.

[158] B. Shen, S.-J. Lee, and S. Basu, "Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 375–386, 2004.

[159] H. Ahlehagh and S. Dey, "Adaptive bit rate capable video caching and scheduling," in *Proc. IEEE Wireless Commun. and Netw. Conf. (WCNC)*, pp. 1357–1362, 2013.

[160] H. Ahlehagh, L. Toni, D. Wang, P. Cosman, S. Dey, and L. Milstein, "Caching and cross-layer design for enhanced video performance," *Intel Technology Journal*, vol. 19, no. 1, pp. 70–91, 2015.

[161] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman and Company, 1979.

[162] Y. Azar and I. Gamzu, "Efficient submodular function maximization under linear packing constraints," in *International Colloquium on Automata, Languages, and Programming*, pp. 38–50, Springer, 2012.

[163] A. Srinivasan, "Improved approximation guarantees for packing and covering integer programs," *SIAM Journal on Computing*, vol. 29, no. 2, pp. 648–670, 1999.

[164] A. Mosek, "The mosek optimization toolbox for matlab manual," *Version 7.1 (Revision 28)*, p. 17, 2015.

[165] A. Marchetti-Spaccamela and C. Vercellis, "Stochastic on-line knapsack problems," *Mathematical Programming*, vol. 68, no. 1-3, pp. 73–104, 1995.

[166] X. Li, X. Wang, S. Xiao, and V. C. Leung, "Delay performance analysis of cooperative cell caching in future mobile networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 5652–5657, 2015.

[167] M. Chen, W. Saad, and C. Yin, "Virtual reality over wireless networks: Quality-of-service model and learning-based resource management," *[available online]: arxiv.org/abs/1703.04209*, Mar. 2017.

[168] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *[available online]: arxiv.org/abs/1710.02913*, Oct. 2017.

[169] T. X. Tran, M.-P. Hosseini, and D. Pompili, "Mobile edge computing: Recent efforts and five key research directions," *IEEE COMSOC MMTC Commun.-Frontiers*, 2017.

[170] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *in Proc. IEEE Symp. Computers and Communications (ISCC)*, pp. 59–66, 2012.

[171] K. Bilal, A. Erbad, and M. Hefeeda, "Crowdsourced multi-view live video streaming using cloud computing," *IEEE Access*, vol. 5, pp. 12635–12647, 2017.

[172] M.-P. Hosseini, T. X. Tran, D. Pompili, K. Elisevich, and H. Soltanian-Zadeh, "Deep learning with edge computing for localization of epileptogenicity using multimodal rs-fMRI and EEG big data," in *Proc. IEEE Int. Conf. Autonomic Computing (ICAC)*, pp. 83–92, 2017.

[173] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, 2015.

[174] V. Cardellini, V. D. N. Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. L. Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, vol. 157, no. 2, pp. 421–449, 2016.

[175] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, 2016.

[176] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, 2013.

[177] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "Music: Mobility-aware optimal service allocation in mobile cloud computing," in *Proc. IEEE Int. Conf. on Cloud Computing*, pp. 75–82, 2013.

[178] X. Lyu, H. Tian, P. Zhang, and C. Sengul, "Multi-user joint task offloading and resources optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, 2017.

[179] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5G ultra-dense cellular networks," *IEEE Wireless Commun.*, vol. 23, no. 1, pp. 72–79, 2016.

[180] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, 2017.

[181] J. O. Fajardo, I. Taboada, and F. Liberal, "Improving content delivery efficiency through multi-layer mobile edge adaptation," *IEEE Network*, vol. 29, no. 6, pp. 40–46, 2015.

[182] Nokia, "LTE and Car2x: Connected cars on the way to 5G." [Online]: http://www.cambridgewireless.co.uk/Presentation/MB06.04.16-Nokia-Uwe Putzschler.pdf.

[183] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[184] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, 2014.

[185] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 190–194, 2013.

[186] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 64–71, 2017.

[187] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1–9, 2017.

[188] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX Conf. Hot Topics Cloud Comput. (HotCloud)*, June 2010.

[189] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 2716–2720, 2012.

[190] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013.

[191] W. Saad, Z. Han, R. Zheng, M. Debbah, and H. V. Poor, "A college admissions game for uplink user association in wireless small cell networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1096–1104, 2014.

[192] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*. Springer Science & Business Media, 2006.

[193] T. X. Tran, N. H. Tran, H. R. Bahrami, and S. Sastry, "On achievable rate and ergodic capacity of NAF multi-relay networks with CSI," *IEEE Trans. Commun.*, vol. 62, no. 5, pp. 1490–1502, 2014.

[194] K. Tammer, "The application of parametric optimization and imbedding to the foundation and realization of a generalized primal decomposition approach," *Mathematical research*, vol. 35, pp. 376–386, 1987.

[195] Y. Du and G. De Veciana, ""Wireless networks without edges": Dynamic radio resource clustering and user scheduling," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 1321–1329, 2014.

[196] B. Bereanu, "Quasi-convexity, strictly quasi-convexity and pseudo-convexity of composite objective functions," *Revue française d'automatique, informatique, recherche opérationnelle. Mathématique*, vol. 6, no. 1, pp. 15–26, 1972.

[197] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in *Proc. Annual ACM Symp. Theory of Comput.*, pp. 323–332, 2009.

[198] X. Chu, D. López-Pérez, Y. Yang, and F. Gunnarsson, *Heterogeneous Cellular Networks: Theory, Simulation and Deployment*. Cambridge University Press, 2013.

[199] Nokia White Paper, "Ultra Dense Network (UDN)." [Online]: https://onestore.nokia.com/asset/200295/Nokia_Ultra_Dense_Network_White_Paper_EN.pdf, 2016.

[200] B. Ager, F. Schneider, J. Kim, and A. Feldmann, "Revisiting cacheability in times of user generated content," in *in Proc. IEEE Conf. Computer Commun. (INFOCOM) Workshops*, pp. 1–6, 2010.

[201] D. Lecompte and F. Gabin, "Evolved multimedia broadcast/multicast service (eMBMS) in LTE-advanced: overview and Rel-11 enhancements," *IEEE Commun. Mag.*, vol. 50, no. 11, 2012.